UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

RICONOSCIMENTO E VALUTAZIONE DELLA QUALITÀ DI ESECUZIONE DELLE ATTIVITÀ QUOTIDIANE PER APPLICAZIONI DI HEALTHCARE

Relatore: Dott. Daniele RIBONI

Correlatore: Prof. Claudio BETTINI

Tesi di laurea di:

Gabriele CIVITARESE

Matricola 826031

ANNO ACCADEMICO 2013–14

A famiglia ed amici

Indice

Introduzione **1**

1 Stato dell'arte **2**

1.1 Sorgenti di dati 2

1.1.1 Informazioni visuali 2

1.1.2 Sensori 2

1.2 Tecniche di riconoscimento di attività 3

1.2.1 Tecniche statistiche 3

1.2.2 Tecniche simboliche 4

1.2.3 Tecniche ibride 4

1.3 Tecniche di riconoscimento di anomalie 4

2 Collocazione della tesi nell'ambito del progetto *SECURE* **6**

2.1 Collocazione e obiettivi del lavoro di tesi 6

2.2 Architettura 7

2.2.1 Nodi sensorizzati 7

2.2.2 Gateway 8

2.2.3 Applicazione mobile 9

Riconoscimento di inizio e fine di attività 9

Raffinamento delle attività riconosciute 9

Riconoscimento di anomalie 9
2.3 Acquisizione di un dataset di attività e anomalie 9
2.3.1 Annotazione dei dati 9
2.3.2 Acquisizione di un dataset di attività guidate 9
2.3.3 Acquisizione di un dataset di attività spontanee 9
2.4 Considerazioni sui sensori utilizzati 9
2.4.1 Sensore Hall Effect *PLA41201* 9
2.4.2 RFID 10
Problematiche 10
Assunzioni 11
2.4.3 Sensore di temperatura *MCP9700A* 11
Problematiche 11
2.4.4 Sensore di pressione *Flexiforce PS-02* 11
Problematiche 11
2.4.5 PIR 11
Problematiche 11
2.4.6 Ambiente di acquisizione 11

3 Riconoscimento di attività e di anomalie 13
3.1 La logica probabilistica *Markov Logic Network* 13
3.1.1 Modelli grafici probabilistici 13
3.1.2 *Markov Random Field* 13
3.1.3 Logica del primo ordine 13
Esempio 14
3.1.4 *Markov Logic Network* 14
Apprendimento generativo 15
Apprendimento discriminativo 15

Apprendimento strutturale 15
Inferenza probabilistica 15
Inferenza *MAP* 16
3.1.5 Esempio 16
3.2 Riconoscimento di attività 16
3.2.1 Costruzione del modello 17
Predicati osservabili 17
Predicati nascosti 17
Regole 18
3.2.2 Raffinamento delle attività riconosciute 18
3.3 Riconoscimento di anomalie 19
3.3.1 Costruzione del modello 19
Rappresentazione delle attività 19
Rappresentazione di informazioni aggiuntive 19
Rappresentazione dello stato di un oggetto 20
Rappresentazione delle anomalie 20

4 Valutazione sperimentale **21**
4.1 Software per *Markov Logic Network* 21
4.1.1 *Alchemy* 21
4.1.2 *thebeast* 21
4.1.3 *Tuffy* 21
4.1.4 *RockIt* 21
4.2 Valutazione del riconoscimento delle attività 22
4.2.1 Metriche 22
4.2.2 Risultati sperimentali 22
4.2.3 Conclusioni 27

4.3 Valutazione del riconoscimento delle anomalie 27
4.3.1 *tuProlog* 28
4.3.2 Anomalie riconosciute 29
4.3.3 Risultati sperimentali 29
4.4 Software realizzato 30
4.4.1 Integrazione semantica dei dati di sensori 30
4.4.2 Riconoscimento e raffinamento di attività 30
4.4.3 Riconoscimento di anomalie 31
4.4.4 Applicazione *Android* 31

Conclusioni e sviluppi futuri **33**

Bibliografia **33**

Introduzione

I recenti sviluppi in ambito medico hanno permesso alle persone di vivere più a lungo rispetto alle generazioni precedenti. È previsto che nel 2050 il 20% della popolazione avrà un'età superiore ai 60 anni [1]. Come effetto collaterale, l'invecchiamento implica nuove complicanze in termini di salute: deterioramento cognitivo, malattie croniche e svariate limitazioni (ad esempio in attività fisiche, vista, udito, eccetera). Al crescere del numero di persone affette da queste problematiche, aumentano anche il costo dell'assistenza sanitaria e la carenza di personale specializzato per lavorare con questo tipo di pazienti. Inoltre, sempre più individui perdono la capacità di poter vivere in maniera autonoma. Tutti questi problemi presentano un grosso impatto sulla società contemporanea. Una delle forme di assistenza sanitaria richieste è l'analisi delle azioni quotidiane svolte da pazienti anziani o affetti da disabilità per comprenderne il livello di menomazione fisico e psichico. L'approccio attualmente utilizzato prevede la compilazione di questionari (da parte di familiari o di personale infermieristico specializzato) il cui scopo è quello di aiutare i medici a determinare l'indipendenza di una persona analizzando le attività che riescono ad essere svolte in autonomia (ad esempio: guidare, pulire, cucinare, lavarsi, vestirsi, ...). Tuttavia, queste metodologie presentano una significativa carenza in termini di accuratezza, efficienza, costi e privacy. Per questi motivi, un sistema automatico di monitoraggio e di analisi del comportamento della vita quotidiana aiuterebbe a ridurre i costi e a migliorare l'accuratezza delle osservazioni. Negli ultimi anni molti lavori di ricerca hanno contribuito in questo am-

bito studiando diverse tecniche. L'analisi del comportamento umano necessita di un modello di riconoscimento volto a rilevare una vasta varietà di attività svolte in diversi modi ed in diverse condizioni ambientali, e a valutare la qualità di esecuzione delle stesse [2]. Per realizzarlo è necessario disporre dell'apparecchiatura per il monitoraggio dei soggetti e dell'ambiente, definire un modello delle attività da riconoscere (che permetta di rappresentare relazioni complesse tra i dati di contesto) ed essere in grado di elaborare le informazioni acquisite tenendo conto del fatto che sono caratterizzate da incertezza.

Il presente lavoro si è svolto nell'ambito del progetto *SECURE* [1], il cui scopo è quello di realizzare un sistema intelligente per la telesorveglianza domiciliare di pazienti con lo scopo di effettuare una diagnosi di deterioramento cognitivo. L'obiettivo della tesi è stato la costruzione di un modello di riconoscimento di attività quotidiane svolte all'interno di un ambiente domestico (come ad esempio l'assunzione di medicinali, la preparazione del pranzo, eccetera) e la raccolta di un dataset di attività e anomalie svolte in un ambiente ricco di sensori, al fine di valutare l'efficacia del sistema di rilevamento di anomalie. A tale scopo è stata definita una tecnica basata sul monitoraggio dell'interazione dei pazienti con gli oggetti domestici (ad esempio: frigorifero, fornello, armadietto delle medicine, ...) utilizzando sensori di ambiente situati nelle loro abitazioni. Vengono quindi analizzate le rilevazioni di sensori acquisite, in modo inferire le attività svolte deducendo gli istanti di inizio e di fine delle stesse. Le attività riconosciute vengono successivamente analizzate per cercare di evidenziare comportamenti anomali. Per il riconoscimento delle attività viene valutato l'utilizzo della logica probabilistica *Markov Logic Network*, un sistema ibrido che unisce ragionamento simbolico e ragionamento statistico. Una *Markov Logic Network* è una base di conoscenza della logica del primo ordine dove ad ogni formula è associato un peso

[1] http://secure.ewlab.di.unimi.it/

(impostato durante una fase di apprendimento utilizzando un opportuno *training set*), che viene tradotta in una *rete di Markov* per effettuare inferenza. È stata quindi modellata e validata una *Markov Logic Network* in grado di inferire le attività svolte dai pazienti partendo dalle misurazioni dei sensori. Il modulo riconoscimento di anomalie è invece realizzato con un sistema a regole che, attraverso dei ragionamenti di tipo temporale codificati usando formule della logica del primo ordine, analizza le attività riconosciute identificando eventuali comportamenti anomali. I modelli di riconoscimento appena descritti vengono integrati in un'applicazione *Android* installata su un dispositivo personale del paziente in grado di raccogliere le informazioni dei sensori, riconoscere attività ed anomalie ed inviare le informazioni prodotte ad un centro di telemedicina per essere visualizzate dai medici. Per poter costruire e validare i modelli di riconoscimento è necessario avere a disposizione un dataset di attività ed anomalie. In assenza di pazienti reali è stato necessario simulare (previa pianificazione dettagliata tramite la collaborazione con esperti di un centro medico) l'esecuzione di attività ed anomalie all'interno di un ambiente sensorizzato per ottenere dei dati su cui lavorare. Il dataset acquisito contiene misurazioni di sensore acquisite simulando il comportamento di tre pazienti: uno sano, uno affetto da deterioramento cognitivo lieve e uno affetto da deterioramento cognitivo grave. Per ogni paziente sono state simulate attività ed anomalie per una settimana. Questo dataset è stato usato per selezionare il modello da utilizzare per il riconoscimento di attività tramite un opportuno processo di *Cross Validazione* e per validare le regole utilizzate nel modello di riconoscimento di anomalie. I risultati ottenuti sono decisamente buoni: nella maggioranza dei casi entrambi i modelli di riconoscimento riescono a predire correttamente le attività e le anomalie presenti nei dati. Una possibile direzione di ricerca futura consiste nel rendere il sistema di riconoscimento di anomalie probabilistico, in modo da ottenere un modello di riconoscimento più flessibile. Un altro possibile sviluppo consiste nel riconoscere attività che vengono eseguite in parallelo, e non in stretta sequenza come

proposto in questo lavoro.

Gli specifici contributi apportati all'attività progettuale appena illustrata sono stati:

- L'acquisizione e l'annotazione di un dataset di attività ed anomalie.
- La modellazione e la validazione del sistema di riconoscimento di attività.
- La collaborazione nella realizzazione del sistema di riconoscimento di anomalie.
- L'integrazione in un'applicazione *Android* dei moduli di riconoscimento realizzati.

I prossimi capitoli sono organizzati nel modo seguente. Nel primo capitolo viene presentato lo stato dell'arte nell'area di ricerca relativa al riconoscimento automatico di attività umane e la valutazione di comportamenti anomali, considerando le diverse tecniche presenti in letteratura. Il secondo capitolo si concentra sul progetto *SECURE*, analizzando l'architettura generale del sistema realizzato e le tecnologie sensoristiche utilizzate. Nel terzo capitolo vengono illustrati gli strumenti matematici utilizzati per il riconoscimento di attività e di anomalie, formalizzando la costruzione dei modelli realizzati. Nel quarto capitolo viene presentata la valutazione sperimentale analizzando: cosa è stato implementato, i prodotti software utilizzati ed i risultati sperimentali dei modelli di riconoscimento. Infine sono sintetizzati i punti di forza ed i problemi aperti del lavoro ed indicate le possibili direzioni di ricerca per il suo miglioramento.

Capitolo 1

Stato dell'arte

Il riconoscimento automatico delle attività umane è un processo tramite cui il comportamento di un attore e l'ambiente in cui è situato vengono monitorati e analizzati per inferire le attività svolte [3]. Questo campo di ricerca è oggetto di un sempre maggiore interesse, presentando una delle principali sfide per la realizzazione di sistemi pervasivi. L'abilità di riconoscere le azioni svolte da un utente interagendo con l'ambiente circostante offre enormi benefici: permette di realizzare applicazioni in grado di reagire e adattarsi al comportamento dell'utente. Negli ultimi anni, sempre più gruppi di ricerca stanno focalizzando la loro attenzione sullo sviluppo di tecniche di riconoscimento di anomalie all'interno delle attività svolte dagli utenti, cercando di evidenziare rilevanti variazioni del comportamento rispetto a quello atteso. Il riconoscimento di comportamenti anomali è particolarmente utile in applicazioni di assistenza medica: è per esempio possibile generare allarmi all'occorrenza di anomalie, o effettuare diagnosi precoci di svariate problematiche.

In Sezione 1.1 vengono introdotte le possibili tipologie di sorgenti di dati per il monitoraggio degli attori, in modo tale da ottenere informazioni rilevanti per il riconoscimento di attività ed anomalie. La Sezione 1.2 presenta alcune delle tecniche presenti

in letteratura per il riconoscimento automatico di attività, mentre la Sezione 1.3 quelle per il riconoscimento di anomalie.

1.1 Sorgenti di dati

In base alla tipologia di dati che si vogliono raccogliere, esistono due approcci principali per il monitoraggio di un attore e dell'ambiente in cui è situato: uno basato sull'analisi di informazioni visuali e uno basato sull'analisi delle informazioni prodotte da sensori.

1.1.1 Informazioni visuali

Questo approccio utilizza tecniche di computer vision con lo scopo di monitorare il comportamento di uno o più attori e la relativa interazione con l'ambiente. Molte modalità di acquisizione sono state sperimentate, come ad esempio videocamere singole, stereoscopiche, ad infrarossi, eccetera [3]. Il processo computazionale tipico prevede questi passi:

- Rilevamento di oggetti (o di persone)
- Tracciamento dei movimenti
- Riconoscimento di attività
- Valutazione ad alto livello delle attività riconosciute

Data la complessità del mondo reale, questo approccio soffre di problemi relativi alla scalabilità e alla riusabilità. In aggiunta, le videocamere usate come dispositivo di registrazione sono percepite come invasive. Questo impedisce l'utilizzo di questo approccio in molte applicazioni (come ad esempio gli ambienti domestici).

1.1.2 Sensori

Questo approccio sfrutta la tecnologia emergente delle reti di sensori per monitorare il comportamento di un attore e il suo ambiente. I dati collezionati dai sensori vengono analizzati tipicamente tramite tecniche di data mining e machine learning per la costruzione di modelli di attività e di riconoscimento di pattern. In questo approccio i sensori possono essere indossati direttamente dall'attore che si intende osservare (sensori indossabili) o integrati in oggetti facenti parte dell'ambiente (sensori di ambiente).

Sensori indossabili

I sensori indossati dagli attori da monitorare possono avere differenti scopi:

- Rilevazione di movimenti (ad esempio accelerometri, giroscopi, magnetometri, ...)
- Rilevazione di segnali vitali (battito cardiaco, temperatura, ...)

L'approccio basato su sensori indossabili risulta efficace per il riconoscimento di alcuni tipi di attività, principalmente per catturare i movimenti fisici umani. Tuttavia, soffre di alcune problematiche. Prima di tutto, svariati problemi tecnici relativi ai sensori (ad esempio dimensione, facilità di utilizzo, durata della batteria ...) determinano problematiche di accettabilità e di volontà di utilizzo da parte degli attori che dovrebbero indossarli. Inoltre, svariate attività nel mondo reale comprendono complessi movimenti e complesse interazione con l'ambiente. L'utilizzo esclusivo di informazione proveniente da sensori indossabili può non essere sufficiente per differenziare attività che consistono in semplici movimenti.

Sensori di ambiente

L'utilizzo di sensori di ambiente è stato particolarmente investigato per la creazione di ambienti intelligenti volti a supportare l'utente nelle attività della vita quotidiana. Alcuni esempi di questo tipo di sensori sono: sensore di temperatura, sensore per il monitoraggio di apertura/chiusura di porte (o sportelli), sensore di presenza, eccetera. Tali sensori possono essere usati anche per effettuare il monitoraggio sanitario di persone (come ad esempio anziani o pazienti in riabilitazione) tramite sistemi di reti di sensori all'interno degli spazi abitativi. Lo scopo principale di questo approccio è quello di rilevare l'informazione sull'interazione del paziente monitorato con gli oggetti presenti nel suo ambiente domestico, in modo tale da poter inferire le attività che vengono svolte al fine di generare allarmi o di effettuare diagnosi precoci di svariate problematiche.

1.2 Tecniche di riconoscimento di attività

Una volta stabilita la sorgente dei dati, è necessario scegliere le modalità di modellazione, rappresentazione e riconoscimento di attività. In letteratura sono presenti tre filoni principali:

- Tecniche che sfruttano metodi di machine learning e data mining per effettuare ragionamento di tipo statistico e probabilistico.
- Tecniche simboliche che sfruttano formalismi logici per modellare le attività e processare i dati provenienti dalle sorgenti.
- Tecniche ibride che combinano i due approcci precedenti in modo da unirne i punti di forza.

In questa sezione vengono presentate alcuni dei principali modelli di riconoscimento di attività umane per queste tipologie di tecniche.

1.2.1 Tecniche statistiche

Le tecniche di tipo statistico per il riconoscimento automatico di attività si concentrano principalmente su metodi di apprendimento supervisionato e non supervisionato. L'apprendimento supervisionato richiede l'utilizzo di dati etichettati sui quali viene allenato un algoritmo, in modo tale da produrre un classificatore in grado di etichettare nuovi dati. Lo svantaggio di questo metodo è che richiede un grossa quantità di dati etichettati per l'apprendimento (la creazione di un insieme di dati manualmente etichettati richiede un elevato costo in termini temporali). L'apprendimento non supervisionato, invece, cerca di costruire modelli di riconoscimento a partire da dati non etichettati, assegnando manualmente una probabilità ad ogni attività e predefinendo un processo stocastico in grado di aggiornare queste probabilità in base a nuove osservazioni e allo stato del sistema. Nella maggior parte dei casi, tuttavia, le tecniche di apprendimento supervisionato si sono dimostrate più adeguate.

Il punto di forza delle tecniche statistiche è che generalmente sono in grado di gestire il rumore e l'incertezza intrinseca nei dati raccolti tramite sensori. Inoltre, le probabilità possono essere usate per catturare euristiche del dominio (ad esempio se un'attività è più probabile di un'altra) [3]. Vengono ora descritte alcune delle principali tecniche di apprendimento supervisionato utilizzate in letteratura per il riconoscimento delle attività.

Hidden Markov Model

Un *Hidden Markov Model* (*HMM*) è un modello statistico in cui si assume che il sistema modellato sia una catena di Markov rappresentante una sequenza di eventi. Una catena di Markov è un processo stocastico con spazio degli stati discreto tale che la probabilità di transizione verso un nuovo stato del sistema dipende unicamente dallo stato immediatamente precedente e non dall'intera sequenza di stati. Un *HMM* è composto da un insieme finito di stati nascosti che evolvono secondo una catena di Markov e da un insieme di osservazioni generate dagli stati secondo una distribuzione di probabilità. Un *HMM* è costruito sulle seguenti assunzioni:

- Ogni stato dipende solo dal predecessore (seguente dalla definizione di catena di Markov)
- Ogni variabile osservata dipende solo dallo stato corrente
- Le osservazioni sono indipendenti tra di loro

Nel campo del riconoscimento delle attività, gli stati nascosti corrispondono alle attività svolte dagli attori considerati, mentre ogni osservazione è costituita dai valori letti dai sensori. Un *HMM* è addestrato con un training set per derivare tre parametri:

1. Probabilità iniziale degli stati (la probabilità che un attore cominci dall'attività corrispondente allo stato considerato)
2. Probabilità di transizione da uno stato all'altro (per ogni coppia di attività, rappresenta la probabilità che l'attore passi da un'attività all'altra).
3. Probabilità di emissione di un'osservazione dato uno stato (la probabilità che si verifichi un osservazione data l'attività svolta)

Il riconoscimento di attività avviene inferendo la sequenza di stati più probabile in base alle osservazioni [4]. In [5] viene proposto un modello di riconoscimento di attività a partire da misurazioni di sensori ambientali basato su *HMM*.

Conditional Random Field

Un *Conditional Random Field* (*CRF*) è un modello grafico probabilistico non diretto (rappresentato da un grafo non diretto) volto a modellare la probabilità condizionale di una sequenza di stati nascosti data una sequenza di osservazioni. Modellando la probabilità condizionale invece di quella congiunta (come nel caso di *HMM*) è possibile utilizzare feature più complesse ed eliminare l'assunzione di indipendenza tra le osservazioni. Le altre assunzioni che sono state fatte per *HMM*, invece, continuano a valere. Per calcolare la probabilità condizionale, l'inferenza avviene in termini di *clique* presenti nel grafo: la probabilità di una configurazione di variabili è proporzionale al prodotto di una serie di funzioni potenziali non-negative (una per ogni *clique* del grafo). Intuitivamente, ogni funzione potenziale computa la probabilità che le variabili nella *clique* corrispondente abbiano una determinata configurazione. L'apprendimento di un *CRF* avviene stimando i valori di queste funzioni potenziali massimizzando la loro probabilità condizionale dato un determinato *training set* [6]. Come abbiamo visto per *HMM*, è possibile effettuare riconoscimento di attività rappresentando le attività svolte tramite stati nascosti e le misurazioni di sensori tramite osservazioni. In [5] viene proposto un modello di riconoscimento di attività a partire da misurazioni di sensori ambientali basato su *CRF*.

Support Vector Machines

Una *Support Vector Machine* (*SVM*) rappresenta un metodo per la classificazione di dati sia lineari che non lineari. I dati del training set vengono proiettati tramite una funzione non lineare in uno spazio ad una dimensionalità maggiore rispetto a quella

di partenza. In questa nuova dimensione l'obbiettivo è quello di trovare il miglior iperpiano che riesca a separare i dati appartenenti alle diverse classi. Utilizzando una proiezione in una dimensionalità sufficientemente elevata, i dati appartenenti a due distinte classi saranno sempre separabili [4]. In [7] viene proposto un modello di riconoscimento di attività basato su *SVM*.

1.2.2 Tecniche simboliche

Le tecniche di tipo simbolico si basano sull'idea di sfruttare la conoscenza pregressa sul sistema che si intende modellare. Le attività e gli eventi di sensori sono rappresentate tramite opportuni formalismi logici ed il riconoscimento avviene tramite un motore di inferenza logica. L'utilizzo di una tecnica simbolica prevede:

- Rappresentazione delle attività del dominio tramite un formalismo logico.
- Aggregazione e trasformazione di dati di sensore in termini e formule logiche.
- Inferenza logica per estrarre le attività svolte partendo da un insieme di osservazioni.

Queste tecniche sfruttano la conoscenza a disposizione per creare opportuni modelli logici in grado di riconoscere le attività svolte. Il principale punto debole è l'impossibilità di rappresentare l'incertezza intrinseca nei dati raccolti tramite sensori [4]. Vengono ora descritte alcune delle principali tecniche simboliche presenti in letteratura.

Ontologie

Le ontologie sono una descrizione formale ed esplicita di un dominio di interesse. Tramite un'ontologia è possibile formalizzare la semantica di un dominio tramite assiomi. L'inferenza logica permette di derivare nuova conoscenza e rilevare inconsistenze. Le ontologie possono essere usate per modellare in maniera espressiva entità e

relazioni utili al riconoscimento di attività (eventi di sensori, elementi di contesto dell'ambiente, informazioni sugli attori, ...) in modo che possano essere comprensibili, condivisibili e riutilizzabili sia per le persone, sia per gli elaboratori [4]. Il processo di riconoscimento delle attività svolte da un attore consiste in:

- Mappare i dati provenienti dai sensori in istanze ontologiche.
- Rappresentare le attività da riconoscere in specializzazioni di una classe astratta (ad esempio *Activity*).
- Applicare tecniche di ragionamento ontologico a partire dal contesto rilevato tramite i sensori.

In [8] viene realizzato un approccio knowledge-based (basato sulla conoscenza) che considera l'informazione contestuale spazio-temporale per riconoscere le attività tramite logiche descrittive. In [9], invece, le attività vengono rappresentate e riconosciute utilizzando il linguaggio ontologico *OWL2*.

Event Calculus

L'Event Calculus è un formalismo espresso tramite la logica del primo ordine volto a rappresentare la relazione tra eventi e i relativi effetti: un particolare effetto è presente in un preciso istante temporale se e solo se è stato *attivato* da un evento e non è stato ancora *terminato* da un altro evento. Il modello temporale utilizzato è lineare, ed è basato su valori interi o reali. Questo formalismo logico si adatta bene ad esprimere vincoli temporali su eventi di basso livello (come le rilevazioni di sensore) che, se soddisfatti, portano al riconoscimento di attività [10].

1.2.3 Tecniche ibride

Le tecniche ibride consistono nella combinazione di tecniche statistiche e simboliche in modo costruire modelli di riconoscimento che sfruttino i punti di forza di entrambi gli approcci:

- La capacità di gestire l'incertezza intrinseca nei dati raccolti tramite sensori.
- L'uso di linguaggi espressivi che permettono di rappresentare la semantica formale di tipi delle attività.

Queste tecniche prevedono due approcci distinti. Il primo prevede l'integrazione in un unico linguaggio di modelli statistici e simbolici. Il secondo prevede un'architettura multi-livello dove i due tipi di ragionamento sono effettuati separatamente. Vengono ora descritte due tecniche ibride, una per ogni tipologia di approccio.

Approccio integrato

Un esempio di tecnica che integra ragionamento statistico e simbolico è quello delle *Markov Logic Network*. Una *Markov Logic Network* (*MLN*) è una base di conoscenza della logica del primo ordine dove ad ogni formula è abbinato un peso. Una *MLN* è un template per la costruzione di una *rete di Markov* (un modello grafico probabilistico non diretto). Viene quindi incorporato in un unico linguaggio il ragionamento statistico e quello logico: la conoscenza del dominio viene utilizzata per costruire in maniera flessibile opportune *reti di Markov* che riescono quindi a considerare l'incertezza [11]. In [12] viene usato questo formalismo per il riconoscimento automatico di attività umane, formalizzando le dipendenze tra eventi di sensore, le attività svolte e il loro contesto temporale. A differenza della maggior parte degli approcci che considera attività svolte in stretta sequenza, questo lavoro considera il problema di riconoscere attività simultanee e innestate. Tuttavia, viene fatta l'assunzione che non possa essere presente più di una istanza della stessa attività.

Approccio multilivello

Una tecnica che invece utilizza un approccio multilivello è quella proposta dal sistema *COSAR*. *COSAR* (*Combined Ontological/Statistical Activity Recognition*) è un sistema mobile per il riconoscimento di attività umane che combina il ragionamento ontologico e quello statistico, eseguendoli in passi successivi [13]. I dati trasmessi da sensori di ambiente e sensori indossabili vengono comunicati tramite una connessione wireless ad un dispositivo mobile dell'attore, il quale costruisce un vettore di feature utilizzato per predire le attività svolte. Vengono quindi utilizzate delle tecniche di ragionamento statistico per ricavare le possibili attività svolte (e i relativi livelli di confidenza) dai dati provenienti dai sensori. Successivamente viene effettuato un ragionamento ontologico per determinare l'insieme di attività che con maggiore probabilità corrispondono a quella attualmente eseguita dall'attore considerato, tenendo conto del contesto spazio-temporale.

1.3 Tecniche di riconoscimento di anomalie

Il riconoscimento di anomalie all'interno di attività umane si occupa di individuare, per un particolare attore monitorato, rilevanti variazioni del comportamento rispetto a quello atteso. In [14] vengono presentate diverse metodologie basate su tecniche di computer vision per riconoscere comportamenti anomali tramite l'utilizzo di videocamere. Questo approccio risulta però invasivo, in quanto comporta problemi di privacy se utilizzato all'interno di abitazioni. In [19] viene invece proposto un sistema di rilevamento di anomalie con lo scopo di effettuare una diagnosi di deterioramento cognitivo. Utilizzando tecnologie non invasive installate nelle abitazioni dei pazienti, vengono monitorate la velocità media di percorrenza di brevi tratti e il tempo di permanenza medio del paziente nelle varie stanze nell'ambiente domestico. Questo approccio ha lo svantaggio di analizzare eventi semplici, non considerando le attività

effettivamente svolte dai pazienti e la relativa qualità. Per determinare in maniera accurata la qualità di esecuzione di attività è infatti necessario comprendere nei dettagli le attività che vengono svolte. In questa sezione vengono presentate delle tecniche che sfruttano l'informazione prodotta da sensori installati negli ambienti domestici, in modo tale da realizzare modelli di riconoscimento poco invasivi che considerando eventi di natura complessa.

Tecniche di Machine Learning

In [7] viene predisposta una smart home per monitorare lo svolgimento di attività quotidiane da parte di attori sani e attori affetti da diversi livelli di deterioramento cognitivo. Un osservatore si occupa di annotare l'inizio e la fine delle attività che vengono svolte dai pazienti, assegnando un punteggio ad ogni attività in base alla qualità di esecuzione della stessa (utilizzando un criterio fissato). Viene quindi utilizzato un algoritmo di apprendimento supervisionato (*SVM*) in due livelli. Il primo livello si occupa di riconoscere le attività a partire dalle misurazioni prodotte dai sensori. In un secondo livello, un classificatore per ogni tipo di attività si occupa riconoscere la qualità di esecuzione della stessa inferendone il punteggio. La stessa tipologia di esperimenti è stata quindi ripetuta usano l'algoritmo di apprendimento non supervisionato *Principal Component Analysis*. Questo lavoro non ha però prodotto risultati soddisfacenti: in entrambi i casi è stata ottenuta una scarsa correlazione (in alcuni casi addirittura negativa) tra i punteggi ottenuti tramite riconoscimento e quelli effettivamente osservati. In [15] viene invece utilizzata una variante di *SVM* chiamata *One Class Support Vector Machine*. Questo classificatore viene allenato utilizzando solo dati di una determinata classe, ed è successivamente in grado di riconoscere se un particolare dato appartiene o meno alla classe specificata. Questo approccio è particolarmente utilizzato in ambito di rilevamento di anomalie, in quanto è spesso difficile ottenere una quantità sufficiente di dati contenenti anomalie. Vengono quindi raccolte ed etichettate attività prive di ano-

malie tramite misurazioni provenienti da sensori installati in un ambiente domestico, in modo da allenare e validare una *One Class Support Vector Machine*. Successivamente, il classificatore prodotto sarà in grado di individuare comportamenti anomali. Questo approccio presenta lo svantaggio di riconoscere solo la presenza e l'assenza di anomalie, senza effettuare una distinzione tra le diverse tipologie di comportamento anomalo. La qualità di esecuzione delle attività non viene valutata.

Temporal Data Mining

Il lavoro proposto in [17] si basa sul fatto che il riconoscimento di anomalie è più accurato quando è basato su comportamenti che sono più frequenti e predicibili. Vengono quindi usate tecniche di temporal data mining: viene monitorato tramite sensori ambientali il comportamento di un attore nella sua abitazione, analizzando le interazioni temporali più frequenti tra gli eventi che vengono rilevati. Ad esempio, può essere frequente che l'evento *accendere la televisione* sia frequentemente preceduto da *sedersi sul divano*. Se questo vincolo viene violato, viene rilevata un'anomalia. Vengono a questo scopo utilizzate alcune delle relazioni temporale proposte in [18]:

- $before(X,Y)$: l'evento X avviene prima dell'evento Y,
- $contains(X,Y)$: l'evento Y inizia dopo l'inizio dell'evento X, ma prima del suo termine.
- $overlaps(X,Y)$: l'evento Y inizia dopo l'inizio di X (mentre è ancora in corso) e termina dopo il suo termine.
- $meets(X,Y)$: il termine dell'evento X coincide con l'inizio dell'evento Y.
- $starts(X,Y)$: l'inizio dei due eventi coincide, ma non la fine
- $finishes(X,Y)$: la fine dei due eventi coincide, ma non l'inizio.

- $equals(X,Y)$: gli eventi X e Y iniziano e terminano ai medesimi istanti di tempo.

Il primo passo è quello di ricavare da un dataset di misurazioni di sensori le relazioni temporali più frequenti tramite l'algoritmo *Apriori*. Il risultato di questa fase è una lista di eventi frequentemente ricorrenti in base alle relazioni temporali sopra specificate. Il rilevamento di anomalie viene effettuato calcolando la probabilità di occorrenza (o di non occorrenza) di un evento: la probabilità che un evento Z occorra è basata sull'occorrenza di altri eventi che presentano una relazione temporale con Z. Quindi, se la probabilità di un evento appena osservato tende a 0, viene segnalata l'anomalia. La problematica con questo approccio è che non è in grado di determinare quali eventi osservati appartengono alla stessa attività, e quindi allo stesso intervallo temporale.

Capitolo 2

Collocazione della tesi nell'ambito del progetto *SECURE*

Questo lavoro di tesi ha come scopo la progettazione, realizzazione e valutazione di un sistema intelligente basato su sensori per monitorare e valutare il comportamento di pazienti affetti da disturbi cognitivi lievi nell'esecuzione di attività quotidiane all'interno delle proprie abitazioni. L'obiettivo è quello di riconoscere:

- Attività quotidiane svolte.
- Comportamenti anomali riscontrati durante l'esecuzione di queste attività.

La metodologia adottata consiste nell'installazione di sensori nell'ambiente domestico (sportelli, fornello, sedie, ...) per ottenere delle misurazioni significative riguardanti l'interazione del paziente con i vari oggetti domestici. Ad esempio, tramite un sensore di contatto installato sulla porta del frigorifero è possibile monitorare l'apertura e la chiusura dello stesso. Queste rilevazioni vengono successivamente analizzate per riconoscere attività ed anomalie.

La Sezione 2.1 presenta il progetto SECURE e il lavoro svolto nell'ambito di questo

progetto. Nella Sezione 2.2 viene presentata l'architettura generale del sistema proposto. Nella Sezione 2.3 viene descritta l'acquisizione di dataset per testare e validare le tecniche di riconoscimento descritte nel capitolo successivo. Infine, nella Sezione 2.4 vengono spiegate le tecnologie sensoristiche utilizzate, evidenziando le problematiche incontrate.

2.1 Collocazione e obiettivi del lavoro di tesi

Questo lavoro di tesi si inserisce nell'ambito del progetto *SECURE*[1]: "Sistema intelligente per diagnosi precoci e follow-up domiciliare", finanziato da *MIUR* e *Regione Lombardia* e realizzato insieme ai partner *3Caravelle*, *Health Telematic Network* e l'*Istituto Centro San Giovanni di Dio Fatebenefratelli* di Brescia. La durata del progetto è prevista da Giugno 2012 a Dicembre 2014. Questo progetto si occupa di effettuare telesorveglianza domiciliare di pazienti affetti da deterioramento cognitivo lieve, monitorando la loro situazione clinica e la situazione comportamentale. Il deterioramento cognitivo lieve è una condizione considerata come uno stato di transizione tra il normale declino cognitivo nell'età avanzata e la demenza, e comporta marcate modificazioni del comportamento ed esecuzione di comportamenti a rischio [20]. I pazienti affetti da questa problematica possono inoltre presentare situazioni cliniche di svariato tipo tra le quali: precario compenso di circolo, rischio di variazioni pressorie, precario bilancio idro-salino, alterazioni della deambulazione, alterazione dell'apporto alimentare, nutrizione non bilanciata. L'obiettivo del progetto è quello di realizzare un sistema basato su sensori che sia in grado di fornire informazioni oggettive circa lo stato cognitivo e le condizioni funzionali di un paziente, tale da rendere possibile una diagnosi accurata di deterioramento cognitivo. Il sistema di sensori deve quindi essere in grado di monitorare:

[1] http://secure.ewlab.di.unimi.it/

- La situazione clinica.
- La situazione comportamentale.

Nel primo caso, la soluzione proposta è quella di utilizzare sensori indossabili dal paziente per la rilevazione di parametri vitali. Nel secondo caso, invece, la soluzione proposta è quella di utilizzare sensori di ambiente per monitorare il livello di menomazione delle abilità in un contesto di vita quotidiana. Questo lavoro di tesi contribuisce all progetto *SECURE* proponendo e sperimentando una tecnica per il monitoraggio delle attività quotidiane e per il riconoscimento di anomalie comportamentali. In base alle indicazioni ricevute dai partner medici del progetto, le attività considerate in questo lavoro sono:

- Assumere le medicine prescritte dai medici
- Preparare i pasti
- Consumare i pasti

Le anomalie che occorre rilevare durante l'esecuzione di queste attività sono classificate come:

- Omissioni Critiche: il paziente si dimentica di compiere un'azione importante (per esempio dimentica di assumere un medicinale prescritto).
- Omissioni Non Critiche: il paziente si dimentica di compiere azioni meno rilevanti (per esempio dimentica di chiudere l'armadietto delle medicine).
- Sostituzioni: il paziente sostituisce un oggetto con un altro (per esempio assume un medicinale non prescritto al posto di quello prescritto)
- Ripetizioni: il paziente si dimentica di aver compiuto un'azione e la ripete (per esempio assume la stessa medicina due volte nello stesso giorno mentre è prescritto che la stessa venga assunta solo una volta al giorno)

2.2 Architettura

L'architettura generale del sistema è stata definita precedentemente al lavoro di tesi dai partner tecnologici del progetto. Le rilevazioni dei sensori di ambiente vengono raccolte, preprocessate ed inviate ad un gateway che le memorizza in un database interno. Queste informazioni vengono quindi inviate ad un dispositivo mobile che effettua periodicamente il riconoscimento di attività ed anomalie utilizzando le rilevazioni accumulate. Il risultato viene infine inviato ad un sistema di telemedicina per essere analizzato dai medici.

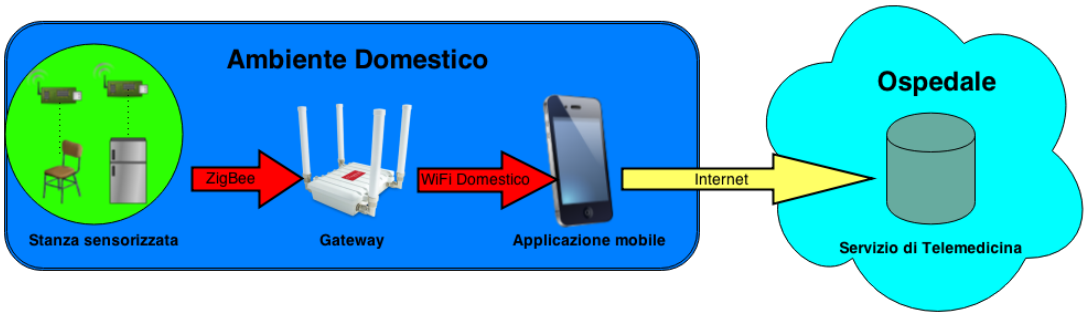

Figura 2.1: Architettura generale del sistema

2.2.1 Nodi sensorizzati

Un *mote* è un nodo facente parte di una rete wireless in grado di raccogliere e preprocessare informazioni dai sensori ad esso collegati e di comunicare con gli altri nodi connessi in rete. Il *mote* scelto per questo progetto è *Waspmote* della *Libelium*, che si focalizza nell'implementazione di modalità di basso consumo:

- *Deep Sleep* mode: in questa modalità vengono alimentati solamente i sensori, che generano degli interrupt per svegliare il micro-controllore principale quando

il valore letto è superiore o inferiore ad una certa soglia fissata. Questo permette un buon risparmio energetico.

- *Hibernate* mode: in questa modalità l'intero sistema è spento per assicurare il minimo consumo. In questo caso il micro-controllore viene svegliato da un segnale proveniente dal clock interno.

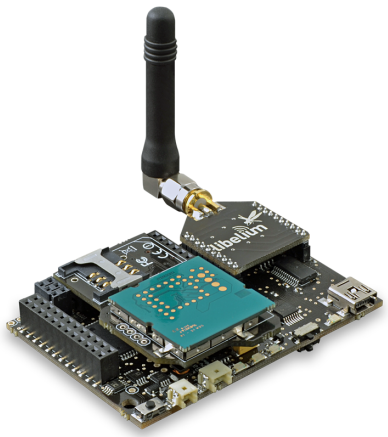

Figura 2.2: *Waspmote*

Waspmote è basato sull'idea di integrare solo le componenti strettamente necessarie in ogni dispositivo, in modo tale da ottimizzare i costi: i moduli possono essere facilmente aggiunti o rimossi in base alle esigenze.
In questo progetto viene installata nell'ambiente domestico una rete di sensori collegati a diversi *Waspmote* in *Deep Sleep* mode.

2.2.2 Gateway

Il gateway utilizzato in questo lavoro di tesi per raccogliere le misurazioni di sensore dai vari *mote* è *Meshlium*. *Meshlium* è un router *Linux* che funziona da gateway per una rete di sensori *Waspmote*, i quali comunicano tramite il protocollo *ZigBee* le rilevazioni

acquisite dai sensori. *Meshlium* ha diverse modalità per la gestione dei dati ricevuti dai vari mote:

- Salvare le informazioni in un database locale *MySQL*.
- Salvare le informazioni in un database esterno *MySQL*.
- Inviare le informazioni su Internet utilizzando diversi tipi di connessione: *Ethernet*, *WiFi*, *3G*, *GPRS*.

In questo progetto le rilevazioni ricevute dai vari *Waspmote* installati nell'ambiente domestico vengono memorizzate nel database interno. Queste informazioni vengono trasmesse periodicamente (ad esempio ogni 30 secondi) ad un dispositivo mobile che si occupa dell'elaborazione dei dati.

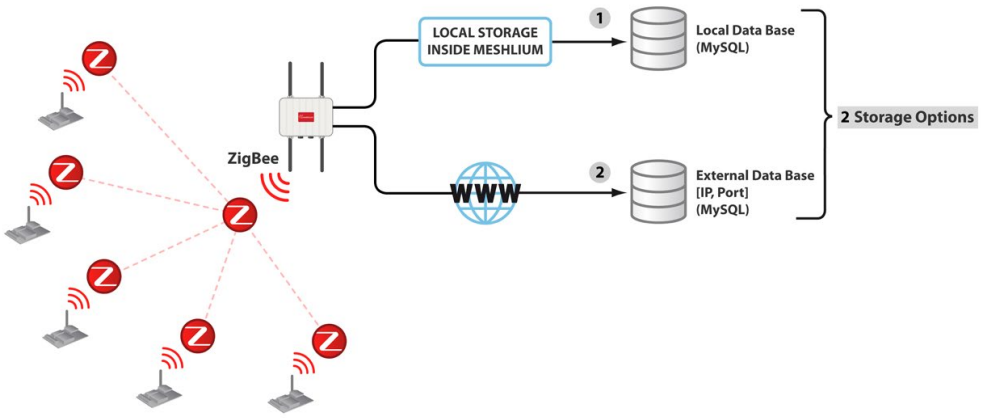

Figura 2.3: Una rete di sensori *Waspmote* con gateway *Meshlium*

2.2.3 Applicazione mobile

Questo lavoro di tesi si è focalizzato sulla realizzazione della parte di analisi delle rilevazioni dei sensori, sviluppando un'applicazione *Android* installata su un dispositivo

mobile presente nella casa del paziente. Le acquisizioni vengono inviate in tempo reale da *Meshlium*, prelevandole dal database interno. L'applicazione mobile accumula quindi queste misurazioni e periodicamente (per esempio, una volta al giorno) esegue il riconoscimento di attività e di anomalie. L'analisi delle rilevazioni è strutturata in tre diversi moduli:

- Riconoscimento di inizio e fine di attività
- Raffinamento delle attività riconosciute
- Riconoscimento di anomalie

Riconoscimento di inizio e fine di attività

Il riconoscitore di inizio e fine di attività prende come input le misurazioni dei sensori ambientali di un determinato intervallo temporale (ad esempio un'intera giornata) con l'obiettivo di inferire gli istanti di tempo di inizio e di fine delle attività quotidiane svolte (ad esempio, un'attività "*assumere le medicine*" è cominciata alle 11:00 ed è terminata alle 11:25). Il riconoscimento di attività avviene tramite un approccio di apprendimento supervisionato, utilizzando come framework *Markov Logic Network*[11].

Raffinamento delle attività riconosciute

Il riconoscimento di inizio e fine di attività è ovviamente soggetto a generare falsi positivi e falsi negativi. Un falso positivo è un inizio o una fine di attività rilevato ma non avvenuto, mentre un falso negativo è un inizio o una fine di attività non inferito ma in realtà avvenuto. Lo scopo del modulo di estrazione di attività è quello di raffinare le attività riconosciute nel modulo precedente prestando attenzione al caso di falsi positivi o falsi negativi, correggendo (se è possibile) i problemi derivanti da predizioni sbagliate.

Riconoscimento di anomalie

Una volta riconosciute le attività, è necessario effettuare il riconoscimento delle eventuali anomalie presenti in esse. Per questo scopo viene utilizzato un sistema di ragionamento basato sulla logica del primo ordine chiamato *TuProlog*, una versione del linguaggio di programmazione *Prolog* per dispositivi mobili [30]. Tramite delle regole vengono quindi inferite le anomalie presenti nelle attività, effettuando ragionamenti di tipo temporale.

Dopo aver riconosciuto attività ed anomalie, queste informazioni vengono comunicate al sistema di telemedicina tramite il quale i medici potranno analizzarle.

2.3 Acquisizione di un dataset di attività e anomalie

Per allenare e validare i sistemi di apprendimento automatico utilizzati per il riconoscimento di attività, una parte di questo lavoro di tesi è stata dedicata all'acquisizione di dati opportunamente annotati di svolgimento di attività quotidiane ed anomalie in una stanza dotata di sensori. In assenza di pazienti ai quali far svolgere le attività, è stato necessario simulare il loro comportamento nell'ambiente per ottenere dei dati.

2.3.1 Annotazione dei dati

Per validare i modelli di riconoscimento, è necessario annotare i dati durante l'acquisizione indicando le attività e le relative anomalie. Per facilitare e velocizzare questo processo, è stato sviluppato un semplice programma a bordo di *Meshlium* che riceve interattivamente comandi remoti per l'inserimento di annotazioni durante l'esecuzione delle attività. Le annotazioni vengono quindi inserite nello stesso database interno a

Meshlium che contiene le misurazioni acquisite dai sensori. Vengono usati tre diversi tipi di annotazione:

- Annotazione di inizio attività
- Annotazione di fine attività
- Annotazione di avvenuta anomalia

L'annotazione di inizio attività viene inserita nel sistema appena prima di cominciare l'attività, mentre l'annotazione di fine attività viene inserita al termine dell'esecuzione di essa. L'annotazione di anomalia, invece, viene inserita nel sistema non appena l'anomalia si è verificata.

2.3.2 Acquisizione di un dataset di attività guidate

Il primo dataset è stato acquisito pianificando nei dettagli le attività quotidiane prima di essere svolte nell'ambiente da parte di attori. Sono state simulate attività (e relative anomalie in esse) svolte da tre differenti pazienti: uno sano, uno affetto da deterioramento cognitivo lieve e uno affetto da deterioramento cognitivo grave. Per ogni paziente sono stati simulati sette giorni di attività, per un totale di 21 giorni di misurazioni. Ovviamente non essendo pratico rispettare realmente i tempi di esecuzione delle attività che sono presenti nella vita di tutti i giorni, è stato sviluppato un sistema per modificare la data e l'ora di *Meshlium* a piacimento, in maniera tale da velocizzare sensibilmente il processo di acquisizione. Questi dati sono stati utilizzati per la modellazione e la validazione dei modelli di riconoscimento.

2.3.3 Acquisizione di un dataset di attività spontanee

Il secondo dataset acquisito prevede attività svolte senza seguire nessuno schema, seguendo un approccio quindi più realistico: quattro individui hanno eseguito una volta

ognuno tutte le attività previste. In questo dataset non sussistono anomalie, essendo i partecipanti all'esperimento soggetti sani. Questo dataset è utile per capire quali differenze vengono riscontrate rispetto al dataset precedente. Inoltre, questo dataset è stato usato per la validazione del modello di riconoscimento di attività costruito con i dati acquisiti artificialmente.

2.4 Considerazioni sui sensori utilizzati

Un sensore è uno strumento che misura una quantità fisica e che la converte in un segnale. In questa sezione vengono presentati i sensori di ambiente utilizzati per realizzare il prototipo, le problematiche riscontrate ed eventuali soluzioni che potrebbero essere utilizzate in futuro. Questi sensori sono installati su oggetti domestici, in modo tale da rilevare l'interazione del paziente con l'ambiente durante l'esecuzione di attività quotidiane.

2.4.1 Sensore Hall Effect *PLA41201*

Il sensore Hall Effect *PLA41201* (mostrato in Figura 2.4) è un trasduttore che varia il suo voltaggio in uscita in base ad un campo magnetico: l'interruttore del sensore è chiuso in presenza del campo magnetico e aperto in sua assenza. Utilizzando un magnete *P6250000* (Figura 2.5), è possibile rilevare due stati tramite il voltaggio in uscita dal sensore:

- Quando il sensore e il magnete sono vicini
- Quando il sensore e il magnete sono lontani

In questo modo è possibile monitorare il meccanismo di apertura e chiusura di sportelli (come mostrato nella Figura 2.6), analizzando l'interazione del paziente con svariati

mobili: frigorifero, armadietto per le medicine ed altri armadietti posizionati in cucina.

Figura 2.4: Sensore Hall Effect *PLA41201*

Figura 2.5: Magnete *P6250000*

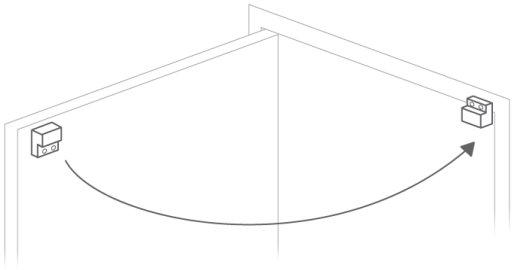

Figura 2.6: Funzionamento

2.4.2 RFID

Un sistema *RFID* (*Radio-Frequency IDentification*) permette di identificare degli oggetti senza il bisogno di contatto fisico. Un sistema *RFID* è composto da due elementi:

- Il modulo *RFID*.
- I tag *RFID*.

I tag possono essere visti come dispositivi di memorizzazione remoti che vengono scritti e/o letti quando sono avvicinati al modulo, che si occupa quindi di interrogare e di mandare istruzioni ai tag. In questo progetto, dei tag *RFID* vengono applicati su confezioni di medicine e alimenti. Questi tag contengono in memoria interna un identificativo dell'oggetto al quale sono correlati. Nel prototipo realizzato, ogni qualvolta

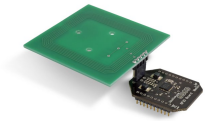

Figura 2.7: Modulo *RFID*

Figura 2.8: Tag *RFID*

che il paziente prende o rimette al suo posto un particolare oggetto, il tag sulla confezione deve essere passato sul modulo, il quale esamina il contenuto della memoria interna del tag per identificare l'oggetto. Questo approccio è utile per comprendere il comportamento del paziente riguardo l'assunzione di alimenti e medicine. Ad esempio, è importante controllare che il paziente assuma le medicine correttamente in base alla prescrizione del medico.

Problematiche

1. Il paziente deve ricordarsi di passare il tag ogni qualvolta prende o rimette a posto un oggetto. Per pazienti che presentano deterioramenti cognitivi questa assunzione non è verosimile. In una versione successiva del prototipo, si ipotizza l'utilizzo di sensori con un raggio di lettura maggiore, in modo da evitare il passaggio esplicito del tag sul lettore da parte del paziente.

2. L'informazione prodotta dal modulo *RFID* non distingue l'azione di prendere e l'azione di rimettere a posto un oggetto.

Assunzioni

Questo tipo di sensore viene usato temporaneamente in attesa di nuove tecnologie che producano informazioni più accurate (ad esempio per rilevare l'effettiva presenza

o assenza di un particolare medicinale nell'armadietto). Si fanno quindi le seguenti assunzioni:

1. Il paziente si ricorda sempre di passare il tag ogni volta che prende o restituisce un oggetto fornito di tag *RFID* (medicinale o articolo alimentare).

2. Per ogni coppia di misurazioni consecutive *RFID* su uno stesso oggetto, la prima indica che l'oggetto è stato preso e la seconda che è stato restituito.

2.4.3 Sensore di temperatura *MCP9700A*

Il sensore *MCP9700A* converte la temperatura esterna in un voltaggio analogico proporzionale ad essa. In questo modo è possibile analizzare la temperatura dell'ambiente nel quale il sensore è posizionato.

Figura 2.9: Sensore di temperatura *MCP9700A*

Nella nostra applicazione, il sensore viene posto sopra il fornello per monitorarne l'utilizzo. Quando viene rilevata una temperatura superiore ad una soglia fissata (da calibrare in base all'ambiente), vuol dire che il fornello è acceso. Quando invece la temperatura diminuisce tornando sotto la soglia vuol dire che il fornello è stato spento.

Problematiche

1. Questo tipo di sensore è poco adatto in caso di fornelli elettrici in quanto questi determinano una minor dispersione di calore rispetto al fuoco.

2. Possono frequentemente occorrere situazioni nelle quali la temperatura rilevata dal sensore continui a fluttuare attorno al valore di soglia.

Viste le problematiche, per ottenere delle misurazioni più accurate è consigliato utilizzare dei sensori di rilevazione di corrente in presenza di fornelli elettrici invece dei sensori di temperatura appena descritti.

2.4.4 Sensore di pressione *Flexiforce PS-02*

Il sensore *Flexiforce PS-02* è un sensore di tipo resistivo: la resistenza in uscita cambia in base alla pressione esercitata sull'area circolare del sensore (vedi Figura 2.10). Ad ogni cambio significativo di resistenza (determinato da una soglia da calibrare), il sensore notifica se la pressione è aumentata o diminuita.

Figura 2.10: Sensore di pressione

In questo esperimento il sensore viene installato sulle sedie, in modo tale da capire quando il paziente si alza o si siede.

Problematiche

Essendo l'area circolare molto piccola, è necessario realizzare un sistema che riesca a concentrare il peso corporeo in quella zona ristretta.

2.4.5 PIR

Il sensore *PIR* (*Passive InfraRed*) è un sensore che misura i raggi infrarossi irradiati da oggetti nel suo campo di osservazione. Nel nostro caso, questo sensore viene utilizzato

Figura 2.11: Sensore *PIR*

per rilevare movimento nelle vicinanze del tavolo dove il paziente è solito mangiare. In questo modo è possibile capire (combinando l'informazione con il sensore di pressione sulle sedie) quando il paziente si trova in presenza del tavolo e per quanto tempo.

Problematiche

Questo sensore invia una misurazione per ogni movimento rilevato. Nel caso in cui il paziente è presente per un lungo periodo al tavolo, un gran numero di rilevazioni superflue vengono acquisite. Queste misurazioni rumorose possono compromettere la buona riuscita degli algoritmi di riconoscimento. Per risolvere questo problema è possibile adottare diverse strategie:

1. Eliminare l'informazione ridondante a priori, impostando un periodo di attesa per il sensore dopo ogni misurazione in modo tale da ridurre il numero di acquisizioni.

2. Eliminare l'informazione ridondante a posteriori, scartando durante l'analisi dei dati le eccessive rilevazioni del sensore.

2.4.6 Ambiente di acquisizione

Una stanza sensorizzata è stata quindi predisposta per l'acquisizione dei dati in un ambiente fornito dal partner *3Caravelle*. In tabella 2.1 vengono mostrati gli oggetti domestici monitorati, specificando per ogni oggetto quale sensore è stato utilizzato. Come già spiegato, le confezioni di medicine e di alimenti sono dotate di un tag *RFID*.

Oggetto monitorato	**Sensore**
Medicine	*RFID*
Alimenti	*RFID*
Armadietto medicine	*Hall Effect*
Frigorifero	*Hall Effect*
Armadietto alimenti	*Hall Effect*
Armadietto con materiale per cucinare	*Hall Effect*
Cassetto delle posate	*Hall Effect*
Fornello	Sensore di tempreatura
Tavolo adibito al pranzo	*PIR*
Sedia	Sensore di pressione

Tabella 2.1: Oggetti monitorati

Ogni volta che viene preso o restituito un oggetto, il tag deve essere passato sul modulo. Nel nostro ambiente sono presenti due moduli *RFID*: uno per le medicine e uno per gli alimenti (entrambi adiacenti ai relativi armadietti). I vari armadietti/cassetti vengono monitorati con dei sensori *Hall Effect*, in maniera tale da rilevare apertura e chiusura. Il fornello usato per cucinare viene monitorato con un sensore di temperatura che, in base ad una soglia, rileva quando il fornello è acceso o spento. Infine, la presenza intorno alla zona adibita al pranzo viene monitorata tramite un sensore di presenza che rileva i movimenti attorno al tavolo da pranzo e con un sensore di pressione installato sulla sedia annessa al tavolo.

Capitolo 3

Riconoscimento di attività e di anomalie

Nel capitolo precedente sono state analizzate le tecnologie utilizzate per l'acquisizione di dati in un ambiente domestico dotato di sensori, in maniera tale da monitorare le attività quotidiane svolte da pazienti affetti da deterioramento cognitivo lieve. In questo capitolo viene discusso l'aspetto principale di questo lavoro di tesi: la realizzazione di un sistema intelligente in grado di interpretare le misurazioni dei sensori acquisite nell'ambiente domestico e di riconoscere le attività quotidiane svolte dai pazienti evidenziando le eventuali anomalie. Verranno quindi introdotti gli strumenti utilizzati a tale scopo, spiegando nei dettagli la modellazione e la realizzazione del sistema. Il riconoscimento delle attività quotidiane viene realizzato tramite la logica probabilistica *Markov Logic Network*. Le attività riconosciute vengono successivamente analizzate da un sistema a regole della logica del primo ordine per inferire eventuali anomalie.

Nella Sezione 3.1 viene spiegato il framework statistico *Markov Logic Network* e i concetti teorici necessari per comprenderlo. La Sezione 3.2 spiega nei dettagli il metodo realizzato per il riconoscimento delle attività quotidiane. Infine, la Sezione 3.3

introduce il sistema a regole della logica del primo ordine utilizzato per individuare le anomalie all'interno delle attività.

3.1 La logica probabilistica *Markov Logic Network*

In questa sezione viene analizzata la logica probabilistica *Markov Logic Network* [11], utilizzata per il riconoscimento delle attività quotidiane a partire dalle misurazioni dei sensori. Questo strumento permette di unire ragionamento logico e ragionamento statistico e per comprenderlo è necessario introdurre i concetti di *rete di Markov* (il modello grafico probabilistico su cui si basa) e di logica del primo ordine (la sua sintassi). Questa tecnica ibrida è stata scelta per tentare di ottenere risultati più promettenti rispetto al lavoro svolto in [7] che utilizza un approccio puramente statistico.

3.1.1 Modelli grafici probabilistici

I modelli grafici probabilistici sono uno strumento per rappresentare complesse distribuzioni di probabilità tramite dei grafi. I nodi del grafo rappresentano le variabili casuali del dominio, mentre gli archi rappresentano la dipendenza probabilistica tra di esse. In caso di grafo diretto, parliamo di modelli grafici probabilistici diretti (ad esempio le reti Bayesane). In caso non diretto, invece, parliamo di modelli grafici probabilistici non diretti (come le reti di Markov). La rappresentazione a grafo ci permette di sfruttare la proprietà per la quale in molte applicazioni reali le variabili casuali tendono a dipendere direttamente solo da poche altre. I modelli grafici probabilistici permettono di effettuare *inferenza*, ovvero di calcolare la probabilità a posteriori di alcune variabili osservando il valore assunto da altre basandosi direttamente sul grafo, invece di gestire la distribuzione congiunta esplicitamente. La costruzione di un modello grafico probabilistico avviene con un approccio di tipo *data-driven* (guidato dai dati) tramite un processo di *apprendimento*: un esperto del dominio esplicita gli attri-

buti e le dipendenze che il modello dovrebbe contenere, mentre i parametri vengono dedotti automaticamente adattando il modello ai dati che rappresentano esperienza del passato (ad esempio potremmo voler modellare un modello probabilistico riguardante una qualche malattia partendo dalle informazioni acquisite in passato da pazienti) [21].

3.1.2 *Markov Random Field*

Un *Markov Random Field* (detto anche rete di *Markov*) è un modello grafico probabilistico non diretto volto a rappresentare la probabilità congiunta di un insieme di variabili casuali $X = (X_1, X_2, \ldots, X_n)$. Una rete di *Markov* è rappresentata da un grafo non orientato. Per ogni *clique* k del grafo, esiste una funzione potenziale ϕ_k non negativa su valori reali in funzione dello stato della *clique* corrispondente. La seguente formula rappresenta la distribuzione congiunta in una rete di *Markov*:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$$

Il termine Z rappresenta un fattore di normalizzazione $\sum_{x \in X} \prod_k \phi_k(x_{\{k\}})$, mentre $x_{\{k\}}$ rappresenta lo stato delle variabili della k-esima *clique*.

Le reti di *Markov* possono essere in alternativa rappresentate con modelli *log-lineari*, dove ogni funzione potenziale viene sostituita da una somma pesata esponenziata di feature:

$$P(X = x) = \frac{1}{Z} \exp(\sum_j w_j f_j(x))$$

Una feature f_j è una funzione su valori reali in funzione dello stato. Nel nostro caso avremo a che fare con feature binarie, ovvero dove $f_j(x) \in \{0, 1\}$ (serviranno a rappresentare il valore di verità di formule della logica del primo ordine). Nella traduzione diretta da un modello all'altro, esiste una feature per ogni stato $x_{\{k\}}$ del grafo (uno per

ogni *clique k*). In questo modo, però, la rappresentazione diventa esponenziale rispetto alla dimensione delle cricche. Si cerca quindi di specificare un minor numero di feature per una rappresentazione più compatta (ad esempio utilizzando funzioni logiche sullo stato di una *clique*). L'inferenza nelle reti di *Markov* è #*P*-completa, rientrando quindi nella classe di problemi di conteggio di soluzioni di problemi difficili. Per un calcolo efficiente, quindi, si ricorre a tecniche di approssimazione utilizzando il metodo *Markov Chain Monte Carlo*, dove le probabilità marginali vengono calcolate campionando alcune variabili casuali ed effettuando il conteggio su questi campioni. Il campionamento può avvenire con diversi criteri, ad esempio tramite il Campionamento di Gibbs. Se invece di calcolare le probabilità marginali si vuole inferire il mondo più probabile, è possibile effettuare l'inferenza tramite le tecniche di massima verosimiglianza (likelihood) o di stima a massima posteriori (*MAP*) tramite metodi standard di ottimizzazione numerica [11].

3.1.3 Logica del primo ordine

Una base di conoscenza del primo ordine (*KB*) è un insieme di formule espresse nella logica del primo ordine [22]. Queste formule sono costruite usando quattro tipi di simboli: *costanti*, *variabili*, *funzioni* e *predicati*. Le costanti rappresentano gli oggetti nel dominio di interesse (ad esempio se vogliamo rappresentare delle persone, delle costanti possono essere: Carlo, Andrea, ...). Le variabili rappresentano individui indeterminati e possono assumere valori sugli oggetti del dominio. Le funzioni rappresentano una mappatura da tuple di oggetti del dominio a singoli oggetti del dominio (ad esempio una funzione potrebbe essere *madreDi*, che mappa un individuo con la relativa madre). Infine, i simboli di predicato rappresentano delle relazioni tra gli oggetti del dominio (ad esempio il predicato *amici* può essere usato per definire che due individui sono amici) oppure attributi di oggetti del dominio (ad esempio, il

predicato *fumatore* può essere utilizzato per indicare che un individuo è un fumatore). Le variabili e le costanti possono essere tipate, in modo tale che le variabili possono assumere solo valori di costanti di un tipo specifico. Un *termine* è un'espressione che rappresenta un oggetto del dominio, ed è definito in questo modo

- Una costante è un termine
- Una variabile è un termine
- Una funzione applicata su una tupla di termini è un termine

Un termine *ground* è un termine che non contiene variabili. Una formula atomica è un predicato applicato ad una tupla di termini.

Definizione 3.1.1. Le formule nella logica del primo ordine vengono definite ricorsivamente basandosi sul concetto di formula atomica:

- Una formula atomica è una formula.
- Se F_1 e F_2 sono formule, tali sono: $\neg F_1$ (vera se F_1 è falsa), $F_1 \wedge F_2$ (vera se sono entrambe vere), $F_1 \vee F_2$ (vera se almeno una delle due formule è vera) e $F_1 \Rightarrow F_2$ (falsa nell'unico caso in cui la prima è vera e la seconda è falsa).
- Se F è una formula, tali sono: $\forall x\ F$ (quantificatore universale, vero solo se F è vera per ogni sostituzione della variabile x con una costante) e $\exists x\ F$ (quantificatore esistenziale, vero se esiste almeno una sostituzione di x con una costante che rende la formula vera).

Un predicato *ground* è una formula atomica dove ogni termine è *ground*.
L'inferenza nella logica del primo ordine è basata sul determinare se una formula F è conseguenza logica di una base di conoscenza KB ($KB \models F$), ovvero se F è vera in tutti i mondi in cui KB è vera. Questo viene effettuato per refutazione, dimostrando

che $KB \bigcup \neg F$ è una contraddizione (non sarà mai vera). Questo problema è di natura semi-decidibile: l'algoritmo di decisione può non terminare in alcuni casi.

Esempio

Vediamo ora un classico esempio di base di conoscenza del primo ordine:

- $\forall x \, Uomo(x) \Rightarrow Mortale(x)$
- $Uomo(Socrate)$

I predicati utilizzati sono $Uomo$ e $Mortale$, ed è presente una costante dal nome $Socrate$. La prima regola esprime il concetto che ogni uomo è mortale, mentre la seconda esprime il fatto che Socrate è un uomo. Per effettuare dell'inferenza, possiamo provare a determinare se la formula $Mortale(Socrate)$ è conseguenza logica della base di conoscenza. Applicando la refutazione aggiungendo alla base di conoscenza la formula $\neg Mortale(Socrate)$ si ottiene una contraddizione: Socrate è un uomo ed ogni uomo è mortale, Socrate deve quindi essere mortale. Quindi, possiamo dedurre che $Mortale(Socrate)$ è conseguenza logica della base di conoscenza.

3.1.4 *Markov Logic Network*

Una base di conoscenza del primo ordine può essere vista come un insieme di regole rigide sull'insieme dei possibili mondi: se un mondo viola anche una sola formula, diventa automaticamente impossibile. L'idea delle *Markov Logic Network* è quella di rilassare questi vincoli: quando un mondo viola una formula, questo diventa meno probabile ma non impossibile. Quindi, meno formule vengono violate, più il mondo diventa probabile. Ogni formula ha associato un peso che indica quanto sia forte quel vincolo. La sintassi di una *MLN* è la sintassi standard della logica del primo ordine, dove le variabili libere (ovvero quelle prive di quantificatori $\forall$ o $\exists$) sono implicitamente trattate come se fossero quantificate universalmente. Diamo ora formalmente

una definizione delle *Markov Logic Network* (che da ora verranno chiamate in maniera compatta *MLN*).

Definizione 3.1.2. Una *MLN L* è un insieme di coppie (F_i, w_i) dove F_i è una formula nella logica del primo ordine e w_i è un numero reale. Dato un insieme di costanti $C = \{c_1, c_2, \ldots, c_{|C|}\}$, si ottiene una rete di *Markov* $M_{L,C}$ tale che:

1. $M_{L,C}$ contiene un nodo binario per ogni possibile grounding di ogni predicato che appare in *L*. Il valore del nodo è 1 se l'atomo ground è vero, 0 altrimenti.

2. $M_{L,C}$ contiene una feature per ogni possibile grounding di ogni formula F_i in *L*. Il valore della feature è 1 se la formula ground è vera, 0 altrimenti. Il peso relativo ad una feature f_i è il valore w_i associato alla formula F_i.

Dalla definizione segue che un arco tra due nodi di $M_{L,C}$ è presente se e solo se i corrispondenti atomi ground appaiono insieme in almeno una delle formule in *L*. Una *MLN* può quindi essere vista come un template per la costruzione di una rete di *Markov* $M_{L,C}$, detta anche rete di *Markov* ground. Ogni differente stato di $M_{L,C}$ rappresenta un mondo possibile: il valore di verità di ogni atomo ground. Sono necessarie però delle assunzioni che garantiscano che l'insieme dei mondi possibili sia finito e che $M_{L,C}$ rappresenti una distribuzione di probabilità unica e ben definita su quei mondi:

- **Nomi unici**. Costanti differenti si riferiscono ad oggetti differenti.

- **Chiusura di dominio**. Gli unici oggetti nel dominio sono quelli rappresentabili con simboli di costanti e simboli di funzione in (L, C).

- **Funzioni note**. Per ogni funzione in *L*, il valore di quella funzione applicata ad ogni possibile tupla di argomenti è noto, ed è un elemento di *C*.

La seguente equazione rappresenta la distribuzione di probabilità sui possibili mondi x di una rete di *Markov* ground $M_{L,C}$:

$$P(X = x) = \frac{1}{Z} \exp(\sum_i w_i n_i(x)) = \frac{1}{Z} \prod_i \phi_i(x_{\{i\}})^{n_i(x)}$$

dove $n_i(x)$ è il numero di grounding veri di F_i in x, $x_{\{i\}}$ è lo stato (valore di verità) degli atomi che appaiono in F_i ed infine la funzione potenziale $\phi_i(x_{\{i\}})$ è equivalente a e^{w_i}.

Apprendimento generativo

I pesi di una *MLN* vengono appresi utilizzando un database di osservazioni (detto anche *training set*) che rispettino l'assunzione di mondo chiuso: se un predicato ground non è presente tra le osservazioni, allora si assume che sia falso. Metodi standard di apprendimento possono essere utilizzati, ad esempio applicando la derivata della log-likelihood:

$$\frac{\partial}{\partial w_i} \log P_w(X = x) = n_i(x) - \sum_{x'} P_w(X = x') n_i(x')$$

La somma è effettuata su tutti i possibili database di osservazioni: la i-esima componente del gradiente è la differenza tra il numero di grounding veri e il suo valore atteso in base al modello corrente. Anche in questo caso il problema non è computazionalmente trattabile:

- Contare il numero di grounding veri di una clausola in un database è $\#P$ completo.
- Calcolare il valore atteso del numero di grounding richiede di effettuare inferenza.

Nel primo caso, il conteggio viene approssimato con tecniche di campionamento. Nel secondo caso, invece, il valore atteso viene approssimato ottimizzando la *pseudo-likelihood*.

Apprendimento discriminativo

In molte applicazioni è noto a priori quali predicati sono osservati e quali sono invece nascosti, con lo scopo di predire correttamente i secondi dati i primi. Nell'apprendimento discriminativo, gli atomi ground vengono quindi divisi nell'insieme di atomi osservabili X e nell'insieme di atomi nascosti Y. La likelihood condizionale di Y dato X è espressa nel seguente modo:

$$P(y|x) = \frac{1}{Z_X} \exp(\sum_{i \in F_Y} w_i n_i(x,y)) = \frac{1}{Z_X} \exp(\sum_{j \in G_Y} w_j g_j(x,y))$$

dove F_y è l'insieme di tutte le clausole della *MLN* che contengono almeno un atomo nascosto, $n_i(x,y)$ è il numero di grounding veri della i-esima clausola con atomi nascosti, G_y è l'insieme di tutte le clausole ground della *MLN* che contengono almeno un atomo nascosto e $g(x,y)$ vale 1 se la j-esima clausola ground è vera nei dati (0 viceversa). La derivata della likelihood condizionale è:

$$\frac{\partial}{\partial w_i} \log P_w(y|x) = n_i(x,y) - E[n_i(x,y)]$$

dove $n_i(x,y)$ è il numero di grounding veri dell'i-esima formula. Anche in questo caso, calcolare il valore atteso $E[n_i(x,y)]$ è intrattabile. In questo caso, il problema viene approssimato applicando l'inferenza *MAP* [24].

Apprendimento strutturale

Un altro tipo di apprendimento è quello di tipo strutturale: oltre ai pesi vengono apprese pure le clausole. Questo tipo di apprendimento può avvenire sia con una *rete di Markov* vuota (dove è presente solo il training set) che non vuota. L'apprendimento strutturale utilizza *beam search*, un algoritmo euristico di ricerca su grafo [25].

Inferenza probabilistica

Una *MLN* può rispondere ad una domanda del tipo: "Qual è la probabilità che la formula F_1 sia vera data F_2?". Se F_1 ed F_2 sono due formule in logica del primo ordine, C un insieme di costanti (che include quelle in F_1 ed F_2) e L è un *MLN*, allora la domanda può essere tradotta in

$$P(F_1|F_2,M_{L,C}) = \frac{P(F_1 \wedge F_2|M_{L,C})}{P(F_2|M_{L,C})} = \frac{\sum_{x\in\chi_{F_1}\cap\chi_{F_2}} P(X=x|M_{L,C})}{\sum_{x\in\chi_{F_2}} P(X=x|M_{L,C})}$$

dove χ_{F_i} è l'insieme di mondi dove F_i è vera. Computare direttamente questa formula non è trattabile, l'inferenza sussume inferenza probabilistica (problema #*P*-completo) ed inferenza logica (problema *NP* anche su domini finiti). È necessario dunque ricorrere a tecniche di approssimazione per ottenere un tempo computazionale accettabile. L'algoritmo di inferenza procede dunque in due fasi:

1. La prima fase produce il minimo sottoinsieme M della rete di Markov ground necessaria a calcolare $P(F_1|F_2,M_{L,C})$

2. La seconda fase calcola l'inferenza sulla rete prodotta dalla prima fase, utilizzando tecniche di campionamento (ad esempio il campionatore di *Gibbs*).

Inferenza *MAP*

In molti casi capita di avere una rete di *Markov* $M_{L,C}$ (dove L è una *MLN* e C un insieme di costanti) ed un insieme di atomi ground osservati per un insieme di predicati O. Indichiamo con H l'insieme di predicati nascosti. L'obiettivo dell'inferenza *MAP* è quello di trovare il mondo più probabile, ovvero il grounding dei predicati nascosti che massimizza la probabilità dato l'insieme di osservazioni:

$$\hat{y} = \arg\max_{y \in Y_{H,C}} p(y|x)$$

dove $Y_{H,C}$ è l'insieme di tutti i possibili mondi costruibili con un insieme dei predicati nascosti H e l'insieme delle costanti C. Consideriamo ora la seguente funzione:

$$s(y,x) = \sum_i w_i n_i(x,y)$$

dove $n_i(x,y)$ indica il numero di grounding veri della i-esima formula dati i predicati osservati x e i predicati nascosti y. Questa funzione scorre quindi tutte le formule della *MLN* per valutare la bontà di una soluzione (x,y). Il problema di inferenza *MAP* può quindi essere riscritto in questi termini:

$$\hat{y} = \arg\max_{y \in Y_{H,C}} s(y,x)$$

Questo problema può essere risolto in diversi modi:

- Approssimando la soluzione con *MaxWalkSAT*: una tecnica di cammino casuale utilizzata per risolvere problemi *SAT* pesati.
- Convertendo il problema in uno di Programmazione Lineare Intera, abbinando alla rete di *Markov* ground una funzione da ottimizzare secondo certi vincoli.

3.1.5 Esempio

Viene ora riportato un esempio per chiarire il concetto di *Markov Logic Network*. Supponiamo di avere un insieme di persone rappresentate dal predicato $Person(x)$. Utilizziamo il predicato $Friends(x,y)$ per indicare che la persona x e la persona y sono amiche, il predicato $Smokes(x)$ per indicare che la persona x è fumatrice ed il predicato $Cancer(x)$ per indicare che la persona x ha il cancro. Definiamo ora una regola che metta in relazione il fatto di fumare e il fatto di avere il cancro:

$$\forall x \; Smokes(x) \Rightarrow Cancer(x) \tag{3.1}$$

Successivamente, possiamo scrivere una regola che relazioni il legame di amicizia tra due persone con il fatto che siano entrambe fumatrici.

$$\forall x,y \; Friends(x,y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y)) \tag{3.2}$$

Queste regole sono di natura probabilistica e non vanno viste come rigide formule della logica del primo ordine. Abbiamo dunque creato una *Markov Logic Network* che viene tradotta nella *rete di Markov* rappresentata in Figura 3.1.

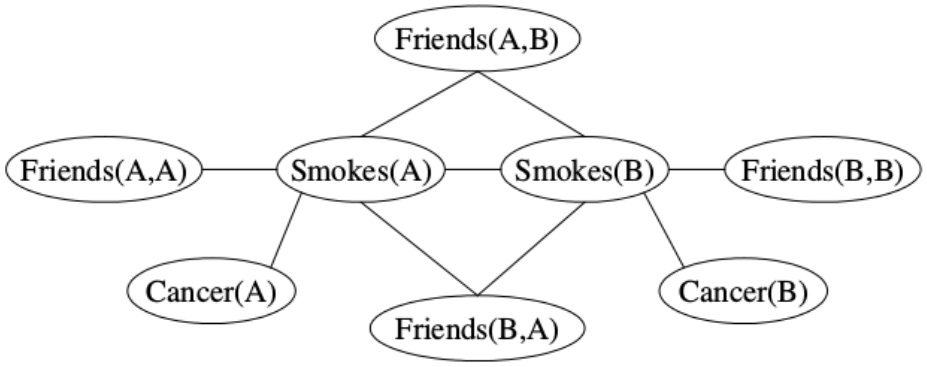

Figura 3.1: *Rete di Markov* prodotta con le regole **??** e **??**

In una prima fase viene utilizzato un *training set* per apprendere i pesi delle regole. Nel nostro caso sarà un insieme di fatti riguardanti un gruppo di persone: legami di amicizia, fumatori e individui affetti da cancro. Verrà considerata vera solo l'informazione espressa esplicitamente (assunzione di mondo chiuso).

Una volta che i pesi sono stati appresi è possibile effettuare inferenza su nuovi dati. Ad esempio potremmo avere informazioni su un insieme di persone (i legami di amicizia e i fumatori) per analizzare la probabilità che una determinata persona abbia il cancro.

3.2 Riconoscimento di attività

In questa sezione viene spiegato il modello progettato e realizzato per riconoscere le attività quotidiane svolte dai pazienti a partire dalle misurazioni prodotte dai sensori. Questo modello è flessibile: non dipende nè dalle specifiche attività che si vogliono riconoscere nè dai sensori utilizzati. Realizzando un'opportuna *Markov Logic Network*, le correlazioni tra misurazioni di sensori ed attività svolte vengono dedotte in una fase di apprendimento e poi utilizzate per l'inferenza.

Le attività svolte vengono riconosciute inferendo le seguenti informazioni:

- Nome dell'attività
- Istante di inizio
- Istante di fine

Tutte le misurazioni di sensore comprese tra l'istante di inizio e istante di fine vengono considerate come facenti parti dell'attività stessa.

In questo lavoro di tesi si assume per semplificazione che non vengano svolte più attività in parallelo, ma al più una per volta.

3.2.1 Costruzione del modello

Veniamo ora alla costruzione del modello per rappresentare il problema di riconoscimento di attività quotidiane tramite *Markov Logic Network*.

Predicati osservabili

Come prima cosa è necessario modellare le osservazioni del sistema. Nel nostro caso le osservazioni corrispondono alle misurazioni prodotte dai sensori. Un singolo rilevamento di sensore può essere rappresentato con il seguente predicato:

$$eventoSensore(evento, timestamp)$$

dove *evento* è il contenuto informativo di una singola misurazione di sensore (ad esempio $aperturaFrigo$) e $timestamp$ è l'istante di tempo nella quale è stata prodotta. Il riconoscimento delle attività si basa su un ragionamento di tipo temporale, ed è quindi necessario aggiungere informazione sull'ordinamento totale sulle misurazioni. Viene quindi usato il predicato

$$prossimoEvento(timestamp_1, timestamp_2)$$

che esplicita il fatto che $timestamp_2$ è l'istante temporale dell'evento successivo alla misurazione avvenuta all'istante $timestamp_1$. Quindi, con i predicati *eventoSensore* e *prossimoEvento*, è possibile catturare tutta l'informazione necessaria per rappresentare le misurazioni prodotte dai sensori.

Gli istanti temporali sono espressi in millisecondi passati dalla mezzanotte, in modo da permettere successivi ragionamenti di tipo temporale tramite espressioni aritmetiche. L'informazione sugli istanti di tempo assoluti (giorno, mese e anno) viene mantenuta a livello applicativo per essere utilizzata in una fase successiva, in modo da ottenere un quadro sulle attività e le anomalie svolte analizzando archi temporali più ampi (ad esempio una settimana).

Predicati nascosti

Veniamo ora a ciò che rappresenta il risultato dell'inferenza in base alle osservazioni, ovvero al riconoscimento delle attività. L'idea è quella di riconoscere l'istante di inizio e l'istante di fine di ogni attività quotidiana effettuata dal paziente. È necessario quindi utilizzare dei predicati nascosti:

- *inizioAttività(nomeAttività,timestamp)* per catturare l'inizio di un'attività.
- *fineAttività(nomeAttività,timestamp)* per catturare la fine di un'attività.

Questi predicati nascosti modellano il contenuto informativo che vogliamo estrarre dal riconoscimento di attività.

Regole

Definiti i predicati che rappresentano le entità di interesse del nostro sistema, lo scopo è quello di scrivere delle regole pesate (dove i pesi vengono impostati dopo una fase di apprendimento) che modellino il riconoscimento di attività. Nel nostro caso, vogliamo partire dai predicati osservabili (*eventoSensore* e *prossimoEvento*) per inferire l'inizio e la fine delle attività svolte dai pazienti (*inizioAttività* e *fineAttività*). Per questo scopo, è necessario scegliere un numero n di eventi di sensori consecutivi che permetta di

inferire la presenza di un inizio o una fine di attività all'interno di quelle n misurazioni. Vediamo per esempio le regole con $n = 1$:

- $eventoSensore(+\mathrm{s}, t) \Rightarrow inizioAttività(+a, t)$
- $eventoSensore(+s, t) \Rightarrow fineAttività(+a, t)$

Queste regole mettono in relazione un unico evento di sensore con l'inizio (o la fine) di un'attività. Ovviamente questo approccio non porta a risultati apprezzabili, essendo una singola misurazione di sensore non sufficiente a catturare l'inizio e la fine di attività.

Il simbolo + precedente ad una variabile esprime il fatto che viene appreso un peso per ogni possibile costante nel dominio della variabile. In questo caso le due regole hanno abbinato un peso per ogni coppia di s e a.

Vediamo ora le regole utilizzabili con $n = 2$:

- $eventoSensore(+s_1, t_1) \wedge eventoSensore(+s_2, t_2) \wedge prossimoEvento(t_1, t_2)$
  $\Rightarrow inizioAttività(+a, t_1)$
- $eventoSensore(+s_1, t_1) \wedge eventoSensore(+s_2, t_2) \wedge prossimoEvento(t_1, t_2)$
  $\Rightarrow fineAttività(+a, t_2)$

In questo caso è necessario utilizzare anche il predicato *prossimoEvento*. In questo modo le regole vengono attivate se e solo se i due eventi di sensori sono consecutivi. La semantica delle seguenti regole è questa: per ogni coppia di rilevazioni di sensori consecutivi, si cerca di inferire se la prima misurazione è un inizio o se la seconda è una fine.

Nel capitolo successivo viene presentata la valutazione sperimentale effettuata per selezionare la *MLN* che massimizza la qualità di predizione in base ai dati raccolti.

3.2.2 Raffinamento delle attività riconosciute

Il riconoscimento di attività ha un approccio di tipo probabilistico e, in quanto tale, è soggetto ad errori nelle predizioni (per quanto buono possa essere il modello selezionato). Un *falso positivo* è un inizio (o una fine) di attività inferito dal processo di riconoscimento ma non avvenuto. Un *falso negativo* è invece un inizio (o una fine) di attività non inferito dal processo di riconoscimento ma in realtà avvenuto. Per estrarre quindi le attività riconosciute (prendendo le misurazioni dei sensori tra un inizio ed una fine), è necessario prestare attenzione a queste problematiche, cercando di correggere (se possibile) le eventuali mancanze dell'algoritmo di riconoscimento. Vengono ora introdotte le euristiche utilizzate per correggere il problema di predizioni sbagliate:

- Se viene riconosciuto più di un inizio di una stessa attività in diversi istanti prima della fine della stessa, viene considerato solo il primo. Gli altri inizi vengono trattati come falsi positivi.

- Se viene riconosciuta più di una fine di una stessa attività in diversi istanti prima dell'inizio di un'altra attività, viene considerata solo l'ultima. Le altre fini vengono considerate come falsi positivi.

- Se è presente l'inizio di un'attività ma non la fine della stessa vuol dire che la fine dell'attività è un falso negativo. Per correggere questo problema, viene aggiunta la fine mancante usando come istante temporale quello della misurazione di sensore precedente all'inizio dell'attività successiva. Se non sono presenti successive annotazioni (si è arrivati alla fine del file contenente il risultato dell'inferenza) la fine mancante viene aggiunta usando l'istante temporale dell'ultima misurazione di sensore presente nei dati.

- Se è presente la fine di un'attività ma non l'inizio della stessa vuol dire che l'inizio dell'attività è un falso negativo. Per correggere questo problema, viene aggiunto l'inizio mancante usando come istante temporale quello della misurazione di sensore seguente alla fine dell'attività precedente. Se non sono presenti precedenti annotazioni (la fine di attività considerata è la prima annotazione del file contenente il risultato dell'inferenza) l'inizio mancante viene aggiunto usando l'istante temporale della prima misurazione di sensore presente nei dati.
- Se l'inizio di un'attività è accoppiato con la fine di un'attività con un nome differente, significa che la predizione della fine della prima attività e la predizione dell'inizio della seconda attività sono dei falsi negativi (non sono stati riconosciuti). In questo caso non è possibile ricostruire le due attività e le misurazioni di sensore comprese tra queste due annotazioni vengono rimossi.

3.3 Riconoscimento di anomalie

In questa sezione viene introdotto il modello sviluppato per il riconoscimento di anomalie. Questo modulo prende come input ogni attività riconosciuta dal riconoscitore di attività nel seguente formato:

- Nome dell'attività riconosciuta (ad esempio "*assumere le medicine*").
- Momento di inizio
- Momento di fine
- Misurazioni di sensori coinvolte

Combinando questi dati con informazioni specifiche sul paziente che si intende monitorare (ad esempio le medicine che deve assumere in determinati orari) l'obiettivo è quello di inferire anomalie tramite un sistema a regole della logica del primo ordine.

3.3.1 Costruzione del modello

Viene ora spiegata nei dettagli la modellazione di una base di conoscenza della logica del primo ordine volta ad analizzare le attività riconosciute, in modo tale da inferire gli eventuali comportamenti anomali effettuati dai pazienti durante l'esecuzione delle stesse.

Rappresentazione delle attività

Un'attività è rappresentata da un nome, un istante di inizio, un istante di fine e le misurazioni dei sensori coinvolte. Le prime tre informazioni possono essere condensate in un unico predicato nel seguente modo:

$$attività(id, nome, istante\_inizio, istante\_fine)$$

In questo modo si ottiene una rappresentazione compatta sulle informazioni principali dell'attività. Il termine *id* è un identificativo che viene utilizzato per distinguere in maniera univoca le singole istanze di attività. Gli istanti di inizio e di fine attività vengono espressi in numero di millisecondi passati dalla mezzanotte del giorno in cui sono stati misurati, in modo tale da permettere un successivo ragionamento di tipo temporale tramite espressioni aritmetiche.

Il passo successivo è quello di modellare l'interazione del paziente con l'ambiente in base alle misurazioni prodotte dai sensori. A questo scopo viene utilizzato il seguente predicato:

$$evento(azione, oggetto, timestamp, id_{attività})$$

Il termine *azione* rappresenta un'azione effettuata sull'ambiente (apertura, chiusura, accensione, ...) svincolandosi dallo specifico oggetto domestico *oggetto* sul quale è

stata effettuata, in modo tale da ottenere regole più generiche per il riconoscimento di anomalie. Il termine *timestamp* è l'istante di tempo nel quale l'evento è stato svolto, rappresentato sempre in millisecondi trascorsi dalla mezzanotte dello stesso giorno. Infine, il termine $id_{attività}$ è utile per collegare l'evento all'attività alla quale è riferito. Per concludere la modellazione di attività, è necessario introdurre un ultimo predicato:

$$prossimoEvento(timestamp_1, timestamp_2)$$

Questo predicato permette di esprimere il fatto che gli eventi avvenuti agli istanti di tempo $timestamp_1$ e $timestamp_2$ (sempre espressi in millisecondi dalla mezzanotte) sono consecutivi.

Rappresentazione di informazioni aggiuntive

Per la corretta rilevazione delle anomalie, è necessario conoscere delle informazioni che rappresentano una specifica conoscenza del paziente e dell'ambiente nel quale è situato. Uno degli aspetti principali, ad esempio, è quello di specificare le prescrizioni dei medicinali che il paziente deve assumere, in modo tale da evidenziare dimenticanze, ripetizioni, farmaci assunti erroneamente, eccetera. In questo specifico caso, viene utilizzato il seguente predicato:

$$prescrizione(nome\_medicina, inizio\_prescrizione, fine\_prescrizione)$$

Questo predicato indica che la medicina *nome_medicina* deve essere assunta in un istante temporale compreso tra *inizio_prescrizione* e $fine\_prescrizione$. In base al giorno della settimana le prescrizioni dei medicinali possono differire. È compito dell'applicazione quello di utilizzare le prescrizioni corrette in base al giorno della settimana corrente.

Un altro esempio di informazione aggiuntiva è quello che descrive quali alimenti presenti nell'ambiente domestico siano da cuocere e quali no, in modo tale da evidenziare anomalie (ad esempio si sta cuocendo il succo di frutta, un alimento che non dovrebbe essere cotto). A questo scopo è possibile utilizzare il predicato

$$cuocibile(nome\_alimento)$$

che indica che l'alimento *nome_alimento* può essere cotto.

Rappresentazione dello stato di un oggetto

Per inferire le anomalie, è risultato necessario aggiungere un predicato ausiliare per gestire il concetto di stato. Lo stato è una proprietà di uno specifico oggetto domestico in un determinato intervallo temporale.

$$stato(nome\_stato, oggetto, timestamp_1, timestamp_2)$$

Questo predicato non fa parte delle osservazioni, ma deve essere inferito. È necessario dunque formalizzare delle regole per riuscire ad ottenere l'informazione sullo stato degli oggetti domestici nei vari intervalli temporali.

Prendiamo come esempio di riferimento il monitoraggio di apertura e chiusura di un generico sportello. Quando viene rilevato l'evento *apertura* di uno specifico sportello, il suo stato di *aperto* inizia ad essere vero a partire da quell'istante temporale. Questo stato continua ad essere vero finchè lo sportello non viene chiuso. Formalizziamo ora questi concetti nella logica del primo ordine in maniera induttiva:

1. $evento(apertura, Sportello, T) \Rightarrow stato(aperto, Sportello, T, T)$

2. $stato(aperto, Sportello, T_i, T_j) \wedge \neg evento(chiusura, Sportello, T_k, A_{id}) \wedge$
   $\wedge\, prossimoEvento(T_j, T_k) \Rightarrow stato(aperto, Sportello, T_i, T_k)$

La prima regola specifica che se si verifica l'apertura di un generico sportello ad un istante di tempo t, il suo stato di apertura è vero nell'istante di tempo che va da t a t (necessario come passo base dell'induzione). La seconda regola definisce il passo induttivo: se uno sportello è aperto nell'intervallo da t_i a t_j e se l'evento avvenuto all'istante t_k (consecutivo a quello avvenuto in t_j) non è la chiusura dello sportello, lo stato continua ad essere vero nell'intervallo che va da t_i a t_k. Un fatto importante da notare è che $Sportello$,T, T_i, T_j, T_k ed A_{id} sono tutte variabili, le uniche costanti sono $apertura$ e $aperto$. A_{id} rappresenta l'identificatore dell'attività abbinata all'evento, informazione non rilevante per quanto riguarda l'inferenza dello stato.

Rappresentazione delle anomalie

L'ultimo concetto che rimane da rappresentare nel modello è quello di anomalia. Introduciamo quindi il seguente predicato:

$$anomalia(id_{attività}, nome\_anomalia, oggetto, timestamp)$$

Questo predicato rappresenta un'anomalia dal nome *nome_anomalia* che coinvolge l'oggetto domestico *oggetto* all'istante di tempo $timestamp$, ed è legata all'attività identificata da $id_{attività}$. Analizziamo nei dettagli tutti gli attori in gioco:

- Il nome dell'anomalia rappresenta in maniera univoca il comportamento anomalo svolto dal paziente (medicina dimenticata, fornello non spento, sportello lasciato aperto, ...).
- L'oggetto coinvolto aggiunge un'informazione che arricchisce il contenuto informativo dell'anomalia riportata. È possibile, ad esempio, indicare quale pre-

cisa medicina è stata dimenticata, quale sportello non è stato chiuso dopo essere stato aperto, quale fornello non è stato spento, eccetera.

- L'istante temporale è espresso nel formato utilizzato anche negli altri predicati, ovvero in millisecondi passati dalla mezzanotte. Nel caso in cui l'anomalia prodotta sia una dimenticanza (ad esempio il paziente si è dimenticato di prendere una medicina) è chiaro che l'anomalia non è abbinabile ad un determinato istante temporale. In questo caso, l'istante temporale utilizzato è la fine prevista nella prescrizione di quell'evento (ad esempio, se il paziente deve prendere una particolare medicina dalle 10:00 alle 12:00 ma si dimentica, l'anomalia prodotta avrà come istante temporale 12:00).

- L'identificativo dell'attività relativa all'anomalia è utile per abbinare l'attività in cui il comportamento anormale è stato rilevato. Tuttavia, nel caso di un omissione (il paziente si è dimenticato di prendere un medicinale prescritto) non è possibile abbinare l'anomalia con un'attività. In questo caso, il campo *id_attività* viene impostato con il valore *null*.

Il predicato *anomalia* viene utilizzato per l'inferenza logica e non sarà ovviamente mai presente nelle osservazioni. Analizziamo ora due esempi differenti di regole per il riconoscimento di anomalie:

1. $stato(acceso, Fornello, T_1, T_2) \wedge (T2 - T1 > threshold)$
   $\Rightarrow anomalia(Aid, dimenticato\_acceso, Fornello, T_2)$

2. $prescrizione(Medicina, T_1, T_2) \wedge \neg evento(presa, Medicina, T, Id_x) \wedge$
   $\wedge (T \geq T_1) \wedge (T \leq T2) \Rightarrow anomalia(null, medicina\_dimenticata, Medicina, T_2)$

La prima regola esplicita che se il fornello è nello stato *acceso* da T_1 a T_2 e la durata di questo intervallo $(T_2 - T_1)$ supera una certa soglia *threshold* calibrata precedentemente, sussiste il comportamento anomalo di aver lasciato acceso il fornello troppo a

lungo. La seconda, invece, specifica che se una medicina *Medicina* è prescritta tra T_1 e T_2 ma in questo intervallo non è stata presa, sussiste il comportamento anomalo di essersi dimenticati di assumerla. Come si può notare, in questo caso di dimenticanza l'identificativo dell'attività non può essere ricavato e viene impostato a *null*. Per lo stesso motivo, l'istante temporale dell'anomalia coincide quello di fine prescrizione.

Capitolo 4

Valutazione sperimentale

In questo capitolo vengono presentati gli esperimenti effettuati per modellare e validare i moduli di riconoscimento. A tale scopo è stato utilizzato un dataset ottenuto simulando il comportamento di pazienti nell'esecuzione di attività ed anomalie. In seguito, viene presentata l'implementazione prototipale di un modulo software per il riconoscimento automatico di attività ed anomalie effettuate da pazienti affetti da deterioramento cognitivo all'interno delle proprie abitazioni. Questo modulo si appoggia al software *thebeast* (un'implementazione di *Markov Logic Network*) per il riconoscimento di attività e a *tuProlog* (un sistema di inferenza della logica del primo ordine) per quanto riguarda il riconoscimento di anomalie.

La Sezione 4.1 presenta le diverse implementazioni di *Markov Logic Network* che sono state studiate ed analizzate in questo lavoro di tesi in modo da selezionare la più adeguata al modello di riconoscimento di attività proposto. In Sezione 4.2 viene descritta la sperimentazione effettuata per selezionare il modello da utilizzare per il riconoscimento di attività. La Sezione 4.3 presenta i risultati sperimentali ottenuti con il sistema di riconoscimento di anomalie. Infine, la Sezione 4.4 illustra ad alto livello l'architettura del software prodotto.

4.1 Software per *Markov Logic Network*

In questa sezione vengono introdotte le diverse implementazioni di *Markov Logic Network* che sono state studiate in questo lavoro di tesi. Lo scopo di questa analisi è il confronto tra i vari strumenti disponibili, in modo tale da poter scegliere il più adeguato per realizzare il sistema di riconoscimento di attività presentato nel precedente capitolo. I requisiti che vengono considerati più importanti sono l'integrabilità all'interno di un'applicazione *Android*, l'efficienza computazionale e la possibilità di gestire espressioni aritmetiche per ragionamenti di tipo temporale.

4.1.1 *Alchemy*

Il sistema *Alchemy* [23], realizzato da un gruppo di ricerca dell'Università di Washington, offre una serie di algoritmi di apprendimento ed inferenza basati sulle *Markov Logic Network*. Il software è sviluppato nel linguaggio di programmazione *C++*. La sintassi di *Alchemy* è molto semplice, il che permette di poter modellare con facilità le formule necessarie. *Alchemy* mette a disposizione algoritmi di apprendimento generativi, discriminativi e strutturali. Inoltre, è supportata sia l'inferenza di tipo *MAP* che l'inferenza basata sulle probabilità marginali. Diverse operazioni aritmetiche e su stringhe sono supportate, ed è inoltre possibile implementare in poche righe di *C++* delle funzioni aggiuntive. Tuttavia, *Alchemy* soffre di svariate problematiche:

- Può essere eseguito solo su sistemi *Linux*. L'unico modo possibile per esportarlo su altre piattaforme è quello di modificare manualmente il codice sorgente.
- È stato testato utilizzando un preciso compilatore *C++*. Se vengono usati compilatori diversi (anche se più recenti) è possibile riscontrare svariati errori durante l'esecuzione.

4.1.2 *thebeast*

Il software *thebeast* [26] è stato realizzato in *Java* dal ricercatore *Sebastian Riedel* dell'università di Edimburgo nel 2008. Le *MLN* in *thebeast* sono leggermente differenti rispetto a come sono state spiegate precedentemente. Vengono introdotte due estensioni:

- Il peso di una formula può essere differente per ogni grounding della stessa.
- Il peso di una formula ground può essere scalato per un valore reale.

L'interazione con *thebeast* avviene tramite un linguaggio di scripting molto semplice, tramite il quale è possibile accedere (con una shell) alle varie funzionalità: definizione di un modello, apprendimento dei parametri, inferenza ed impostazione dei vari parametri. *thebeast* offre differenti algoritmi per effettuare apprendimento di tipo discriminativo. Inoltre, offre un algoritmo di inferenza di tipo *MAP* molto efficiente che combina il metodo Cutting Plane alla programmazione lineare intera [26]. *thebeast* supporta i vincoli di cardinalità e semplici operazioni aritmetiche. Purtroppo la sintassi è molto complessa, fatto che rende difficile il processo di modellazione delle regole.

4.1.3 *Tuffy*

Il software *Tuffy* [27] è stato sviluppato nel linguaggio *Java* da un gruppo di ricerca dell'Università del Wisconsin-Madison. *Tuffy* si appoggia ad un database relazionale per migliorare e velocizzare la fase di grounding. Inoltre, utilizza una tecnica chiamata *Markov Random Field partitioning* per migliorare la fase di inferenza. Questa tecnica sfrutta il parallelismo per dividere il problema generale in sottoproblemi, in modo da diminuire sensibilmente il tempo computazionale richiesto. Grazie a queste tecniche, il tempo computazionale complessivo è esponenzialmente ridotto rispetto a quello ot-

tenibile con il sistema *Alchemy*. Un'altra innovazione rispetto agli altri strumenti è quella di poter avere osservazioni *soft*: un atomo osservato può essere abbinato ad una probabilità a priori. La sintassi di *Tuffy* è molto simile a quella di *Alchemy*, con la restrizione che le regole devono obbligatoriamente essere in forma clausale o in forma implicativa. Un punto debole è che per apprendere un peso per ogni valore di una particolare variabile di una formula è necessario esplicitare ogni possibile grounding (negli altri strumenti può essere svolto automaticamente se specificato). Inoltre, la stretta dipendenza da un database relazionale implica problematiche di portabilità.

4.1.4 *RockIt*

Il software *RockIt* [28] è stato sviluppato da un gruppo di ricerca dell'Università di Mannheim. *RockIt* si appoggia al database relazionale *MySQL* e al solutore di programmazione lineare intera *Gurobi* per effettuare inferenza unendo la tecnica di *Cutting Plane Inference* [26] e quella di *Cutting Plane Aggregate* [28]. Con la combinazione di queste tecniche è possibile ottenere tempi computazionali molto ridotti. L'inferenza in *RockIt* può essere sia *MAP* che basata sulle probabilità marginali. L'apprendimento avviene tramite un algoritmo di tipo discriminativo chiamato *Voted Percepetron*. Un altro punto di forza di *RockIt* è la sua disponibilità tramite un webservice *RESTful*, caratteristica particolarmente interessante in ambito di sistemi pervasivi. Come punto debole, *RockIt* non è in grado di gestire espressioni aritmetiche.

4.2 Valutazione del riconoscimento delle attività

In questa sezione viene presentato il processo sperimentale effettuato per modellare e validare una *Markov Logic Network* in grado di inferire dalle misurazioni prodotte dai sensori le attività svolte dal paziente. In questo prototipo, le attività che si intende riconoscere sono le seguenti:

- Assunzione di medicinali prescritti dai medici
- Preparazione di pasti
- Consumazione di pasti

L'implementazione di *Markov Logic Network* scelta è stata quella di *thebeast*, che è risultata la più adeguata a rappresentare il modello di riconoscimento proposto e ad essere integrata all'interno di un applicativo *Android*.

4.2.1 Metriche

Una *MLN* impara ad abbinare un peso per ogni regola utilizzando un insieme che contiene misurazioni di sensore e attività quotidiane coinvolte (detto *training set*), in maniera tale da apprenderne la correlazione. Il modello predittivo generato con l'apprendimento può essere testato effettuando inferenza su un insieme di nuove osservazioni (non presenti nel *training set*) chiamato *test set*. Anche nel *test* set sono presenti le informazioni sulle attività svolte, in modo tale da poterle confrontare con le attività inferite dal modello predittivo utilizzando opportune metriche:

$$Precision = \frac{VP}{VP+FP}$$

$$Recall = \frac{VP}{VP+FN}$$

$$F_1 = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall)}$$

dove VP indica il numero di *veri positivi* (inizio e fine di attività inferiti in maniera corretta), FP il numero di *falsi positivi* (inizio e fine di attività inferiti ma non avvenuti) ed infine FN indica il numero di *falsi negativi* (inizio e fine di attività non inferiti ma in realtà avvenuti). La *Precision* indica la percentuale di inizio e fine di attività

predetti in maniera corretta su tutti quelli effettivamente predetti. La *Recall* indica la percentuale di inizio e fine di attività predetti in maniera corretta su tutti gli inizi e fini di attività effettivamente avvenuti. Infine, la metrica *F1* unisce i valori di *Precision* e *Recall* per ottenere un'informazione più concisa sulla qualità del modello predittivo. È importante notare che viene considerato un errore di predizione anche nel caso in cui l'inizio e la fine delle attività vengono predetti in istanti temporali diversi ma molto adiacenti rispetto a quelli effettivi.

4.2.2 Risultati sperimentali

Per validare uno specifico modello viene effettuato un processo di *k-fold Cross Validazione*. Questo processo prevede di partizionare il dataset in k parti di uguale dimensione. Una delle k parti viene utilizzata come *test set*, mentre le restanti $k-1$ vengono utilizzate come *training set*. Questo processo viene iterato k volte, dove ad ogni iterazione ogni partizione del dataset viene utilizzata esattamente una volta come *test set*. In questo modo è possibile ottenere un valore medio dei valori di *Precision*, *Recall* e F_1 ed avere una stima più consistente della qualità predittiva del modello. Il dataset utilizzato è quello introdotto nella Sezione 2.3, e prevede 21 giornate di attività quotidiane svolte da 3 pazienti distinti. Questo dataset viene diviso in 21 partizioni (una per ogni giornata di attività) per effettuare la *k-fold Cross-Validazione* sui vari modelli analizzati. Per dedurre gli istanti di inizio e di fine di un'attività vengono analizzate sequenze di n misurazioni di sensore consecutive. L'obiettivo è quello di trovare il valore di n che massimizzi la qualità delle predizioni. La sperimentazione è consistita nel costruire differenti *MLN* con diversi valori di n e diverse combinazioni di regole con lo scopo di effettuare la *Cross Validazione* per ottenere le metriche di *Precision*, *Recall* ed F_1. A questo scopo è stato sviluppato uno script in *Python* che implementa la *Cross-Validazione* su un singolo modello, in modo da ottenere i valori *Precision*,

Recall ed F_1. Questo programma è stato eseguito con tutti i modelli considerati, in modo tale da poterli confrontare in termini di qualità di predizione.

Caso *n=1*

Nel caso $n = 1$ viene utilizzata l'informazione di un solo sensore per il riconoscimento di inizio e di fine di un'attività. In questo caso esiste un'unica combinazione di regole:

- $eventoSensore(+e,t) \Rightarrow inizioAttività(+a,t)$
- $eventoSensore(+e,t) \Rightarrow fineAttività(+a,t)$

Per ogni regola viene appreso un peso per ogni possibile abbinamento di una misurazione di sensore con un'attività.

Recall	Precision	F1
0.658	0.739	0.693

Tabella 4.1: Caso $n = 1$

I risultati ottenuti non sono soddisfacenti. Questo è dovuto al fatto che il dato proveniente da un singolo evento di sensore non è sufficiente a catturare l'inizio o la fine di un'attività.

Caso *n=2*

In tutti i casi dove $n \geq 2$ è possibile ottenere diverse combinazioni di regole a parità del valore di n. Il primo esperimento per $n = 2$ prevede l'utilizzo delle seguenti due regole:

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$ $\Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$ $\Rightarrow fineAttività(+a,t_2)$

La prima regola cerca di correlare una coppia di misurazioni consecutive con un inizio di attività che avviene alla prima di esse, mentre la seconda cerca di correlare una coppia di misurazioni consecutive con una fine di attività che avviene alla seconda di esse.

Recall	Precision	F1
0.785	0.799	0.790

Tabella 4.2: Prima combinazione per $n = 2$

I risultati sono migliorati rispetto al caso $n = 1$, in quanto l'informazione due sensori consecutivi è più rilevante per il riconoscimento di inizio e di fine.

Un secondo esperimento possibile con $n = 2$ è quello di aggiungere alle regole appena viste quelle del caso $n = 1$, per valutare se aiutano a migliorare l'accuratezza delle predizioni.

- $eventoSensore(+e,t) \Rightarrow inizioAttività(+a,t)$
- $eventoSensore(+e,t) \Rightarrow fineAttività(+a,t)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$ $\Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$ $\Rightarrow fineAttività(+a,t_2)$

Recall	Precision	F1
0.773	0.801	0.785

Tabella 4.3: Seconda combinazione per $n = 2$

Come si evince dai risultati, in questo caso si ottiene una diminuzione del valore di *Recall*. Questo è dovuto al fatto che le regole che catturano $n = 1$ non sono significative e che comportano addirittura ad un peggioramento delle predizioni.

Caso *n=3*

Analizziamo ora il caso $n = 3$, il quale permette di provare un numero più alto di combinazioni di regole differenti. Il primo esperimento prevede di estendere le regole viste precedentemente utilizzando tre misurazioni di sensori consecutivi:

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_3)$

Recall	Precision	F1
0.753	0.845	0.795

Tabella 4.4: Prima combinazione per $n = 3$

Dai risultati è si evidenzia un miglioramento del valore di *Precision* e un peggioramento del valore di *Recall*. Questo vuol dire che, con l'aumento delle misurazioni di sensori utilizzate, questa tipologia di regole tende a diminuire il numero di inizio o fine di attività riconosciuti.

Il prossimo esperimento prevede di accostare le regole appena viste con quelle usate per il caso $n = 2$.

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_3)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow inizioAttività(+a,t_1)$

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow fineAttività(+a,t_2)$

In questo caso si ottiene un ulteriore peggioramento della qualità di predizione. Quello che si può dedurre è che le regole che cercano di correlare l'inizio di attività con la prima misurazione di n consecutive (o con l'ultima in caso di fine) non sono particolarmente utili.

Recall	Precision	F1
0.745	0.841	0.788

Tabella 4.5: Seconda combinazione per $n = 3$

Visti i risultati, è necessario trovare alternative a questo approccio che migliorino la predizione.

Il prossimo esperimento prevede regole che analizzano ancora tre misurazioni di sensore consecutive. In questo caso, però, si cerca di correlare sia l'inizio che la fine di un'attività con la misurazione centrale:

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_2)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_2)$

Recall	Precision	F1
0.895	0.987	0.938

Tabella 4.6: Terza combinazione per $n = 3$

La qualità di predizione è notevolmente migliorata, dovuto al fatto che questo tipo di regole cattura il passaggio tra un'attività e l'altra.

Il prossimo esperimento cerca di catturare lo stesso tipo di eventi, utilizzando però esclusivamente regole che analizzano coppie di misurazioni di sensori:

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2) \Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2) \Rightarrow inizioAttività(+a,t_2)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2) \Rightarrow fineAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2) \Rightarrow fineAttività(+a,t_2)$

In questo caso, il riconoscimento di inizio (e analogamente quello di fine) viene guidato da due regole distinte, catturando però lo stesso evento dell'esperimento precedente.

Recall	Precision	F1
0.967	0.967	0.967

Tabella 4.7: Quarta combinazione per $n = 3$

I risultati mostrano un ulteriore miglioramento della predizione.

L'ultimo esperimento effettuato per il caso $n = 3$ prevede di unire le regole degli ultimi due modelli:

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge \wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_2)$

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_2)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow inizioAttività(+a,t_2)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow fineAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow fineAttività(+a,t_2)$

Recall	Precision	F1
0.965	0.974	0.968

Caso *n=4*

In fase sperimentale è stato appurato che utilizzare regole che analizzano direttamente 4 o più misurazioni di sensori produce un tempo computazionale non accettabile in fase di apprendimento. Questo è dovuto al fatto che il numero di grounding aumenta esponenzialmente con l'aumentare della lunghezza delle regole. È necessario quindi utilizzare combinazioni di regole che analizzano un numero di sensori più piccolo.

Il seguente modello combina regole che analizzano 2 e 3 sensori.

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_1)$

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_3)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow inizioAttività(+a,t_2)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow fineAttività(+a,t_1)$

Recall	Precision	F1
0.968	0.962	0.964

Tabella 4.8: Prima combinazione per $n = 4$

In questo caso i risultati ottenuti sono molto simili ai precedenti.

Il prossimo modello aggiunge al precedente ulteriori regole da due sensori, per valutare se la qualità della predizione migliora.

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_3)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow inizioAttività(+a,t_2)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow fineAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow inizioAttività(+a,t_1)$

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2) \Rightarrow fineAttività(+a,t_2)$

Recall	Precision	F1
0.964	0.964	0.963

Tabella 4.9: Seconda combinazione per $n = 4$

I risultati sono praticamente invariati, dovuto probabilmente al fatto che le due regole aggiunte non introducono nuova informazione (coprono le stesse misurazioni analizzate dalle regole a tre sensori).

Caso *n=5*

Anche in questo caso verranno usate esclusivamente regole che tratteranno al massimo 3 misurazioni consecutive di sensori.

In questo esperimento vengono combinate quattro regole, dove ognuna analizza tre misurazioni di sensori:

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge \wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge \wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_3)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge \wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_3)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge \wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_1)$

Recall	Precision	F1
0.945	0.952	0.947

Tabella 4.10: Prima combinazione per $n = 5$

In questo modello vengono considerate 5 misurazioni di sensore consecutive, e la misurazione centrale viene correlata con l'inizio e la fine di attività.
I risultati prodotti sono buoni, anche se leggermente più bassi dei modelli precedenti.

Nel prossimo esperimento vengono aggiunte regole a due sensori al modello precedente:

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_3)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow inizioAttività(+a,t_3)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge eventoSensore(+e_3,t_3) \wedge$
  $\wedge prossimoEvento(t_1,t_2) \wedge prossimoEvento(t_2,t_3) \Rightarrow fineAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow inizioAttività(+a,t_1)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow inizioAttività(+a,t_2)$
- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow fineAttività(+a,t_1)$

- $eventoSensore(+e_1,t_1) \wedge eventoSensore(+e_2,t_2) \wedge prossimoEvento(t_1,t_2)$
  $\Rightarrow fineAttività(+a,t_2)$

Recall	Precision	F1
0.962	0.964	0.962

Tabella 4.11: Seconda combinazione per $n = 5$

Come si può notare la qualità predittiva è migliorata, tornando ai livelli ottenuti con i modelli precedenti.

4.2.3 Conclusioni

In base agli esperimenti condotti, possiamo osservare l'andamento della qualità dei risultati in Figura 4.1 esaminando la variazione della misura F_1 con la crescita di n (per ogni valore di n è stato considerato solo il modello migliore). La misura F_1 viene utilizzata in quanto rappresentativa della qualità predittiva di un singolo modello.

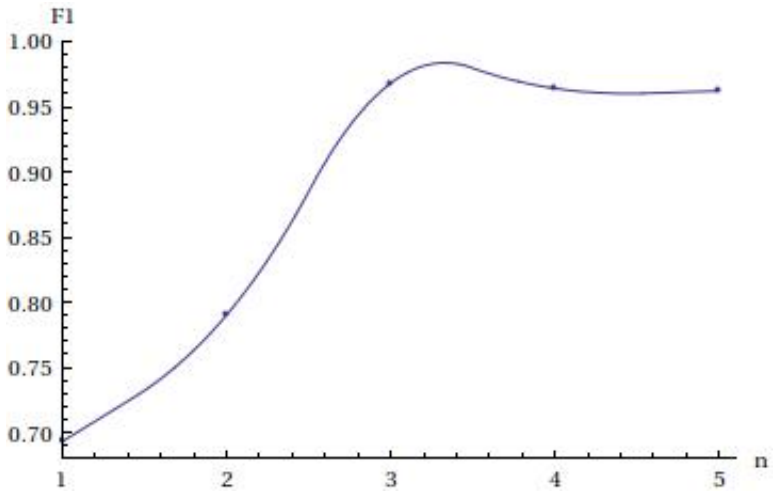

Figura 4.1: *Andamento di F_1 al crescere di n*

La scelta è quindi ricaduta su $n = 3$, optando per il modello che ha presentato una qualità di predizione migliore e un tempo computazionale accettabile.

4.3 Valutazione del riconoscimento delle anomalie

In questa sezione viene spiegato il processo sperimentale di validazione del modulo di riconoscimento di anomalie. A differenza del riconoscimento di attività, il modello non è probabilistico: delle regole della logica del primo ordine sono state codificate in *Prolog* in modo tale da catturare le eventuali anomalie presenti dei dati sfruttando la conoscenza specifica sul paziente (ad esempio i medicinali che deve assumere). La validazione è avvenuta calcolando il numero di *veri positivi*, *falsi positivi* e *falsi negativi* prodotti dall'inferenza sul dataset acquisito, utilizzando come motore di inferenza *tuProlog*.

4.3.1 *tuProlog*

Per implementare il sistema di ragionamento basato sulla logica del primo ordine per inferire le anomalie viene utilizzato *Prolog*, un linguaggio di programmazione che adotta il paradigma di programmazione logica. La programmazione logica differisce significativamente dalla programmazione tradizionale, in quanto consente al programmatore di descrivere la struttura logica del problema piuttosto che il modo di risolverlo. È compito del programmatore far sì che il programma logico rappresenti adeguatamente il problema. È invece compito del sistema *Prolog* utilizzare le informazioni presenti nel programma per effettuare le operazioni necessarie a dare risposte al problema, sfruttando un meccanismo di deduzione logica insito in esso [29].

Il motore di inferenza logica *tuProlog* [30] è una versione lightweight di *Prolog* progettata per avere le seguenti caratteristiche:

- Facilmente installabile: l'unico requisito è la presenza di una *Java VM* standard invocando un singolo file *JAR*.

- Minimale: Il nucleo di *tuProlog* consiste in un contenuto oggetto *Java* che contiene esclusivamente le proprietà essenziali di un motore di inferenza *Prolog*.
- Configurabile: un semplice e potente meccanismo permette di caricare e rimuovere predicati, funtori ed operatori (sia staticamente che dinamicamente).
- Integrato con *Java*: *tuProlog* può essere invocato e usato come un semplice oggetto *Java* all'interno di applicazioni (anche *Android*).
- Interoperabile: *tuProlog* supporta interazione tramite *TCP/IP* e *RMI*.

Viste le caratteristiche, *tuProlog* è particolarmente adatto in ambito di sistemi pervasivi. In questo lavoro di tesi viene utilizzato da un'applicazione *Android* come modulo esterno per inferire le anomalie all'interno delle attività svolte dal paziente. L'inferenza logica avviene

- L'inizio e la fine delle attività riconosciute vengono codificati in linguaggio *Prolog* come fatti in base al modello introdotto precedentemente.
- Le informazioni aggiuntive (ad esempio le prescrizioni dei medicinali) vengono codificate in linguaggio *Prolog* come fatti in base al paziente che deve essere monitorato.
- Per completare la base di conoscenza, vengono aggiunte le regole necessarie a riconoscere le anomalie.
- La base di conoscenza viene interrogata per inferire predicati di tipo *anomalia*, che rappresentano gli specifici comportamenti anomali svolti dal paziente (ad esempio dimenticare di assumere un determinato medicinale).
- Il risultato dell'interrogazione viene estratto e convertito in opportuni messaggi da trasmettere al servizio di telemedicina.

4.3.2 Anomalie riconosciute

Vengono ora introdotte le anomalie che vengono riconosciute nel prototipo realizzato per ognuna delle attività considerate. Le anomalie sono state specificate nell'ambito del progetto *SECURE* dal partner *Fatebenefratelli* e codificate in linguaggio *Prolog* in collaborazione con un dottorando del laboratorio *Everyware*.

Anomalie riguardanti l'assunzione di medicinali

- *Medicina non assunta*: Il paziente si dimentica di assumere un medicinale che è stato prescritto dal medico.
- *Medicina errata*: Il paziente ha assunto un medicinale in un giorno o in una fascia oraria errati.
- *Medicina assunta ripetutamente*: Il paziente si dimentica di aver assunto un medicinale, ripetendo l'azione a distanza di poco tempo.
- *Medicinale non rimesso a posto*: Il paziente si dimentica di rimettere il medicinale assunto nell'armadietto delle medicine.

Anomalie riguardanti la preparazione di pasti

- *Fornello non spento*: Il fornello rimane acceso per una quantità di tempo eccessiva.
- *Recipiente di cottura non utilizzato*: Durante la preparazione di un alimento che necessita di essere cotto, il paziente si dimentica di prendere un recipiente di cottura (ad esempio una pentola).
- *Fornello non acceso*: Il fornello non è stato utilizzato durante la preparazione di un alimento che necessita la cottura.

- *Mancata preparazione di un pasto*: Il paziente non prepara un pasto (colazione, pranzo o cena). Questa anomalia diventa rilevante se viene ripetuta frequentemente nel tempo, dimostrando che il paziente ha smesso di cucinare (può essere sintomo di deterioramento cognitivo).

Anomalie riguardanti la consumazione dei pasti

Nel caso della consumazione dei pasti, l'unica anomalia considerata è *Posate non utilizzate*: il paziente si dimentica di prendere le posate prima di consumare un pasto.

Anomalie generiche

Viene anche riconosciuta un'anomalia generica, ovvero che non dipende dalla specifica attività effettuata: *Sportello non chiuso*. In questo caso il paziente si dimentica di chiudere uno sportello (ad esempio quello dell'armadietto delle medicine o del frigorifero) per una quantità di tempo considerevole.

4.3.3 Risultati sperimentali

Il dataset acquisito contiene la simulazione del comportamento di tre pazienti: uno sano, uno con problemi cognitivi lievi e uno con problemi cognitivi più gravi. Per ogni paziente sono presenti 7 giorni di attività. Il modulo di riconoscimento di anomalie è quindi stato testato separatamente per ogni paziente utilizzando le informazioni prodotte dal riconoscimento (e raffinamento) di attività, calcolando il numero di *veri positivi*, *falsi positivi* e *falsi negativi* prodotti dall'inferenza (confrontando il risultato dell'inferenza con le annotazioni delle anomalie presenti nei dati). Può quindi succedere che attività non riconosciute producano *falsi positivi* o *falsi negativi* nel riconoscimento di anomalie. Prima di analizzare i risultati ottenuti, è necessario specificare che queste regole modellano precisamente le anomalie presenti nel dataset. Con un altro

dataset (ad esempio acquisito tramite pazienti reali) il tasso di errore potrebbe essere più elevato.

Paziente sano

Analizziamo i risultati ottenuti inferendo le anomalie sui dati acquisiti dal primo paziente, che non presenta disturbi cognitivi.

Veri Positivi	Falsi Positivi	Falsi negativi
5	4	2

Essendo il paziente in questione sano, solo 7 anomalie sono state introdotte. Il sistema è riuscito ad identificarne correttamente 5, inferendo tuttavia anche 4 anomalie che in realtà non sono avvenute. Questo è dovuto al fatto che delle attività non sono state precedentemente riconosciute, ottenendo in questo modo mancanza di informazione.

Paziente affetto da deterioramento cognitivo lieve

Vengono ora presentati i risultati ottenuti con il paziente che presenta deterioramento cognitivo lieve.

Veri Positivi	Falsi Positivi	Falsi negativi
32	0	0

Essendo le regole scritte appositamente per catturare le anomalie presenti nei dati, viene ottenuto un riconoscimento totale.

Paziente affetto da deterioramento cognitivo grave

Analizziamo infine l'ultimo paziente, che presenta deterioramento cognitivo grave.
Anche in questo caso otteniamo dei risultati molto buoni, per lo stesso motivo riportato nel caso precedente. Gli unici falsi positivi ottenuti sono dovuti ad errori di predizione da parte del modulo di riconoscimento di attività.

Veri Positivi	Falsi Positivi	Falsi negativi
25	2	0

4.4 Software realizzato

In questa sezione viene fornita una spiegazione ad alto livello del software prodotto in questo lavoro di tesi. I dati che vengono prodotti dall'ambiente sensorizzato vengono rilevati e convertiti per essere forniti in input ad una *Markov Logic Network* (con i pesi già appresi in una fase precedente) implementata utilizzando *thebeast*. Le attività riconosciute vengono successivamente raffinate per tentare di correggere eventuali errori di predizione. L'output del raffinamento di attività viene quindi convertito in un formato per *tuProlog*, in modo da effettuare l'inferenza logica necessaria a dedurre eventuali anomalie. In questa fase vengono anche utilizzate informazioni specifiche sul paziente monitorato (ad esempio le medicine prescritte dai medici). L'architettura appena descritta viene integrata all'interno di un'applicazione *Android* utilizzata nelle abitazioni dei pazienti, in modo da poter monitorare ed analizzare il loro comportamento ed inviare le informazioni prodotte ad un servizio di telemedicina.

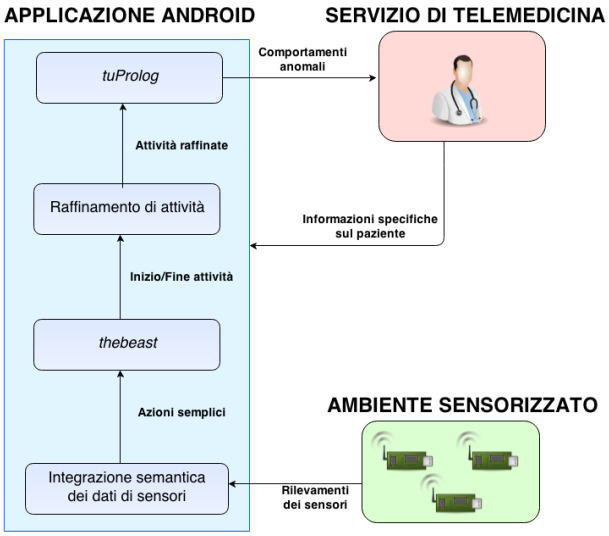

Figura 4.2: Architettura del software

4.4.1 Integrazione semantica dei dati di sensori

Le rilevazioni dei sensori sono dati grezzi che devono essere preprocessati e trasformati prima di essere effettivamente utilizzati dai moduli di riconoscimento. Per migliorare la riusabilità del codice le misurazioni dei sensori vengono inizialmente convertite in un formato intermedio, in modo tale da poter realizzare moduli in grado di produrre l'input per uno specifico modulo di riconoscimento senza dover dipendere direttamente dal formato utilizzato dalla sorgente dei dati (che differisce in base alla strumentazione sensoristica utilizzata). Nel prototipo realizzato viene implementato il parsing delle informazioni prodotte dalla rete di sensori estrapolando i rilevamenti acquisiti e i relativi timestamp (mantenendo l'informazione sull'ordinamento temporale, indispensabile per generare il predicato $prossimoEvento$ discusso nel capitolo precedente). Sono stati quindi realizzati dei moduli in *Java* per generare i file di input per *thebeast* ed *Alchemy* partendo dal formato intermedio (anche se viene effettivamente utilizzato solamente *thebeast* in questa fase prototipale). Il vantaggio di usare questo approccio è che al variare della sorgente dei dati è necessario solamente implementare una nuova conversione per il formato intermedio, mentre il codice per generare i file di input per i moduli di riconoscimento non varia.

Questo sistema di conversione multi-livello è stato utilizzato anche in una precedente fase di costruzione e validazione dei modelli di riconoscimento per trasformare il dataset acquisito in file di input per gli strumenti che implementano le *MLN*. È quindi in grado di gestire l'informazione riguardo alle annotazioni, applicando le seguenti regole:

- Ogni annotazione di *inizio di attività* ha come istante temporale quello associato alla prima misurazione di sensore successiva.
- Ogni annotazione di *fine di attività* ha come istante temporale quello associato

alla prima misurazione di sensore precedente.

- Ogni annotazione di *anomalia* ha come istante temporale quello associato alla prima misurazione di sensore precedente.

4.4.2 Riconoscimento e raffinamento di attività

Il riconoscimento di attività viene effettuato da *thebeast* utilizzando l'informazione acquisita nella fase precedente utilizzando una *MLN* che abbia ovviamente già appreso i pesi delle regole. Il risultato è un file contenente inizio e fine di attività dedotti. Per raffinare il riconoscimento effettuato viene utilizzato l'algoritmo spiegato in 3.2.2, che tenta di correggere il problema dovuto alla presenza di possibili *falsi positivi* e *falsi negativi*. A tale scopo, le misurazioni di sensore e le informazioni dedotte su inizio e fine di attività vengono ordinate in una lista in ordine temporale crescente. Questa lista viene letta sequenzialmente da una *macchina a stati finiti* composta da tre stati:

- Stato *nessun riconoscimento avvenuto*
- Stato *riconosciuto un inizio*
- Stato *riconosciuto un inizio e una fine*

In base allo stato corrente, l'algoritmo tratta in maniera differente l'elemento della lista considerato. Viene mantenuto in memoria un *buffer* (inizialmente vuoto) per contenere temporaneamente le misurazioni di sensore da associare alle attività che vengono riconosciute. L'output di questo algoritmo è una lista di attività formate da nome, istante di inizio, istante di fine e misurazioni di sensori coinvolte. Viene ora descritto in dettaglio il comportamento per ogni stato dell'algoritmo.

Stato *nessun riconoscimento avvenuto*

Inizialmente, la macchina a stati finiti si trova nello stato nel quale nulla è stato ancora riconosciuto. Analizziamo i tre casi possibili:

- Se l'elemento letto è una misurazione di sensore, viene aggiunta al *buffer*.
- Se l'elemento letto è l'informazione su un inizio di attività, lo stato diventa *riconosciuto un inizio*.
- Se l'elemento letto è una fine di attività, si assume che l'inizio della stessa sia un *falso negativo*. Viene quindi creato un inizio utilizzando come istante temporale quello della misurazione di sensore alla cima del *buffer* e lo stato diventa *riconosciuto un inizio e una fine*.

Stato *riconosciuto un inizio*

Se l'algoritmo si trova in questo stato, vuol dire che è stato riconosciuto l'inizio di una specifica attività.

- Se l'elemento letto è una misurazione di sensore, viene aggiunta al *buffer*.
- Se l'elemento letto è l'informazione su un inizio della stessa attività, questo viene assunto come *falso positivo* ed ignorato.
- Se l'elemento letto è l'informazione su un inizio di un'attività differente, vuol dire che la fine della prima si assume che sia un *falso negativo*. Viene quindi creata una fine di attività usando come istante temporale quello dell'ultima misurazione di sensore presente nel *buffer* e riconosciuta un'attività (le misurazioni di sensore associate sono quelle presenti nel *buffer*). Il *buffer* viene svuotato, e lo stato rimane lo stesso: l'inizio di attività ora riconosciuto è quello corrispondente all'elemento letto.

- Se l'elemento letto è l'informazione sulla fine della stessa attività, lo stato diventa *riconosciuto un inizio e una fine*.

- Se l'elemento letto è la fine di un'altra attività, si assume che la fine della prima e l'inizio della seconda siano *falsi negativi*. In questo caso, però, non è possibile ricostruire le attività. Il *buffer* viene svuotato e lo stato diventa *nessun riconoscimento avvenuto*.

Stato *riconosciuto un inizio e una fine*

In questo stato sono stati riconosciuti l'inizio e la fine di una particolare attività.

- Se l'elemento letto è una misurazione di sensore, viene aggiunta al *buffer*.

- Se l'elemento letto è l'inizio di una nuova attività, l'attività corrente viene riconosciuta (le misurazioni di sensore abbinate sono quelle del *buffer* con istante temporale compreso tra l'inizio e la fine dell'attività). Il *buffer* viene svuotato e lo stato diventa *riconosciuto un inizio*.

- Se l'elemento letto è la fine di una nuova attivitá, vuol dire che l'inizio della stessa è un *falso negativo*. Viene quindi riconosciuta l'attività corrente utilizzando le misurazioni di sensore nel *buffer* comprese tra l'inizio e la fine, che vengono successivamente rimosse. Infine, viene creato un inizio per la nuova attività utilizzando come istante temporale quello della misurazione di sensore in cima al *buffer*.

4.4.3 Riconoscimento di anomalie

Le attività prodotte dall'algoritmo di raffinamento vengono codificate per *tuProlog*, utilizzando il modello descritto in 3.3.1. Per poter riconoscere gran parte delle anomalie, vengono utilizzate delle informazioni specifiche sul paziente monitorato e sull'am-

biente circostante. Se consideriamo ad esempio le anomalie riguardanti l'assunzione delle medicine, è necessario avere l'informazione su quali medicine siano prescritte dai medici, in che giorni e in che fasce orarie della giornata. Viene quindi eseguita l'inferenza con *tuProlog* per ottenere le informazioni sulle eventuali anomalie svolte dal paziente. L'output di questa fase è una lista delle anomalie riscontrate.

4.4.4 Applicazione *Android*

I moduli presentati in questa sezione vengono integrati in un'applicazione *Android* sviluppata dal partner *3Caravelle*. Questa applicazione è installata su un dispositivo mobile situato nell'ambiente domestico del paziente che si intende monitorare, occupandosi di ricevere le informazioni dai sensori, elaborarle e produrre informazioni che vengono inviate ai medici. La figura 4.3 mostra la schermata principale.

Figura 4.3: Screenshot dell'applicazione

Conclusioni e sviluppi futuri

Questo lavoro di tesi si è concentrato sullo studio di una tecnica ibrida per il riconoscimento automatico di attività umane e di comportamenti anomali, sulla sua implementazione prototipale e sulla valutazione sperimentale della tecnica, con l'obbiettivo di realizzare un sistema di monitoraggio domiciliare di pazienti affetti da deterioramento cognitivo. Dei sensori di ambiente sono installati nell'abitazione del paziente con lo scopo di monitorarne l'interazione con gli oggetti domestici. Il modello di riconoscimento di attività considerato unisce l'approccio simbolico e quello probabilistico grazie all'ultilizzo della logica probabilistica *Markov Logic Network*. La tecnica considerata consiste nell'analizzare finestre di misurazioni di sensori consecutive per individurare l'inizio e la fine delle attività svolte. Il modello di riconoscimento di anomalie utilizza invece un sistema a regole della logica del primo ordine individuando, tramite opportuni ragionamenti di tipo temporale, specifici tipi di anomalie presenti nelle attività riconosciute. I punti di forza di questa metodologia sono:

1. La tecnologia sensoristica non è invasiva e viene installata direttamente nell'abitazione del paziente.

2. Il modulo di riconsocimento di attività riesce a gestire l'incertezza intrinseca delle misurazioni dei sensori.

3. Il modulo di riconoscimento di anomalie riesce a produrre una diagnostica precisa sul tipo di anomalia effettuato (ad esempio, quale specifica medicina è stata

dimenticata).

I risultati ottenuti da questa sperimentazione sono promettenti: le attività e le anomalie vengono riconosciute nella maggior parte dei casi. Diversi possibili miglioramenti possono essere realizzati in lavori futuri. Un primo miglioramento potrebbe essere quello di trattare l'incertezza anche nel modulo di riconoscimento di anomalie utilizzando una tecnica statistica (come ad esempio un'ontologia probabilistica). Sarebbe anche interessante aggiungere un livello di confidenza sull'inferenza prodotta da entrambi i moduli di riconoscimento, calcolando le probabilità marginali invece di derivare il mondo più probabile tramite un'inferenza di tipo *MAP*. In questo modo si riuscirebbe ad avere un sistema di riconoscimento più sofisticato, in grado di tenere conto dei valori di probabilità di ogni possibile attività o anomalia svolta. Un'altra miglioria potrebbe essere tenere conto del fatto che le attività non siano svolte per forza in stretta sequenza, ma che invece possano anche essere svolte parallelamente (come nel lavoro presentato in [12]). Infine sarebbe interessante ampliare la sperimentazione della metodologia qui presentata utilizzando differenti dataset acquisiti con le seguenti migliorie:

- Monitorando il comportamento di pazienti reali.
- Utilizzando sorgenti di dati più ricche di contesto.
- Incrementando i tipi di attività da riconoscere.

Bibliografia

[1] Parisa Rashidi, Alex Mihailidis. *A Survey on Ambient-Assisted Living Tools for Older Adults*. IEEE Journal of biomedical and health informatics, 2013.

[2] B. Kaluza, M. Gams. *An Approach to Analysis of Daily Living Dynamics*. Proceedings of the World Congress on Engineering and Computer Science, 2010.

[3] L. Chen, I. Khalil. *Activity Recognition: Approaches, Practices and Trends*. Activity Recognition in Pervasive Intelligent Enviroments, Atlantis Ambient and Pervasive Intelligence, Volume 4, Atlantis Press, 2011.

[4] Juan Yea, Simon Dobsona, Susan McKeeverb. *Situation identification techniques in pervasive computing: A review*. Pervasive and Mobile Computing, 2008.

[5] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, Ben Krose. *Accurate activity recognition in a home setting*. UbiComp, 2008.

[6] D. Vail, M. Veloso, J. Lafferty. *Conditional Random Fields for Activity Recognition*. Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, 2007.

[7] P. Dawadi , D. Cook, M. Schmitter-Edgecombe, C. Parsey. *Automated assessment of cognitive health using smart home technologies*. Technology and Health Care, IOS Press, 2013.

[8] Liming Chen, Chris D. Nugent, Hui Wang. *A Knowledge-Driven Approach to Activity Recognition in Smart Homes*. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2012

[9] Daniele Riboni, Claudio Bettini. *OWL 2 modeling and reasoning with complex human activities*. Pervasive and Mobile Computing, 2011.

[10] Alexander Artikis, Marek Sergot, Georgios Paliouras. *A logic programming approach to activity recognition*. EiMM '10 Proceedings of the 2nd ACM international workshop on Events in multimedia, 2010.

[11] Matthew Richardson, Pedro Domingos. *Markov Logic Networks*. Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2006.

[12] R. Helaoui, M. Niepert, H. Stuckenschmidt. *Recognizing interleaved and concurrent activities using qualitative and quantitative temporal relationships*. Pervasive and Mobile Computing 7 (6), 660-670.

[13] Daniele Riboni, Claudio Bettini. *Context-aware Activity Recognition through a Combination of Ontological and Statistical Reasoning*. UIC '09 Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing, 2009.

[14] Oluwatoyin P. Popoola, Kejun Wang. *Video-Based Abnormal Human Behavior RecognitionA Review*. Systems, Man, and Cybernetics, 2012.

[15] Vikramaditya R. Jakkula, Diane J. Cook. *Detecting Anomalous Sensor Events in Smart Home Data for Enhancing the Living Experience*. Artificial Intelligence and Smarter Living, 2011.

[16] G. Virone, A. Sixsmith. *Activity Prediction for In-borne Activity Monitoring*. Intelligent Environments, 2008.

[17] V. Jakkula, D.J. Cook, A.S. Crandall. *Temporal pattern discovery for anomaly detection in a smart home*. Intelligent Environments, 2007.

[18] Allen JF, Ferguson G. *Actions and events in interval temporal logic*. Journal of Logic and Computation, 1994.

[19] Tamara L. Hayesa, Francena Abendrothb,Andre Adamid, Misha Pavela, Tracy A. Zitzelbergerb, Jeffrey A. Kaye. *Unobtrusive assessment of activity patterns associated with mild cognitive impairment*. Alzheimer's Dementia, 2008.

[20] Laura Serra, Carlo Caltagirone. *Mild Cognitive Impairment ovvero la fase preclinica della demenza*. La neurologia italiana, Roma, 2008.

[21] Daphne Koller, Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Massachusetts, 2009.

[22] Silvio Ghilardi. *Risoluzione e Sovrapposizione*. Dipartimento di Informatica, Università degli Studi di Milano, 2007.

[23] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, and P. Domingos. *The Alchemy system for statistical relational AI*. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2007.

[24] Parag Singla, Pedro Domingos. *Discriminative Training of Markov Logic Networks*. Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2005.

[25] Stanley Kok, Pedro Domingos. *Learning the Structure of Markov Logic Networks*. Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2005.

[26] Sebastian Riedel. *Improving the Accuracy and Efficiency of MAP Inference for Markov Logic*. Institute for Collaborating and Communicating Systems, School of Informatics, University of Edinburgh, 2008.

[27] Feng Niu, Christopher R, AnHai Doan, Jude Shavlik. *Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS*. Proceedings of the VLDB Endowmen, 2011.

[28] Jan Noessner, Mathias Niepert, Heiner Stuckenschmidt. *RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models*. AAAI'13, 2013.

[29] F. Furlan, G.A. Lanzarone. *PROLOG: Linguaggio e metodologia di programmazione logica*. Dipartimento di Informatica, Università degli Studi di Milano.

[30] Enrico Denti, Andrea Omicini, Alessandro Ricci. *tuProlog: A Light-weight Prolog for Internet Applications and Infrastructures Practical Aspects of Declarative Languages*. Lecture Notes in Computer Science 1990, 2001.