ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Scuola di Ingegneria e Architettura
Corso di Laurea in Ingegneria e Scienze Informatiche

SISTEMI INTELLIGENTI A SUPPORTO DELLA GESTIONE DEI TRAUMI: UN CASO DI STUDIO

Elaborata nel corso di: Programmazione concorrente e distribuita

Tesi di Laurea di:
ENRICO CAGNAZZO

Relatore:
Prof. ALESSANDRO RICCI

Correlatore:
Dott.ssa SARA MONTAGNA

ANNO ACCADEMICO 2017–2018
SESSIONE I

PAROLE CHIAVE

Trauma Tracker

Metrica

Long Short-Term Memory

Recurrent Neural Network

eHealth

”Il testo che avrei voluto scrivere
Non è di certo questo
Perciò dovrò continuare a scrivere
Perché di certo riesco
Prima o poi”
Caparezza

Indice

Introduzione **ix**

1 Background **1**
1.1 Confronto tra pazienti 3
1.2 Predizione di serie temporali 4

2 Trauma Tracker **7**
2.1 Architettura 8
2.2 Funzionalità 9
2.2.1 Tracking 9
2.2.2 Assistance 10
2.3 Uso di Trauma Tracker 11

3 Metriche per l'identificazione di pazienti simili **13**
3.1 Metriche 14
3.1.1 Normalizzazione 15
3.1.2 Metrica euclidea 16
3.1.3 Metrica unificata 17
3.2 Pesi 19
3.3 Algoritmo sviluppato 20
3.4 Validazione dei risultati 21
3.4.1 Metrica euclidea 23
3.4.2 Metrica unificata 24
3.4.3 Commenti generali 25

4 Previsione dello stato di shock **27**
4.1 Reti neurali ricorrenti 27
4.1.1 LSTM-RNN 29

4.2 LSTM-RNN in Trauma Tracker 31
4.3 Implementazione della rete 32
4.3.1 Struttura della rete 32
4.3.2 Pre-processing 34
4.3.3 Input 35
4.3.4 Output 36
4.3.5 Allenamento 37
4.4 Validazione dei risultati 38

5 Architettura dei sotto sistemi implementati 41
5.1 Confronto pazienti simili 41
5.2 Previsione dello stato di shock 44

6 Conclusioni e possibili sviluppi 47
6.1 Calcolo di pazienti simili 47
6.2 Previsione dello stato di shock 48
6.3 Commenti generali 49

A Materiale informatico 51

Introduzione

L'obiettivo del lavoro svolto in questa tesi è quello di sperimentare l'adozione di tecniche e algoritmi dell'Intelligenza Artificiale per supportare le operazioni dei medici del Trauma Center dell'ospedale Maurizio Bufalini di Cesena durante le prime fasi del trattamento del paziente con trauma grave.

La tesi si colloca all'interno di un più ampio lavoro che ha l'obiettivo di integrare *Trauma Tracker* – un sistema sviluppato presso l'università Alma Mater Studiorium di Bologna, in collaborazione con i medici e gli infermieri dell'ospedale Maurizio Bufalini di Cesena, per il tracciamento del trauma e attualmente in uso nel centro traumatologico del suddetto ospedale - con un sistema per l'analisi real-time dei dati raccolti sul paziente in cura, al fine di fornire al medico suggerimenti e allarmi ritenuti utili per un processo di cura più rapido ed efficace. Il sistema *Trauma Tracker* verrà descritto nel capitolo 2.

Allo scopo in questa tesi è stato sviluppato un sistema informatico composto da due parti principali: la prima è responsabile di individuare dei pazienti simili all'ultimo degente arrivato presso il centro e suggerire ai medici un piano d'azione mentre la seconda cerca di predire se il paziente sotto trattamento andrà in stato di shock.

La prima parte sarà di aiuto all'inizio del trattamento, quando i medici dovranno scegliere cosa fare e potranno vedere i risultati ottenuti dalle varie operazioni effettuate su pazienti simili all'attuale.

Questo sottosistema, basandosi sulle informazioni riguardanti lo stato iniziale dei vari degenti in cura presso il centro traumatologico (incluso quello appena arrivato), cerca di trovare dei pazienti simili e per mostrare la loro risposta alle cure e il rispettivo andamento clinico.

La seconda parte invece sarà utile durante il trattamento per poter suggerire ai medici quali azioni intraprendere in base alla predizione

che il paziente vada in stato di shock. Si pensi, ad esempio, alle possibili conseguenze subite da un paziente, sottoposto ad un esame TAC; infatti durante questo esame sarebbe impossibile per i medici agire immediatamente in caso di complicazioni.

Per effettuare queste predizioni il sistema utilizza quasi tutte le informazioni presenti nei report generati da *Trauma Tracker* (descritti nella sezione 2.2.1). Più precisamente le informazioni dei report che sono state utilizzate da questo sottosistema sono lo stato iniziale del paziente e l'elenco delle azioni effettuate durante il trattamento.

Questa parte del sistema sviluppato risponderà in tempo reale all'inserimento di nuovi eventi relativi al trattamento del paziente. Inoltre sfruttando le informazioni relative ai pazienti precedenti inseriti nel sistema *Trauma Tracker* cercherà di predire se il paziente possa andare o meno in stato di shock.

Questi due sottosistemi sono stati realizzati e, sebbene non siano ancora completamente integrati con *Trauma Tracker*, hanno prodotto dei buoni risultati.

Le validazioni dei due sottosistemi sono molto incoraggianti e fanno pensare che con un numero maggiore di dati e qualche piccola modifica si potrebbe riuscire a fornire al centro traumatologico dell'ospedale Bufalini di Cesena degli strumenti utili durante la prima fase di rianimazione dei pazienti presso il centro traumatologico.

Capitolo 1

Background

Il primo intervento a cui è sottoposto un paziente che ha subito un trauma è determinante al fine di aumentare le possibilità di successo del trattamento effettuato sul paziente stesso. In diversi articoli è stata sottolineata l'importanza della "golden hour" [2, 6] ovvero delle operazioni effettuate sul paziente durante i primi 60 minuti a partire dal momento dell'incidente e di come queste abbiano una grande influenza sull'esito globale del trattamento.

Questo termine generalmente sottolinea come la mortalità e la morbidità possano aumentare se non vengono prestate le cure necessarie nella prima ora dopo l'incidente.

Durante questa fase l'equipe medica è quindi sottoposta ad un lavoro cruciale e a dei ritmi frenetici che le lasciano poco tempo per decidere quali azioni intraprendere.

Si pensa che, al fine di aiutare i medici durante il loro intervento, possa essere utile un sistema informatico che agevoli tutta l'equipe medica senza essere d'intralcio nel loro lavoro. Per sviluppare questo sistema al meglio si possono sfruttare i vantaggi ottenuti dall'utilizzo dell'informatica negli ospedali e in tutto il mondo sanitario a 360°.

Uno di questi vantaggi è rappresentato dagli "electronic medical record" (EMR) cioè delle strutture dati con all'interno le informazioni mediche di un paziente.

Anche l'utilizzo di dispositivi *wearable* ha permesso di ideare soluzioni alternative poiché permettono di creare interfacce uomo-computer

che non siano d'intralcio durante le operazioni che la persona deve svolgere.

L'università Alma Mater Studiorum di Bologna, in collaborazione con l'ospedale Maurizio Bufalini di Cesena, ha sviluppato il sistema Trauma Tracker con l'intento di supportare medici ed infermieri durante le prime fasi della rianimazione di un paziente che ha subito un trauma (il sistema è descritto nel capitolo 2).

Tra i vari modi possibili in cui progettare il sistema è stato scelto il paradigma ad agenti. Quest'ultimo appare il più idoneo nell'ambito dell'eHealth perché permette di migliorare le prestazioni del sistema sanitario tramite l'infrastruttura informatica.

Per agente si intende un'entità autonoma in grado di percepire l'ambiente (virtuale o reale) che lo circonda e agire sia in maniera reattiva (rispondere ad uno stimolo) che proattiva (iniziare un'azione in maniera autonoma). Se l'agente è inserito in un contesto fisico è presente, oltre alla componente software, anche una hardware per permettergli di poter interagire con il mondo che lo circonda.

Il paradigma ad agenti permette, quindi, di costruire dei sistemi multiagente in cui ogni servizio è svolto da un singolo agente e tutti quanti comunicano in maniera proattiva tra di loro, mediante un'opportuna organizzazione, interagendo anche con l'ambiente che li circonda.

Secondo [11] un sistema ad agenti può fornire i seguenti vantaggi ad un'organizzazione sanitaria:

1. gestione dei dati medici: gestire l'accesso, l'integrazione e la condivisione dei dati dei pazienti tramite più sorgenti remote per poter facilitare il lavoro dei dottori oltre che per scopi statistici;

2. sistema di supporto alle decisioni: supportare l'equipe medica sulle scelte da effettuare;

3. pianificazione e allocazione delle risorse: l'allocazione dei medici e dei vari professionisti deve essere coerente con le principali tecniche di pianificazione in modo da ottimizzare le risorse;

4. trattamento remoto: assistere a distanza i pazienti in modo tale che questi non siano obbligati a spostarsi permettendo quindi, ai dottori, di monitorare a distanza lo stato di salute dei pazienti.

Tra tutti i reparti ospedalieri soprattutto quelli d'emergenza richiedono un'alta reattività, una risposta veloce e coordinata e una capacità di prendere decisioni in maniera accurata e rapida.

In questi scenari, dunque, la tecnologia ad agenti può aiutare sotto più punti di vista [5]:

- suggerire la migliore soluzione in maniera proattiva agli operatori umani che hanno poco tempo per decidere quale possa essere la migliore cura a causa del susseguirsi di eventi in maniera frenetica;

- creare una documentazione accurata della rianimazione del paziente, in modo da poter migliorare la qualità delle cure. Riportando [3] "La qualità delle cure del trauma può essere definita come ottenere il miglior risultato per un determinato insieme di circostanze cliniche";

- aumentare la coordinazione tra i vari partecipanti alla gestione del trauma.

1.1 Confronto tra pazienti

Si è pensato, in maniera congiunta con i membri del centro traumatologico dell'ospedale Bufalini di Cesena, che ottenere dei report relativi a pazienti simili a quello in cura possa aiutare i medici nelle prime fasi della rianimazione.

Per questo motivi sono stati ricercati in letteratura degli studi in cui si affrontasse il problema della comparazione dei segni vitali di due pazienti generici. Poiché non sono stati trovati studi di questo tipo si è effettuata una seconda ricerca, di spettro più ampio, spostando il focus ad un confronto generico tra insiemi di dati.

Confrontare lo stato iniziale di due pazienti è risultato essere un problema più complesso di quanto non ci si aspettasse poiché in pochi studi sono stati sviluppati algoritmi in grado di calcolare la similarità tra dataset non omogenei.

Ad aumentare la complessità di questo confronto è stato il fatto che non tutti i report avessero le informazioni relative a tutti i cam-

pi. Infatti sono presenti diversi report in cui lo stato iniziale non è compilato in maniera esaustiva.

Per effettuare questa comparazione sono state valutate più tecniche: una si basava del confronto tra i singoli campi e un'altra confrontava l'intero dataset dello stato iniziale.

Per confrontare i singoli campi sono state analizzate varie metriche che fossero in grado di valutare la distanza tra insiemi di valori numerici, categorici e misti[15, 20, 8, 18].

Si è pensato anche di confrontare i dataset degli stati iniziali nella loro interezza mediante l'algoritmo NCD (*Normalized Compress Distance*)[4].

La scelta finale è stata quella di confrontare i singoli campi in modo tale da aver un maggior controllo su di essi potendo inserire eventuali pesi e gestire in maniera più precisa i campi assenti nel report.

1.2 Predizione di serie temporali

Per poter predire lo stato di shock di un paziente è necessario analizzare tutta la sequenza di eventi relativi al suo trattamento. Quando si esaminano queste sequenze è importante non dimenticare l'ordine temporale con i vari eventi si susseguono. Per questo motivo queste sequenze vengono dette *serie temporali*.

In letteratura sono stati trovati pochi studi che abbiano analizzato e implementato sistemi atti a predire serie temporali in ambito medico.

Uno di questi studi è stato presentato in [17] e dimostra l'efficacia di una rete neurale ricorrente nel riconoscere un'apnea del sonno utilizzando solo le informazioni relative al battito cardiaco.

Tale studio, tuttavia, sfrutta le reti neurali per classificare delle sequenze di dati in varie categorie, ma il problema analizzato non risulta essere affine a quello che si vuole risolvere in questa tesi.

Un altro studio che cerca di predire delle valutazioni mediche basandosi su serie temporali è presentato in [19]. In questo caso si cerca di predire la qualità del sonno utilizzando i dati ottenuti mediante dei particolari dispositivi *wearable* usati per studiare i pattern dell'attività fisica.

I dati ottenuti mentre il paziente era sveglio sono stati utilizzati per predire la qualità del sonno. I dati acquisiti quando il paziente dormiva sono stati confrontati con la predizione effettuata per addestrare e validare i risultati della rete.

Tuttavia nell'analisi della letteratura si è notata l'assenza di una base solida di studi nei quali le serie temporali vengono calcolate (e predette) in base a dati eterogenei e campionati in maniera irregolare. Per questo motivo è stata necessaria un'analisi approfondita della letteratura per trovare almeno uno studio simile a quello svolto in questa tesi.

Lo studio presentato in [1] ha trattato una problematica molto più simile a quella affrontata durante questo lavoro poiché ha fornito ad un ospedale pediatrico un sistema basato su reti neurali ricorrenti in grado di predire il rischio di mortalità di un paziente durante la sua permanenza nel reparto di terapia intensiva.

In questo studio sono stati utilizzati come input i parametri vitali, i farmaci somministrati, le procedure effettuate e i risultati delle prove di laboratorio.

Ogni input veniva quindi immesso nel sistema solo quando variava un componente dell'input sopra descritto. Di conseguenza bisogna sottolineare che in questo studio gli input non hanno una cadenza fissa ma il tempo tra uno e l'altro potrebbe essere differente.

Capitolo 2

Trauma Tracker

Al giorno d'oggi le tecnologie di *wearable computing* e di *eyewear computing* hanno raggiunto un livello di maturità tale da poter essere adottate anche in contesti reali permettendo di progettare sistemi innovativi. Ad esempio gli smart glass consentono la creazione di interfacce uomo-macchina in cui non è necessario l'uso delle mani o distogliere lo sguardo da ciò che si sta facendo per guardare un monitor.

Lo sviluppo di queste tecnologie permette l'ideazione di nuovi tipi di *software personal agents* poiché consente di collaborare con i membri dell'equipe medica senza ostacolarli nel loro lavoro.

Trauma Tracker si appoggia proprio su tecnologie di questo tipo (tablet e smart glass) al fine di creare un assistente medico digitale che faciliti l'operato di tutta l'equipe medica durante il suo lavoro nel trattamento di un paziente appena arrivato al centro. Al momento l'assistenza è limitata alla creazione del report dell'intervento e alla generazione di warning nel caso in cui le varie procedure non siano effettuate correttamente.

Trauma Tracker è un sistema informatico già installato ed operativo presso il centro traumatologico dell'ospedale Bufalini di Cesena, centro che si occupa del trattamento di pazienti che hanno subito un trauma di qualsiasi tipo e che non possono essere trattati in altri reparti data la gravità del trauma stesso.

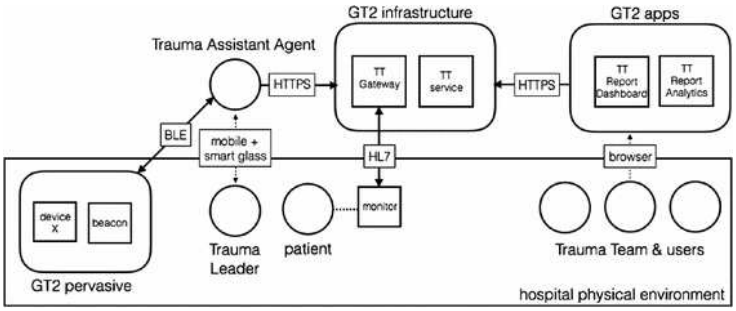

Figura 2.1: Architettura del sistema Trauma Tracker attualmente in uso; da [5]

2.1 Architettura

Trauma Tracker è stato sviluppato seguendo il paradigma ad agenti. Questi sono immersi in un ambiente virtuale che viene descritto di seguito.

L'architettura generale del sistema, rappresentata nella figura 2.1, è del tipo agent-oriented e service-oriented ed include quattro parti principali:

- il *Personal Medical Digital Assistant Agents* (PDMA): presente in figura con il nome di *Trauma Assistant Agent*;
- un insieme di servizi Web-based sviluppati nella rete LAN dell'ospedale riportati come GT^2 *infrastructure*;
- un insieme di servizi pervasivi forniti da dispositivi distribuiti nell'ambiente fisico dell'ospedale, riportati in figura come GT^2 *pervasive*;
- un insieme di web-app, riportate come GT^2 *apps*, che permettono agli utenti di accedere ed interagire con alcuni servizi della GT^2 *infrastructure*.

Il *Trauma Assistant Agent* è eseguito sia su dispositivi mobili (tablet) che su dispositivi indossabili (smart-glasses) utilizzati dal team leader dell'equipe medica (*Trauma Leader*). L'agente interagisce con i servizi di GT^2 *infrastructure* e di GT^2 *pervasive*; mentre la GT^2

infrastructure include un insieme di servizi web sfruttati dal *Trauma Assistant Agent* e dalle web-app. La GT^2 *pervasive* include, al momento, un insieme di beacon piazzati nelle varie stanze coinvolte nella gestione del trauma al fine di poter localizzare la stanza in cui è presente il paziente.

2.2 Funzionalità

Attualmente *Trauma Tracker* svolge due funzionalità chiave:

- *tracking*: tenere traccia di ciò che avviene durante il trattamento del paziente in modo tale da avere un'accurata documentazione del trauma, automatizzare la creazione e gestione del report e permettere l'analisi dei dati in un secondo momento;

- *assistance*: assistere in real time l'operato dell'equipe medica generando automaticamente avvisi e suggerimenti in base agli eventi registrati durante l'operazione.

2.2.1 Tracking

La funzionalità di tracking e di creazione automatica dei report è stata inserita all'interno di *Trauma Tracker* sia per diminuire il carico delle operazioni che l'equipe medica deve svolgere durante il trattamento del trauma che per migliorare la qualità dei report finali.

Ciò avviene perché durante la fase di rianimazione del paziente si susseguono rapidamente molte operazioni e il soggetto incaricato di creare il report, avendo anche altri compiti da svolgere durante il trattamento del paziente, potrebbe non essere in grado di produrre un lavoro preciso tanto quanto un sistema informatico appositamente creato.

I report sono dei documenti che contengono all'interno varie informazioni tra cui quali membri dell'equipe medica sono coinvolti, chi è il *Trauma Leader*, la data di inizio e fine dell'intervento, l'*Injury Severity Score* (ISS, indicatore della gravità del trauma) per ogni regione del corpo, le informazione anagrafiche del paziente quali età e genere, i segni vitali del paziente nel momento in cui arriva presso il centro

traumatologico e l'insieme di eventi, in ordine cronologico, che si sono succeduti durante la rianimazione del paziente.

All'interno dei segni vitali sono presenti informazioni riguardanti i parametri vitali, le vie aeree e la respirazione, l'esame neurologico, la valutazione dei polsi periferici, la presenza di eventuali accessi vascolari, informazioni su fratture ossee e/o ustioni (se presenti).

Gli eventi invece servono per descrivere meglio il corso delle operazioni e cosa avviene durante il trattamento del paziente. Per questo motivo vengono registrati, tra gli altri, i seguenti tipi di evento: procedura o manovra effettuata, esame clinico svolto, farmaci somministrati (in maniera continua o singola), acquisizione dei parametri vitali da monitor, variazione di altri parametri vitali, acquisizione di documentazione multimediale (foto, video, audio o testo) e cambio di stanza.

Infine le informazioni riguardanti gli eventi riportano il numero progressivo, la data e l'ora, il luogo e il tipo di evento; in base alla tipologia dell'evento riportato sono presenti ulteriori campi necessari a descrivere meglio l'accaduto (es: risultati di un esame diagnostico, la quantità di farmaco somministrato, i valori di un parametro vitale in quel momento e così via).

Durante la rianimazione del paziente il *Trauma Leader* può anche decidere di inserire foto, video e registrazioni audio da includere nel report fatte tramite gli smart-glasses che indossa. Queste azioni vengono riportate nel report come eventi ed hanno il rispettivo tipo di evento.

2.2.2 Assistance

Sfruttando il livello di tracking è stato costruito il livello di assistenza che in real-time colleziona dati e produce degli avvisi per il *Trauma Leader*.

Le regole utilizzate per generare i messaggi di avviso si basano sulle direttive suggerite dal Trauma Team che sono codificate all'interno del programma.

Sfruttando il modello BDI (*Beliefs, Desires, Intentions*) del paradigma ad agenti è stato possibile aggiungere, al livello di tracking, quello di assistenza in maniera naturale e senza utilizzare forzature.

Tabella 2.1: Esempio delle regole dell'assistenza di *Trauma Tracker*

Rule no.	Condition	Warning
1	Zero Negative Blood administered at the time t but at the time t + 5min not administered fibrinogen and tranexamic acid yet.	Message: "Administer Fibrinogen and Tranexamic Acid" for 10 secs
2	When administered 3 unit of zero negative blood and tranexamic acid and fibrinogen	Message: "Activate Massive Transfusion Protocol (MTP)?"

Infatti ogni volta che un evento viene registrato nel livello di tracking vengono aggiornati i *Beliefs* dell'agente (la conoscenza che ha del mondo che lo circonda) ed eventualmente vengono aggiunti dei piani o dei timeout nel caso in cui all'evento appena registrato siano associate delle regole.

Queste regole possono essere dipendenti o meno dal tempo e servono a generare degli avvertimenti per il *Trauma Leader* al fine di suggerire dei compiti da svolgere per rispettare determinate prassi. Due esempi di queste regole sono riportate in tabella 2.1

Affinché un piano venga attuato si deve verificare che il contesto in cui la regola è descritta sia concorde con la situazione attuale del paziente (i *Beliefs* dell'agente); successivamente il piano verrà eseguito e sarà generato un nuovo messaggio d'avviso ai *Beliefs* dell'agente che sarà notificato al *Trauma Leader* oppure, in accordo con il piano eseguito, verrà rimosso un avviso dai *Beliefs* rimuovendo anche il relativo messaggio dal display.

2.3 Uso di Trauma Tracker

I dati acquisiti tramite il sistema *Trauma Tracker* con la funzionalità di tracking possono essere riutilizzati per creare ulteriori supporti per

i medici durante le prime fasi della gestione di un paziente arrivato al centro traumatologico.

Una delle due funzionalità sviluppate in questa tesi permette di ottenere informazioni sui pazienti simili a quello appena arrivato in ospedale al fine di dare una prima indicazione ai medici circa l'effetto che hanno avuto le azioni effettuate in passato sugli altri pazienti e guidare le prime operazioni sulla base dei dati precedentemente accumulati.

L'altra funzionalità invece serve a prevedere la possibilità che il paziente sotto trattamento vada in stato di shock. Quest'informazione è utile per permettere ai medici di decidere in maniera più consapevole quali esami effettuare e come procedere nella gestione del trauma.

Queste due funzionalità sono descritte in maniera più approfondita nei capitoli seguenti.

Capitolo 3

Metriche per l'identificazione di pazienti simili

In vari studi si è notato che, nel trattamento di un paziente che ha subito un trauma, sono molto importanti le operazioni eseguite dallo staff medico nella prima ora successiva al trauma.

Per agevolare il lavoro dell'equipe durante la prima ora e dare dei possibili consigli su quale trattamento effettuare al paziente è stata implementata una nuova funzionalità di *Trauma Tracker* che, partendo da tutti i pazienti già trattati, riporta al medico quelli più simili a colui che è appena arrivato. Questo sottosistema si basa sulle differenze tra tutti i parametri che descrivono lo stato iniziale dei vari pazienti.

L'idea di avere una lista di pazienti simili ha suscitato interesse dei medici dell'ospedale Bufalini che vorrebbero provarla sul campo per testarne l'utilità. Infatti una funzionalità di questo tipo permetterebbe a medici con meno esperienza sul campo di avere un feedback sul trattamento più opportuno da eseguire al paziente.

Inoltre l'unione di database da più ospedali potrebbe permettere al *Trauma Leader* di confrontare le proprie idee con quelle di altri medici, riuscendo, ipoteticamente, ad avere nuovi incipit utili alla cura del paziente appena arrivato.

3.1 Metriche

Per calcolare la similarità tra due pazienti sono state sviluppate più metriche che si differenziano per il modo di combinare la distanza tra i campi numerici e quelli categorici.

I campi numerici comprendono sia i campi che rappresentano valori continui (es: la temperatura corporea, pressione sistolica e via dicendo) che quei campi che accettano solo un ristretto insieme di valori ma che è possibile ordinare (es: il grado di ustione, il grado di Gustilo di una frattura, ecc...).

I campi categorici comprendono sia i campi booleani [8](es: la presenza di battito in un determinato polso) che tutti quei campi in cui l'operatore deve scegliere tra una serie di possibilità che non possono essere ordinate (es: il tipo di ustione, la decompressione pleurica, ecc...).

Nel caso in cui un campo sia riempito solo in uno dei due report si è preferito non effettuare il confronto per i campi numerici ed escludere il campo stesso in quanto potrebbe portare ad eventuali errori di valutazione; per i campi categorici, invece, si assume che i campi siano diversi e quindi verrà aumentata la distanza tra i report. In questo modo si può considerare vera la seguente equivalenza

$$p_{cat} \neq q_{cat} \iff D(P,Q)_{cat} \neq 0 \tag{3.1}$$

dove P e Q si riferiscono ai report relativi a due generici pazienti e con p_{cat} e q_{cat} si considera la sottoparte di report che include solo i dati di tipo categorico mentre con $D(P,Q)_{cat}$ la distanza degli attributi categorici per i report P e Q. Questa distanza verrà poi definita in 3.1.2 e in 3.1.3

L'equivalenza corrispondente a (3.1) per la controparte numerica del report non si può assumere vera poiché, se si considerano due report uguali ad eccezione di almeno un campo numerico presente solo in un report si otterrebbe comunque $D(P,Q)_{num} = 0$. Di conseguenza nemmeno un'equazione derivata da (3.1) ma che consideri l'intero report risulta essere vera.

Tuttavia si tratta di casi particolari e quindi si è scelto di ignorare il problema per evitare di ottenere una distanza maggiore a quella reale.

Di seguito vengono descritti due metodi di normalizzazione dei campi numerici per fare in modo tale che nessuno abbia un peso maggiore rispetto all'altro.

Infine sono riportate nel dettaglio le due metriche sviluppate (*Metrica euclidea* e *Metrica unificata*) descrivendo come vengono combinati i campi numerici e quelli categorici.

3.1.1 Normalizzazione

La normalizzazione si applica ai soli campi numerici. Essa è necessaria per uniformare i valori che hanno range differenti e per evitare che campi con range più estesi possano avere un'influenza maggiore sul calcolo del risultato.

- Massimo e minimo.

  La normalizzazione che considera il massimo e il minimo (di seguito *MinMax*) risulta essere la più intuitiva in quanto si riportano tutti i valori ad un numero appartenente all'intervallo [0,1].

  Otterremo quindi il valore normalizzato relativo all'*i-esimo* campo del *P-esimo* report $x_{P,i,norm}$ mediante la seguente formula:

  $$x_{P,i,norm} = \frac{x_{P,i} - min(X_i)}{max(X_i) - min(X_i)} \tag{3.2}$$

  Dove per X_i si intende l'insieme dei valori che assume il campo i-esimo nei vari pazienti, escludendo il valore nullo.

  La distanza $d(P,Q)_i$ tra due report P e Q si può misurare attraverso la normalizzazione (3.2) come:

  $$d(P,Q)_i = x_{P,i,norm} - x_{Q,i,norm} = \frac{x_{P,i} - x_{Q,i}}{max(X_i) - min(X_i)} \tag{3.3}$$

- Media e varianza

  Un altro modo per normalizzare i valori dei campi numerici consiste nell'utilizzare la media e la varianza come proposto in [18].

In questo caso la differenza tra i due valori analizzati non è divisa per l'ampiezza dell'intervallo di appartenenza dei dati ma per la varianza dei dati stessi.

In questo caso non si ha la garanzia che i risultati siano compresi tra 0 e 1, invece viene dato maggior peso al percentile in cui si trova il valore del campo che si sta calcolando, considerando quanto questo sia anomalo in relazione con quelli degli altri pazienti.

Di conseguenza $x_{P,i,norm}$ viene calcolata mediante la seguente formula:

$$x_{P,i,norm} = \frac{x_{P,i} - E(X_i)}{Var(X_i)} \tag{3.4}$$

Dove per $E(X_i)$ si intende la media dell'insieme X_i e per $Var(X_i)$ la relativa varianza.

La formula per calcolare la distanza $d(P,Q)_i$ tramite la normalizzazione (3.4) è la seguente:

$$d(P,Q)_i = x_{P,i,norm} - x_{Q,i,norm} = \frac{x_{P,i} - x_{Q,i}}{Var(X_i)} \tag{3.5}$$

Nel seguito della tesi questa normalizzazione verrà chiamata *MeanVariance*.

3.1.2 Metrica euclidea

Per i valori categorici la metrica euclidea è stata definita come segue:

$$d(P,Q)_i = \begin{cases} 0, & \text{se } x_{P,i} = x_{Q,i}, \\ 1, & \text{se } x_{P,i} \neq x_{Q,i}. \end{cases} \tag{3.6}$$

In cui $x_{P,i} \neq x_{Q,i}$ comprende anche il caso in cui uno solo dei due campi non è presente nel report.

La distanza totale tra due report P e Q risulta essere quindi pari a:

$$D(P,Q)_{=}\frac{\sqrt{\sum_{i=1}^{N} d(P,Q)_i^2}}{N_{comp}} \tag{3.7}$$

Dove N corrisponde al numero totale di campi (categorici e numerici) e N_{comp} il numero di campi comparati ovvero la somma dei campi numerici presenti in entrambi i report e quelli categorici presenti in almeno uno dei due.

In questo caso quindi sono stati sommati indistintamente tutti i campi numerici e categorici, successivamente il risultato è stato normalizzato per il numero di campi confrontati in modo tale da non penalizzare una coppia di report per la quale sono stati esaminati più campi. Questo accadrebbe poiché $d(P,Q)_i^2 \geq 0$ per ogni coppia di report P e Q.

Si è pensato di considerare la più generica distanza di Minkowski[1] ma come riportato in [18] i risultati migliori si sono ottenuti per $n = 2$ ovvero utilizzando la distanza euclidea.

3.1.3 Metrica unificata

Uniformare i campi categorici a quelli numerici per il calcolo della similarità tra due dataset generici non porta sempre a dei buoni risultati, come dimostrato in [15]. Per questo motivo la distanza totale tra i due report è stata ottenuta come una combinazione di due metriche distinte: una utilizzata per i campi numerici e l'altra per quelli categorici.

La metrica utilizzata per i campi numerici è tratta da [16] e riportata in [18] poiché sono stati ottenuti risultati migliori nel determinare la distanza percepita tra due vettori numerici. Di seguito è riportata la formula:

$$D(P,Q)_{num} = 1 - \exp(-d(P,Q)_{num}^{\tau}) \tag{3.8}$$

Dove τ è un parametro libero che è stato posto a 1 nell'implementazione della metrica, mentre $d(P,Q)_{num}$ è la distanza euclidea normalizza-

[1] $D(X,Y) = (\sum_{i=1}^{N} |X_i - Y_i|^n)^{1/n}$

ta tra tutti i campi numerici dei report, ottenuta tramite la seguente formula:

$$d(P,Q)_{num} = \frac{\sqrt{\sum_{i=1}^{N_{num}} d(P,Q)_i^2}}{N_{comp,num}} \tag{3.9}$$

dove N_{num} è il numero di campi numerici totali e $N_{comp,num}$ è il numero di campi numerici confrontati.

Per i campi categorici è stata considerata la misura di similarità proposta in [20] e riportata di seguito:

$$S(P,Q)_{cat} = \frac{|P_{cat} \bigcap Q_{cat}|}{|P_{cat} \bigcap Q_{cat}| + a|P_{cat} - Q_{cat}|} \tag{3.10}$$

dove P_{cat} e Q_{cat} rappresentano rispettivamente l'insieme dei campi categorici dei report P e Q, $|P_{cat} \bigcap Q_{cat}|$ indica il numero di campi comuni tra i due pazienti, a è un parametro libero e $|P_{cat} - Q_{cat}|$ rappresenta la differenza tra i due insiemi, ovvero il numero di campi diversi tra i report P e Q.

In base ai risultati ottenuti in [20] si è scelto di porre $a = 1$ e quindi di utilizzare la metrica di similarità di Simpson[2] che in questo caso, potendo applicare le stesse ipotesi presenti in [20], coincide anche con il secondo coefficiente di Kulczyński[3].

Per rendere la misura simmetrica $|P_{cat} - Q_{cat}|$ è sostituita con $|(P_{cat} - Q_{cat}) \bigcup (Q_{cat} - P_{cat})|$ considerando sia i campi presenti in P_{cat} e non in Q_{cat} che il viceversa ed ottenendo quindi la differenza simmetrica degli insiemi P_{cat} e Q_{cat} ($P_{cat} \Delta Q_{cat}$).

Per ottenere una metrica che possa quindi misurare la differenza, e non la somiglianza, (3.10) è stata modificata in:

$$D(p,q)_{cat} = 1 - \frac{|P \bigcap Q|}{|P \bigcap Q| + |(P - Q) \bigcup (Q - P)|} \tag{3.11}$$

[2] $S(X,Y)_{Simpson} = \frac{|X \bigcap Y|}{min(|X|,|Y|}$

[3] $S(X,Y)_{\text{Kulczyński}} = 0.5 * \left(\frac{|X \bigcap Y|}{|X \bigcap Y| + |X - Y|} + \frac{|X \bigcap Y|}{|X \bigcap Y| + |Y - X|} \right)$

La funzione di similarità unificata riportata in [15] è la seguente:

$$S(p,q)_{Cheung} = \frac{1}{d_f}\sum_{i=1}^{N_{cat}} s(p,q)_i + \frac{1}{d_f} s_{num}(p,q) \tag{3.12}$$

dove d_f è pari al numero dei campi categorici più 1.

Sostituendo le funzioni di similarità con le metriche che definiscono la distanza e considerando che non è necessaria un'ulteriore normalizzazione poiché queste sono già realizzate nelle due metriche si ottiene quindi:

$$D(p,q) = D(p,q)_{num} + D(p,q)_{cat} \tag{3.13}$$

3.2 Pesi

Durante lo sviluppo della tesi si è tenuta in considerazione l'ipotesi di pesare i vari campi in modo tale da dare un peso maggiore a quelli ritenuti più importanti.

Per introdurre dei pesi nei campi numerici sarebbe sufficiente moltiplicare il valore normalizzato per il peso relativo al campo, di conseguenza (3.2) diventerebbe:

$$x_{P,i,norm} = w_i \frac{x_{P,i} - min(X_i)}{max(X_i) - min(X_i)} \tag{3.14}$$

dove w_i corrisponderebbe al peso dell'*i-esimo* campo.

Di conseguenza (3.3) dovrebbe essere ricalcolata tramite la seguente formula:

$$d(P,Q)_i = w_i \frac{x_{P,i} - x_{Q,i}}{max(X_i) - min(X_i)} \tag{3.15}$$

Un discorso analogo potrebbe essere fatto nelle formule (3.4) e (3.5) per la normalizzazione rispetto alla media e alla varianza.

Bisogna ricordare che, nel momento in cui venissero introdotti dei pesi, bisognerebbe anche andare a modificare (3.7) e (3.9) in modo tale che la normalizzazione non sia più rispetto al numero di campi confrontati ma rispetto alla somma dei pesi di questi campi.

Per pesare i campi categorici, invece, si potrebbe sostituire in (3.11) il conteggio dei campi (uguali o differenti) con la somma dei pesi associati ad ogni campo. In questo caso non sarebbero necessarie ulteriori modifiche alla formula.

Tuttavia durante un incontro con il Dott. Gamberini si è scelto di non inserirne nessuno per evitare un possibile condizionamento umano sull'operato del programma.

Ciò non esclude comunque la possibilità di determinare automaticamente quali siano i campi più rilevanti in base alla distribuzione dei valori in tutto il database come è stato fatto in [15].

3.3 Algoritmo sviluppato

Per effettuare il controllo in maniera esatta e trovare con precisione i pazienti più simili al nuovo arrivato è stata calcolata la distanza tra questo e coloro già presenti nel database.

Inizialmente era stata presa in considerazione l'ipotesi di effettuare un clustering tra i pazienti già esistenti in modo tale che, nel momento in cui ne arrivasse un nuovo degente, sarebbe necessario effettuare un numero minore di confronti.

Successivamente, dato il numero esiguo di pazienti e di campi da confrontare, quest'idea è stata scartata poiché un'eventuale euristica avrebbe potuto escludere dei pazienti affini a quello sotto esame, introducendo un errore. Tuttavia, dalle prove svolte si è notato che il sistema risultava essere molto veloce anche confrontando tutti i pazienti e quindi, al momento, l'euristica non risulta essere necessaria.

Si sarebbe rischiato infatti di ottenere un calo delle performance dovuto all'aggiunta di questa fase di pre-processing dei dati che non sarebbe riuscita a bilanciare il guadagno ottenuto diminuendo il numero di confronti da effettuare.

Per questi motivi l'algoritmo sviluppato è risultato essere molto semplice una volta definita con precisione la metrica da utilizzare per il confronto.

Di seguito è riportata la sequenza di istruzioni che viene utilizzata per calcolare la distanza tra il nuovo paziente ed un singolo paziente del database.

Innanzitutto è stata creata un'interfaccia *Metric* che all'interno ha i seguenti tre metodi: *computeNumbersDistance*, *computeObjectsDistance* e *computeFinalDistance*. Quest'interfaccia è stata implementata da due classi, una per ogni metrica presentata nello studio.

Per ogni gruppo di valori viene richiamata una funzione che effettua il confronto tra tutti i campi che appartengono a tale gruppo. Per ciascun campo viene poi richiamata una funzione differente in base al tipo di dati che si vogliono confrontare: *computeNumbersDistance* per i campi numerici piuttosto che *computeObjectsDistance* nel caso di campi categorici.

Entrambe le funzioni non restituiscono nulla poiché la distanza calcolata, relativa al singolo campo, è memorizzata all'interno della classe che implementa l'interfaccia *Metric*.

La funzione *computeNumbersDistance* ha come parametri il fattore di normalizzazione, che cambia in base a quello scelto nell'interfaccia grafica e i due parametri relativi ai valori del campo nel paziente nuovo e nel paziente presente nel database. Questi due parametri sono gli unici della funzione *computeObjectsDistance*.

Dopo che tutti i campi sono stati confrontati viene richiamata la funzione *computeFinalDistance*. Quest'ultima calcola la distanza totale tra i due pazienti presi in esame in base alle distanze memorizzate all'interno della classe durante tutto il confronto.

Questi calcoli vengono iterati con tutti i pazienti presenti nel database in modo tale da poterli ordinare in base alla somiglianza con l'ultimo degente arrivato in reparto.

La descrizione dell'architettura di questa componente e di come si potrebbe interfacciare con Trauma Tracker è fornita nel paragrafo 5.1.

3.4 Validazione dei risultati

Per validare i risultati sono stati estratti in maniera aleatoria 16 report dal database dell'ospedale, di questi uno è stato considerato il paziente appena arrivato in reparto mentre gli altri 15 sono stati inseriti in un nuovo database.

Il database di partenza contiene i report dei pazienti registrati da quando è stato installato *Trauma Tracker* fino al 24 aprile 2018.

Si è creato un database di dimensioni ridotte in modo tale che possa essere analizzato completamente da un medico senza che l'operazione richieda troppo tempo.

La riduzione del database ha però un risvolto negativo: avendo meno dati i fattori di normalizzazione si basano su un campione minore e quindi hanno una precisione inferiore rispetto a quelli ottenuti considerando l'intero database.

Per validare i risultati è stato chiesto al Dott. Emiliano Gamberini del centro traumatologico dell'ospedale Bufalini di Cesena di stilare una classifica dei pazienti presenti nel secondo database che vada dal più simile al più differente rispetto al paziente preso come riferimento.

Di seguito sono riportati i risultati delle due metriche utilizzando le normalizzazioni implementate.

Per valutare i risultati ottenuti dal sistema sono state utilizzate due formule d'errore:

$$E_l(C_d, C_s) = \frac{\sum_{i=1}^{N} \left| C_d(i) - C_s(i) \right|}{N} \tag{3.16}$$

$$E_l(C_d, C_s) = \frac{\sum_{i=1}^{N} \left(C_d(i) - C_s(i) \right)^2}{N} \tag{3.17}$$

dove N rappresenta il numero di pazienti, C_d rappresenta la classifica stilata dal medico, C_s la classifica calcolata dal sistema mentre $C_d(i)$ e $C_s(i)$ restituiscono la posizione dell'*i-esimo* paziente rispettivamente nella classifica del medico e in quella del sistema.

La funzione (3.17), utilizzando il quadrato dello scostamento, risalta maggiormente gli errori più rilevanti effettuati dal sistema.

Considerando N pari a 15 e due classifiche opposte tra loro (il primo in una è l'ultimo nell'altra, il secondo è penultimo e così via) la funzione definita in (3.16) restituirebbe 7,5 mentre quella che considera il quadrato delle distanze (ovvero (3.17)) restituirebbe un valore pari a 74,7.

Tabella 3.1: Risultati della validazione per la metrica euclidea

Id paziente	Medico	MinMax	MeanVariance
20170612-203351	1	4	3
20180412-022951	2	8	6
20180422-212014	3	2	2
20180120-103751	4	1	1
20180329-211303	5	11	9
20171228-134747	6	7	4
20180126-000450	7	12	12
20180210-165729	8	5	10
20180310-015132	9	6	11
20180118-012811	10	9	7
20170919-071524	11	3	5
20170831-225801	12	14	15
20170515-143945	13	15	13
20170422-121200	14	13	14
20170807-221441	15	10	8

3.4.1 Metrica euclidea

Nella tabella 3.1 sono riportati: l'identificativo del paziente, la posizione del paziente nella classifica stilata dal medico, la posizione nella classifica ottenuta utilizzando la normalizzazione *MinMax* e la posizione nella classifica derivante dalla normalizzazione *MeanVariance*. La metrica utilizzata in entrambi i casi è quella euclidea.

Si può notare in ambedue i casi che tre dei cinque pazienti più simili, secondo il medico, a quello appena arrivato sono presenti nelle prime cinque posizioni definite dal sistema informatico.

Utilizzando la normalizzazione *MeanVariance* il paziente più simile risulta essere il terzo della classifica; con *MaxMin* questo viene riportato in quarta posizione.

Applicando la funzione d'errore (3.16) alle classifiche ottenute dal sistema si ottiene un errore pari a 3,3 per la normalizzazione *MaxMin* e 2,9 per la normalizzazione *MeanVariance*.

Calcolando l'errore secondo (3.17) la normalizzazione *MaxMin* ottiene un valore pari a 15,6 mentre quella *MeanVariance* ha un pun-

Tabella 3.2: Risultati della validazione per la metrica unificata

Id paziente	Dottore	MinMax	MeanVariance
20170612-203351	1	4	4
20180412-022951	2	5	6
20180422-212014	3	6	7
20180120-103751	4	1	1
20180329-211303	5	8	9
20171228-134747	6	3	3
20180126-000450	7	12	12
20180210-165729	8	11	11
20180310-015132	9	12	13
20180118-012811	10	2	2
20170919-071524	11	9	10
20170831-225801	12	15	15
20170515-143945	13	10	8
20170422-121200	14	14	14
20170807-221441	15	7	5

teggio di 12,4.

Si nota quindi, per la metrica euclidea, che la normalizzazione basata sulla varianza funziona meglio e riesce ad ottenere dei buoni risultati poiché per ogni paziente si ottiene un errore medio minore a tre posizioni.

3.4.2 Metrica unificata

Per validare i risultati ottenuti con la metrica unificata è stata compilata la tabella 3.2 (simile alla tabella 3.1) in cui sono riportati i dati ottenuti utilizzando la metrica unificata.

Dai risultati riportati si nota che nelle prime cinque posizioni della classifica ottenuta con *MinMax* sono presenti sia i primi due pazienti più simili secondo il medico che il quarto. Invece, con la normalizzazione *MeanVariance*, nelle prime cinque posizioni è presente solo il primo e il quarto paziente della classifica stilata dal Dott. Gamberini.

In entrambi i casi il paziente più simile è stato relegato alla quarta posizione.

Calcolando l'errore mediante la funzione (3.16) si ottiene un valore pari a 3,5 per la normalizzazione *MaxMin* mentre la *MeanVariance* ha ottenuto una valutazione pari a 4. Utilizzando invece la funzione errore (3.17) i valori ottenuti dalle due normalizzazioni sono, rispettivamente, pari a 16,5 e 21,6.

L'errore commesso da entrambe le normalizzazioni che ha pregiudicato maggiormente la bontà della metrica è stato quello relativo al paziente 20180118-012811 che è stato posizionato al secondo posto della classifica quando avrebbe dovuto occuparne il decimo.

Analizzando i due report si è notato come non sia stato possibile confrontare nessun campo numerico, mentre solo in un terzo dei campi categorici uno dei due report aveva un valore non nullo.

Di conseguenza questo comportamento anomalo potrebbe essere corretto inserendo una penalità per i campi non confrontabili poiché con le attuali metriche questi campi sono semplicemente ignorati.

3.4.3 Commenti generali

I risultati ottenuti dalle validazioni non sono dei migliori ma bisogna sempre considerare che è stato utilizzato un database ridotto che non ha garantito una buona normalizzazione dei valori numerici.

Inoltre si suppone che, con un numero maggiore di report, anche se quelli riportati dal sistema non fossero i più simili al paziente appena arrivato possano comunque essere di interesse per i medici del centro traumatologico.

Tuttavia potrebbe essere comunque interessante applicare questo sottosistema nella realtà per verificare se apporti o meno dei vantaggi nell'operato dei medici.

Un altro aspetto degno di nota è la differenza tra quale delle due normalizzazioni sia risultata migliore in base alla metrica utilizzata; infatti in un caso *MaxMin* ha riportato i valori migliori mentre nell'altro è stata la normalizzazione *MeanVariance* ad avere l'errore minore.

Complessivamente questo sottosistema presenta risultati interessanti e che potrebbero essere migliorati applicando dei pesi ai singoli campi, come definito in [15], in modo tale da far risaltare maggiormente i campi che apportano una maggiore quantità di informazione e migliorare i risultati ottenuti.

Capitolo 4

Previsione dello stato di shock

Un'altra funzionalità utile ai medici durante il loro operato è quella di prevedere se il paziente attualmente in cura potrebbe andare in uno stato di shock.

Questa funzionalità è stata richiesta dai medici dell'ospedale Bufalini soprattutto per prevenire situazioni d'emergenza durante un esame TAC che impedirebbe un pronto intervento per cercare di ripristinare una situazione migliore.

4.1 Reti neurali ricorrenti

Le reti neurali artificiali sono dei modelli matematici utili a risolvere vari problemi ingegneristici di intelligenza artificiale. Il loro nome è dovuto alla somiglianza tra il modello matematico e le reti neurali biologiche che presentano tutti gli animali.

Le reti neurali sono formate da neuroni artificiali ovvero strutture dati con all'interno un valore numerico, raggruppati in più livelli. Ogni neurone è presente in uno e un solo livello ed è collegato ai neuroni (tutti o una parte) del livello successivo. In questo modo l'informazione si propaga dal livello di input a quello di output muovendosi in avanti (per questo motivo sono anche dette *feedforward networks*). Una rappresentazione grafica di queste reti è presente nella figura 4.1.

Queste reti vengono utilizzate per tutti i problemi in cui ogni input è indipendente dal successivo e non ci sono correlazioni temporali tra essi, un caso d'uso frequente delle reti neurali *feedforward* è la classificazione di immagini.

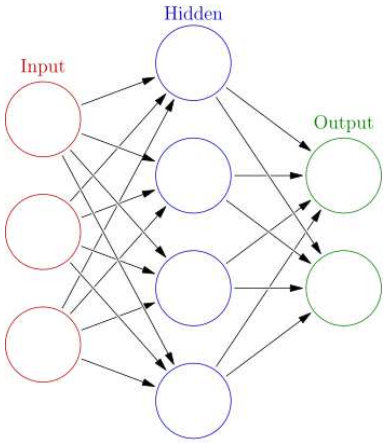

Figura 4.1: Topologia delle reti neurali *feedforward*

Queste reti non hanno la capacità di immagazzinare dei dati e quindi non sono utili nell'analisi di sequenze di dati. Per superare questa impasse sono state sviluppate le reti neurali ricorrenti.

Se invece all'interno della rete almeno un neurone è collegato a sé stesso o ad un altro neurone dello stesso livello o di un livello precedente la rete si dice ricorrente. Questo collegamento forma un ciclo che consente alla rete di memorizzare gli input ricevuti ed utilizzare queste informazioni per calcolare il nuovo output. Una rappresentazione grafica di queste reti è presente nella figura 4.2.

Quest'ultime (*Recurrent neural network*, di seguito RNN) sono utilizzate quando gli input non sono indipendenti tra di loro ma rappresentano i vari stati nel tempo di una determinata sequenza; per questo motivo queste reti si rilevano utili, ad esempio, nell'elaborazione del parlato.

Tutte le reti neurali, prima di poter essere applicate ai sistemi reali devono essere addestrate tramite delle coppie input-output. Per ognuno di questi esempi si inserisce l'input nella rete, si confronta l'output prodotto dalla rete con quello esatto (che si vorrebbe ottenere)

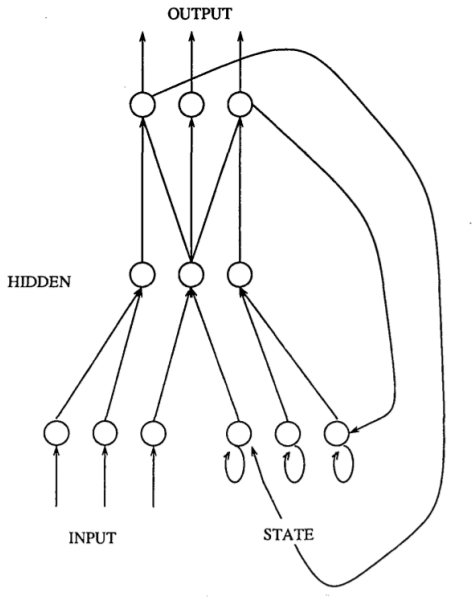

Figura 4.2: Topologia delle reti neurali ricorrenti; da [7]

e si propaga all'indietro la differenza tra i due output, in altre parole, l'errore.

4.1.1 LSTM-RNN

La *Long short-term memory - RNN* (di seguito LSTM-RNN) è una tipologia di reti ricorrenti presentata da Sepp Hochreiter e Jürgen Schmidhuber in [10].

Il vantaggio di questa rete è che non soffre del problema della scomparsa del gradiente, ovvero l'appiattimento della funzione di retropropagazione dell'errore che causa difficoltà nel momento in cui si vuole addestrare la rete.

Le reti LSTM inoltre contengono, oltre al normale flusso di dati della rete ricorrente, ulteriori informazioni che vengono salvate in delle celle *gated*. Questi dati possono essere memorizzati, scritti o letti da altre celle, ottenendo un comportamento simile a quello della memoria di un computer.

Sono poi le celle stesse, durante l'addestramento, a decidere cosa memorizzare e quando permettere lettura, scrittura e cancellazione tramite dei *gate* che si aprono e chiudono.

Tuttavia, diversamente dalle celle di memoria del computer, che sono digitali, questi gate sono analogici, implementati tramite il prodotto di sigmoidi[1], funzioni che appartengono all'intervallo $]0, 1[$, che permettono una migliore retropropagazione dell'errore rispetto a dei gate digitali.

Questi gate, come i nodi delle reti neurali, variano il loro comportamento in base ai segnali, pesati, ricevuti dagli altri neuroni e in base ai loro valore di soglia bloccando o facendo passare l'informazione. Questi pesi e valori vengono modificati durante la fase di apprendimento, come avviene in tutte le reti neurali. Il modo in cui l'informazione fluisce attraverso le varie celle e i vari gate è mostrato nella figura 4.3.

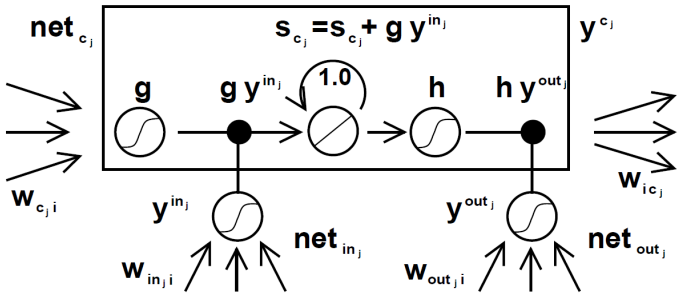

Figura 4.3: Funzionamento delle celle di una rete LSTM-RNN; da [10]

Questa è la tipologia di rete usata negli studi [1],[19] e [17] illustrati nel paragrafo 1.2.

Inoltre questa rete è stata anche utilizzata in altri studi relativi ad ambiti differenti da quello medico.

Ad esempio, nello studio effettuato in [9] è stata utilizzata una LSTM-RNN in ambito industriale per ottenere un indicatore della salute di alcuni supporti. Nonostante la differenza di ambito, si nota comunque una certa analogia con il caso analizzato in questa tesi.

[1]Generalmente si considera la seguente funzione: $P(t) = \dfrac{1}{1+e^t}$ ma possono essere pure utilizzate delle varianti.

Invece nello studio effettuato in [21] è stata sviluppata una LSTM-RNN modificata per riuscire ad ottenere delle previsioni meteorologiche a breve termine. In questo caso si sono ottenuti dei buoni risultati che hanno dimostrato l'affidabilità di questo tipo di rete.

Per avere una visione più dettagliata delle LSTM-RNN e dei casi in cui vengono applicate si può consultare la review svolta da Lipton ([14]) in cui vengono analizzate varie RNN per l'apprendimento di sequenze.

4.2 LSTM-RNN in Trauma Tracker

In questo lavoro è stata quindi sviluppata una rete di tipo LSTM-RNN per cercare di predire l'eventualità che un paziente vada in stato di shock.

Uno dei problemi inizialmente sollevato dai medici dell'ospedale Bufalini era quello relativo al rischio di un possibile stato di shock del paziente durante un esame TAC. Considerando la durata media di tale esame, si è deciso di scegliere il tempo di 15 minuti come orizzonte temporale rispetto al quale il sottosistema deve valutare il possibile stato del paziente. Quest'intervallo di tempo è stato determinato effettuando una ricerca riguardo la durata media di questo tipo di esame.

La rete andrà ad analizzare solo una porzione degli eventi che verranno man mano inseriti nel report generato da *Trauma Tracker* (riportati nel paragrafo 2.2.1) in quanto alcuni sono stati considerati irrilevanti e non modificano la probabilità che un paziente possa andare in stato di shock. Alcuni di questi eventi scartati sono il cambio di stanza o di Trauma leader.

In generale si è voluto creare un sistema simile a quello presentato in [1] data la somiglianza tra i due casi. Per questo motivo sono stati riportati gli eventi relativi a procedure, farmaci, esami diagnostici e parametri vitali.

Lo scopo della rete è quindi quello di stabilire se il paziente andrà o meno in stato di shock per ogni evento che riceverà in input, ricordando l'andamento degli eventi precedenti grazie alla sua struttura.

In questo modo si cercherà di fornire un assistente informatico che possa supportare e consigliare i membri dell'equipe medica su quali procedure effettuare e quali farmaci somministrare nel caso in cui si preveda una degenerazione delle condizioni cliniche del paziente sotto esame.

La struttura della rete e l'organizzazione dell'input è descritta in maniera più approfondita nei paragrafi seguenti.

4.3 Implementazione della rete

Per vari motivi non è stata implementata una rete da zero ma si è partiti da un framework in cui fossero già implementate reti neurali di tipo LSTM-RNN e che si potessero parametrizzare in base alle esigenze.

Data la necessità di implementare l'intero sistema nel linguaggio Java (motivo per cui è stata escluso *TensorFlow*), consentendo eventuali parallelizzazioni e garantendo un facile trasporto del calcolo della CPU alla GPU, la scelta è ricaduta sul framework *Deeplearning4j* [22] poiché soddisfa queste caratteristiche.

Questo framework ha il vantaggio di essere riutilizzabile in altri linguaggi quali C++, python e tutti quelli basati sulla JVM. Inoltre è ben documentato ed è open-source, caratteristiche che ne facilitano l'apprendimento e l'utilizzabilità.

La rete è organizzata in maniera analoga a quella realizzata in [1] ma si è cercato di semplificare la struttura dato il minor numero di dati che non permetterebbe di effettuare un addestramento con la stessa efficacia.

4.3.1 Struttura della rete

La rete è quindi organizzata in tre livelli:

1. il livello di input è formato da 74 neuroni, numero pari agli elementi del vettore di input (descritto nel paragrafo 4.3.3);

2. il livello hidden ha al suo interno 50 neuroni; questo numero è stato scelto in modo tale da mantenere la stessa proporzione tra input e hidden utilizzata in [1];

3. il livello di output è composto da 2 neuroni, corrispondenti alle classi *Non shockato* e *Shockato* che rappresentano il possibile stato del paziente.

Con questa configurazione della rete sono presenti circa 46.000 parametri da configurare correttamente per ottenere delle buone prestazioni.

Nella rete i parametri vengono inizializzati con dei valori stocastici che rispettano la distribuzione normale con media pari a 0 e varianza posta a 1.

Questi parametri vengono poi aggiornati mediante l'algoritmo Adam. Si è scelto questo metodo tra i vari poiché in [12] è risultato essere il più performante nell'aggiornamento dei pesi delle reti neurali. Il *learning rate* utilizzato è quello proposto in [12].

La funzione di attivazione del livello di input e del livello hidden è la softsign (riportata in (4.1)) poiché in [13] risulta essere la funzione migliore per le reti neurali LSTM-RNN in cui si effettua un problema di classificazione.

$$Softsign(x) = \frac{x}{1 + |x|} \tag{4.1}$$

Come si può notare in figura 4.4 la funzione softsign è quella che si satura meno velocemente, in questo modo l'algoritmo di backpropagation sarà più efficace consentendo alla rete di imparare più in fretta e con meno dati in ingresso.

Per il livello di output è stata scelta la funzione di attivazione softmax (riportata in (4.2)) che è la più utilizzata in tutte le applicazioni nel livello di output. La peculiarità di questa funzione è che restituisce, per ogni neurone, un valore compreso nell'intervallo $]0, 1]$ e che sommando tutti i valori dei neuroni si ottiene 1.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad j = 1, ..., K \tag{4.2}$$

dove K è il numero dei neuroni di output e j è il neurone di cui si sta calcolando il valore.

Seguendo gli esempi forniti dal framework si è deciso di settare la negative log likelihood (riportata in (4.3)) come loss function[2] della

[2]Funzione che indica la differenza tra i valori predetti e quelli reali

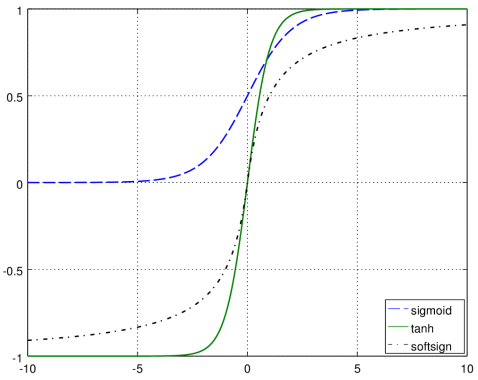

Figura 4.4: Confronto tra funzioni di attivazione; da [13]

rete.

$$
\begin{aligned}
-\log P(y|x;\theta) &= -\log \prod_{i=1}^{m} (\phi^{(i)})^{y^{(i)}} (1-\phi^{(i)})^{1-y^{(i)}} \\
&= -\sum_{i=1}^{m} \log \left(((\phi^{(i)})^{y^{(i)}} (1-\phi^{(i)})^{1-y^{(i)}}) \right) \\
&= -\sum_{i=1}^{m} y^{(i)} \log (\phi^{(i)}) + (1-y^{(i)}) \log (1-\phi^{(i)}) \qquad (4.3)
\end{aligned}
$$

4.3.2 Pre-processing

Prima di inserire i dati nella rete si effettua un confronto per verificare che l'evento registrato da *Trauma Tracker* produca una variazione dei dati presenti nel vettore di input. Questo avviene nel momento in cui si verificano degli eventi che non sono registrati nella struttura di input (es.: cambio di stanza, scatto di una foto, ecc...).

Un evento definisce l'inizio dello stato di shock del paziente quando a quest'ultimo è somministrato uno dei seguenti farmaci: adrenalina, noradrenalina, dopamina, cristalloide, colloidi, emazie concentrate, soluzione ipertonica. Questa lista è stata fornita dai medici dell'ospedale Bufalini.

Nel caso in cui un paziente non vada in stato di shock durante la fase di rianimazione tutti gli eventi (significativi) vengono immessi nella rete neurale.

Se, invece, il paziente dovesse andare in stato di shock durante il trattamento ricevuto in reparto non sarebbero immessi nella rete tutti gli eventi successivi alla somministrazione di uno dei farmaci sopra riportati. Questi eventi vengono scartati poiché sono ininfluenti rispetto alla possibilità che un paziente vada in stato di shock.

4.3.3 Input

L'input è rappresentato da un vettore che unisce i seguenti dati:

- Segni vitali: in questa sezione vengono riportati tutti i parametri che possono evolvere nel corso dell'intervento, per questo motivo sono stati esclusi i valori sulle ustioni e sulle fratture; sono invece stati inclusi i parametri vitali, i dati degli esami neurologici, le rilevazioni dei polsi, gli accessi vascolari e i dati sulle vie aeree e sulla respirazione;

- Procedure: in questa sezione viene riportato solo l'identificativo della procedura che viene effettuata. Nel caso in cui si tratti di una procedura protratta nel tempo si riporta sia l'evento in cui si definisce l'inizio che quello in cui viene definita la fine;

- Esami diagnostici: per ridurre la quantità di informazioni inserite nella rete neurale sono riportati solo gli esami per cui si hanno risultati numerici che possono essere inseriti nella rete, per questo motivo vengono considerati soltanto l'emogasanalisi e l'esame ROTEM;

- Farmaci: nella sezione dei farmaci sono stati creati vari campi, uno per ogni farmaco somministrabile in maniera continua nel tempo in cui si riporta la quantità del determinato farmaco somministrato in quel momento, inserendo 0 se non è somministrato. Oltre a questi campi sono presenti due campi relativi ai farmaci "one-shot", uno in cui viene riportato il farmaco e l'altro in cui è espressa la relativa quantità.

Sia i campi dei segni vitali che quelli degli esami diagnostici vengono riportati ad ogni passo. Se non si hanno nuove informazioni si riutilizzano i valori degli input precedenti perché si assume, come in [1], che le misurazioni sono effettuate solo se ci si aspetta delle variazioni.

Poiché tutti i dati devono essere uniformi tra loro è necessario definirne un tipo in cui tutti i valori possano essere convertiti. Per questo motivo si è optato per il tipo *double* e sono state effettuate le seguenti conversioni per omogenizzare i dati: i campi numerici sono stati trasformati in *double* senza effettuare modifiche; i campi booleani sono stati trasformati in 0 se il loro valore fosse *false*, altrimenti in 1 nel caso in cui valgano *true*; i campi *Enum* sono stati trasformati in *double* utilizzando il metodo *ordinal*; invece ai campi di tipo *String* in cui all'interno è presente del testo è stato calcolato il valore di hash ed è stato utilizzato questo come input. A tutti i campi nulli è stato sostituito -100 poiché non è un valore che potrebbe essere presente nei vari campi.

Dato che la rete è ricorrente bisogna tenere in considerazione la correlazione tra gli input immessi nella rete. Per questo motivo nella rete vengono inseriti blocchi di input in modo tale da immettere insieme, e nel giusto ordine, tutti i vettori relativi allo stesso paziente.

Alla fine quindi risulta che l'input della rete è un vettore bidimensionale di elementi di tipo *double* in cui in ogni colonna c'è lo stato del paziente in un determinato momento mentre nelle righe si possono vedere gli andamenti dei vari valori nel corso del tempo. In figura 4.5 è riportata la matrice che rappresenta i dati di input per ogni singolo paziente.

4.3.4 Output

L'output rappresenta la possibilità che un paziente vada in stato di shock nei 15 minuti successivi all'ultimo evento registrato.

Il problema potrebbe essere considerato sia come classificazione in due classi (*Shockato* e *Non shockato*) che come una regressione rispetto alla funzione che indica la probabilità che il paziente vada in stato di shock nell'intervallo di tempo considerato. Un'altra visione del problema potrebbe essere quella in cui la rete neurale individui il

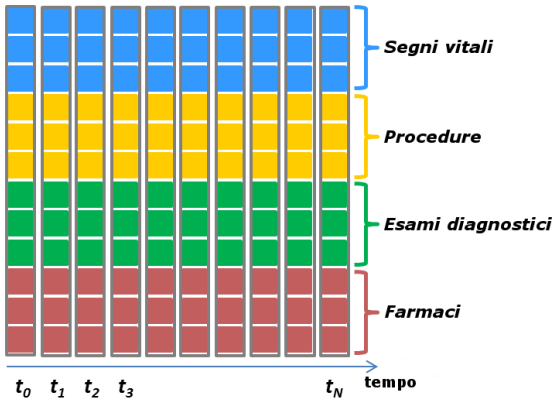

Figura 4.5: Struttura dei dati dopo il pre-processing; tratta da [1]

tempo entro il quale il paziente possa andare in shock, considerando un tempo massimo oltre al quale considerare il paziente non a rischio.

Data la bassa quantità di dati si è deciso di effettuare una classificazione più che una regressione della probabilità che il paziente vada in stato di shock o una previsione del tempo entro il quale ciò potrebbe avvenire. Questo perché il problema della classificazione è il più semplice per la rete neurale e quindi quest'ultima necessita meno dati per poter apprendere correttamente ad elaborare l'input.

Per stabilire a quale delle due classi appartenga il paziente si valutano i risultati presenti nei due neuroni di output. Il primo neurone indica la possibilità che il paziente non vada in stato di shock; viceversa per il secondo neurone.

Di conseguenza la classificazione viene effettuata scegliendo la classe corrispondente al valore più alto tra i due. I valori presenti nei neuroni di output potrebbero essere utilizzati anche per calcolare la probabilità che il paziente vada in stato di shock; data la bassa accuratezza si è preferito rimandare questo calcolo, che richiede un'accuratezza maggiore, ad un momento successivo.

4.3.5 Allenamento

Il database utilizzato in questa tesi ha all'interno i dati dei pazienti ricoverati presso il centro traumatologico dell'ospedale Bufalini fino al

24 aprile 2018. È composto da 281 pazienti di cui 161 hanno subito uno shock durante la rianimazione.

La rete è stata allenata utilizzando l'80% del database sopra descritto. La restante parte dei dati è stata utilizzata successivamente per effettuare la validazione della rete.

I pazienti utilizzati sono stati scelti in base all'ordine presente nel database senza aggiungere una componente stocastica nella scelta di quale utilizzare per questa fase. Si è deciso di operare in questo modo dato che i pazienti finiti in stato di shock erano già sparsi nel database e non raggruppati in un solo settore.

Sono stati utilizzati i dati di 225 pazienti ottenendo circa 1100 eventi per addestrare la rete. Si ha quindi una media poco superiore a 4 eventi significativi prima che il paziente vada in stato di shock.

Questo numero così basso di eventi per paziente è dovuto al fatto che, essendo il centro traumatologico dell'ospedale Bufalini uno dei più importanti della zona, molti pazienti arrivano in condizioni critiche quando lo shock è imminente, con poco margine temporale di intervento e prevenzione da parte dei medici.

Inoltre dato che il sistema *Trauma Tracker* è stato introdotto da poco tempo non sono stati registrati numerosi pazienti nel database. Inoltre è importante evidenziare che è stato necessario un periodo di tempo rilevante prima che *Trauma Tracker* venisse utilizzato a pieno regime.

4.4 Validazione dei risultati

Data la bassa quantità di dati presenti nel database rispetto ai parametri della rete non è stato possibile addestrarla in maniera adeguata.

Come riportato nella pagina del framework utilizzato in cui vengono descritte le reti di tipo LSTM-RNN[3] è necessario, per avere una rete di buona qualità, che il numero di esempi sia maggiore di quello dei parametri. In questo caso, invece, il numero di parametri è circa 40 volte più grande di quello degli esempi forniti alla rete e per questo motivo non è possibile sperare di ottenere dei buoni risultati.

[3]https://deeplearning4j.org/lstm

Tabella 4.1: Confusion matrix

	Non shockato	Shockato
Non shockato	227	18
Shockato	68	37

Tabella 4.2: Risultati ottenuti in 5 esecuzioni

Risultato	1	2	3	4	5
Accuratezza	75,4%	65%	80%	78,8%	74,7%
Precisione	72,1%	63,8%	77,1%	72,3%	72,4%
Recall	64%	60,2%	66,1%	66,2%	65,9%
F1	46,3%	45%	49,2%	49,2%	51,2%

Per la validazione della rete sono stati utilizzati i dati relativi a 56 pazienti, per un numero totale di input pari a 331 eventi.

Validando la rete si è ottenuta un'accuratezza del 75,4% e per la classe *Shocked* sono stati ottenuti i seguenti risultati: precisione del 72,1%, recall del 64% e F1 del 46,3%. Inoltre nella tabella 4.1 è riportata la *confusion matrix*[4] relativa alla validazione della rete.

Effettuando ulteriori validazioni si è notata una varianza non indifferente dei risultati. Nella tabella 4.2 sono riportati i valori di accuratezza, precisione, recall e F1 in 5 esecuzioni differenti.

Da queste esecuzioni, svolte in sequenza e senza aver modificato nulla, si nota come tutti gli indicatori fluttuino su vari intervalli.

Si può dedurre quindi che l'efficacia della rete dipenda dai valori iniziali dei parametri, che sono settati in maniera aleatoria.

Per questo motivo non è stato ritenuto di particolare interesse provare la rete con differenti configurazioni modificando il numero di parametri, l'inizializzazione dei parametri, le funzioni di attivazioni e la loss function (tutto ciò che è stato definito inizialmente).

[4]La matrice è in formato "Classe riga effettiva predetta come classe colonna N volte"

Capitolo 5

Architettura dei sotto sistemi implementati

Il lavoro svolto in questa tesi non è stato collegato con *Trauma Tracker* ma è stato sviluppato con l'obiettivo che la fusione tra i due sistemi possa essere quanto più semplice possibile.

Di seguito è descritta l'architettura di entrambi i sottosistemi mostrando come questi possano essere integrati con il sistema *Trauma Tracker*.

5.1 Confronto pazienti simili

Per il confronto tra i pazienti il sistema creato si interfaccia con il database MongoDB in cui sono memorizzati tutti i report dei pazienti già ricoverati precedentemente presso l'ospedale Bufalini di Cesena.

Nel momento in cui arriva un nuovo paziente il sistema si connette al database per caricare in memoria tutti i report precedenti al fine di poterli trasformare in strutture dati apposite che facilitino il confronto tra i pazienti stessi, per ottimizzare il processo questa trasformazione è eseguita in parallelo.

Una volta conclusa questa fase si trasforma il report del nuovo paziente nella stessa struttura in cui sono stati convertiti i report precedenti.

A questo punto il controllo passa alla classe *CalculateSimilarPatients* che confronta tutti i pazienti presenti nel database con quello

attuale (anche quest'operazione è svolta in parallelo) e ad ogni paziente viene associato un valore numerico che rappresenta la distanza dal nuovo arrivato. Questo passaggio è descritto in maniera più approfondita nel paragrafo 3.3.

Infine i risultati sono ordinati dal paziente più simile a quello più differente e sull'interfaccia sono mostrati gli N pazienti più simili. N è un valore arbitrario che può essere deciso in accordo con i medici in base alle loro necessità e preferenze; pochi pazienti potrebbero portare poca informazione ma troppi risulterebbero d'intralcio e rallenterebbero le operazioni.

Tutte queste operazioni sono rappresentate in forma grafica nell'activity diagram presente in figura 5.1. In questo diagramma una singola freccia da un'operazione all'altra rappresenta una computazione sequenziale mentre più frecce rappresentano un calcolo effettuato in parallelo.

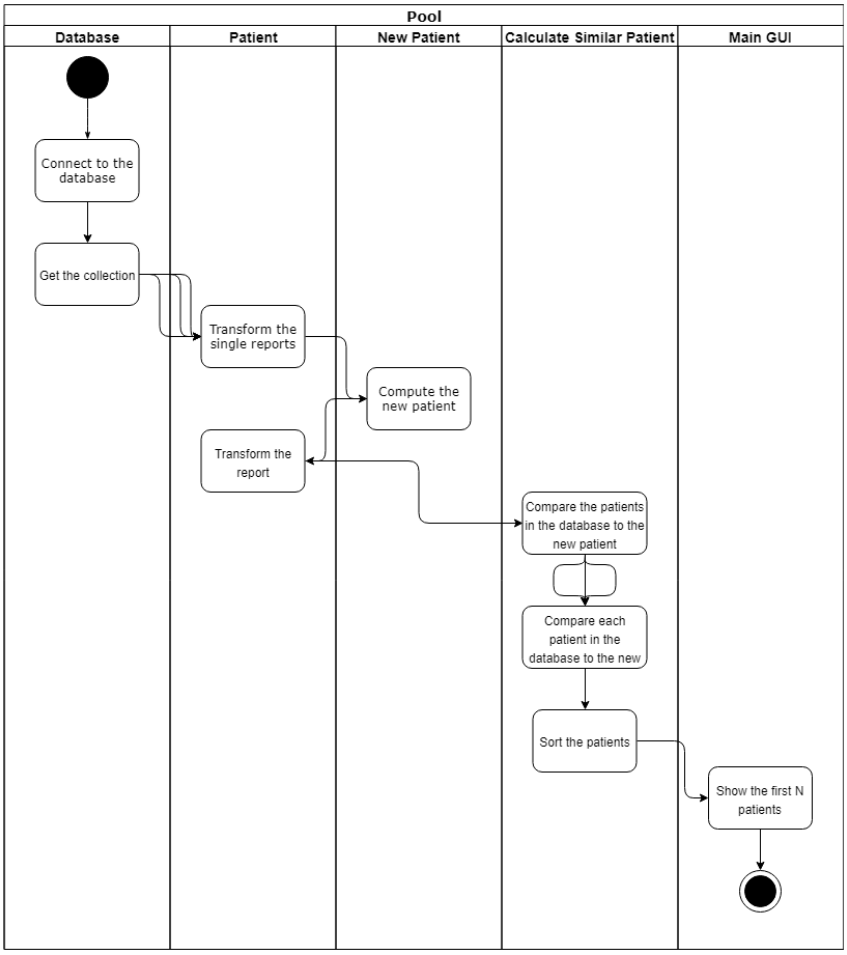

Figura 5.1: Activity diagram della sottoparte relativa al confronto tra pazienti simili

5.2 Previsione dello stato di shock

Questa seconda parte del sistema ha un'architettura differente rispetto alla precedente poiché si basa su un sistema completamente diverso.

Poiché la rete neurale è un modello matematico in cui l'apprendimento è una parte fondamentale, sarebbe opportuno che questa continuasse a imparare anche in seguito alla sua installazione.

Per fare ciò sono possibili diverse alternative:

- apprendimento on-line: in questo modo la rete apprende man mano che predice i risultati durante il suo utilizzo. Tuttavia dato che questi dipendono da eventi futuri questa possibilità potrebbe essere complessa da realizzare;

- apprendimento incrementale: viene memorizzata la rete ottenuta al termine del primo apprendimento. Successivamente si aspetta di memorizzare un numero rilevante di dati e si effettua un nuovo apprendimento partendo dalla rete già esistente.

Supponendo di utilizzare il secondo metodo di apprendimento si potrebbe creare un nuovo componente di *Trauma Tracker* che abbia al suo interno la rete neurale e un buffer in cui memorizzare i dati dei pazienti che non sono stati ancora utilizzati per l'apprendimento della rete.

Una rappresentazione grafica del possibile componente da installare per fornire la predizione dello stato di shock è presente in figura 5.2.

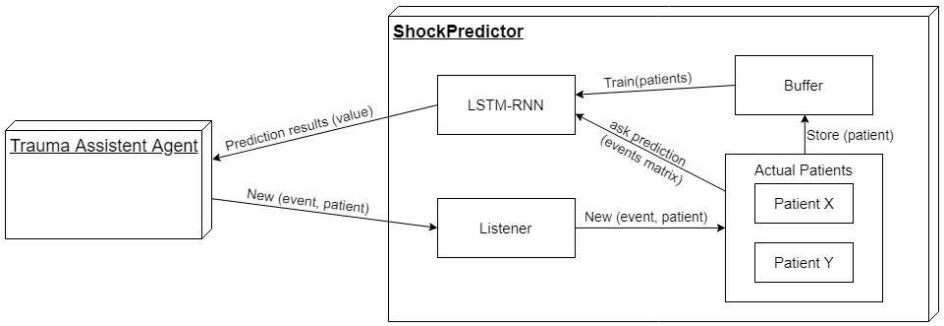

Figura 5.2: Possibile architettura del sotto sistema *Shock Predictor*

Il nuovo componente *Shock Predictor* si interfaccia esternamente solo con *Trauma Assistent Agent* poiché è questo agente incaricato a registrare i nuovi eventi che avvengono durante il trattamento ed è anche colui che gestisce l'interfaccia tra l'equipe medica e il sistema informatico.

Ogni volta che viene registrato un nuovo evento relativo ad un paziente viene notificato il *Listener* che è sempre in ascolto e che inoltra l'evento e il codice del paziente ad *Actual Patients* che gestisce i pazienti attualmente in rianimazione.

Actual Patients confronterà l'identificativo del paziente a cui si riferisce l'evento con quelli dei pazienti presenti nella sua memoria. Se il paziente non rientrasse tra quelli memorizzati ne viene creato uno nuovo, invece se si trattasse di un evento di fine intervento si andrebbe ad immagazzinare il report completo in *Buffer*. Nel caso in cui l'evento non sia nè il primo nè l'ultimo viene creato il vettore relativo al nuovo stato e confrontato con quello dello stato precedente per verificare se l'evento appena inserito abbia fornito indicazioni utili per poter determinare la possibilità che il paziente vada in stato di shock.

Se l'evento di cui sopra fosse il primo per il paziente o avesse fornito nuove informazioni, il sotto-componente *Actual Patients* invierà i dati del paziente in forma matriciale (come descritto nel paragrafo 4.3.3) al sotto-componente corrispondente alla rete neurale, *LSTM-RNN*.

Questo, dopo aver elaborato i dati invierà i risultati all'agente *Trauma Assistent Agent*, il quale si occuperà di mostrarli all'equipe medica.

La presenza di una lista permette di gestire in maniera corretta più pazienti contemporaneamente. Per lo stesso motivo è importante che ogni volta la rete rielabori tutta la storia del paziente e che non aggiunga agli ultimi risultati il nuovo evento.

Nel momento in cui sia stato raggiunto un numero congruo di pazienti nel buffer e in cui non ci sia nessuna persona sotto trattamento si possono aggiornare i parametri della rete addestrandola con gli ultimi pazienti in cura presso il centro traumatologico.

Questo è un possibile modo di integrare la nuova componente al sistema *Trauma Tracker*. Lo studio su quale possa essere la scel-

ta migliore e sull'effettiva integrazione è rimandato ad uno sviluppo futuro.

Capitolo 6

Conclusioni e possibili sviluppi

I due sottosistemi sviluppati in questa tesi potrebbero rivelarsi utili nell'operato del centro traumatologico dell'ospedale Bufalini di Cesena.

Di seguito viene effettuata una ricapitolazione dei risultati ottenuti e vengono descritti quelli che potrebbero essere i prossimi passi da realizzare relativi ai due sottosistemi.

6.1 Calcolo di pazienti simili

Il primo sottosistema descritto ha ottenuto dei risultati mediocri in fase di validazione, tuttavia, come descritto alla fine del capitolo 3 potrebbero esserci dei margini di miglioramento nel momento in cui si riuscisse ad effettuare una comparazione pesando ogni singolo campo e attribuendo maggior importanza a quei campi con valori che si discostano maggiormente dal resto dei report.

Un possibile sviluppo potrebbe essere quello di inserire una normalizzazione anche per i campi categorici, in maniera tale che i valori meno usuali abbiano un maggior peso durante il calcolo della distanza. In questo modo si andrebbe ad assegnare ad ogni singolo campo un peso differente in base al valore contenuto.

Inoltre si potrebbe modificare l'algoritmo attualmente implementato in maniera tale che, dopo esser stato analizzato, il nuovo paziente

venga aggiunto al database aggiornando costantemente i fattori di normalizzazione.

In questo modo si potrebbe tenere il database sempre in memoria senza doverlo caricare ogni volta che un nuovo paziente viene ricoverato presso il centro traumatologico.

Nel momento in cui il confronto con tutti i report dovesse iniziare ad essere lento diventerebbe necessario implementare un'euristica per non dover confrontare il nuovo report con tutti quelli presenti nel database.

6.2 Previsione dello stato di shock

In fase di validazione questo sottosistema ha ottenuto dei buoni risultati nonostante la bassa quantità di dati che non hanno permesso di addestrare la rete ad elaborare correttamente i valori ricevuti in input.

Prima di rendere attivo questo sottosistema bisognerebbe valutare la possibilità di modificare la loss function ed utilizzarne una asimmetrica. In questo modo si potrebbe aumentare l'indice di recall per la classe *Shockato* in maniera tale che i medici abbiano una maggiore probabilità di avere l'informazione che il paziente vada in stato di shock anche a discapito di un aumento dei falsi allarmi.

Tuttavia certe configurazioni dovrebbero essere esaminate attentamente con i medici in modo da ottenere il loro parere su cosa sia più importante durante il corso del trattamento di rianimazione.

Inoltre, nel momento in cui si avrà una quantità maggiore di dati, si potrebbero aumentare le dimensioni della rete in maniera tale da utilizzare un numero maggiore di input e poter ottenere risultati più precisi.

Nel caso in cui fosse possibile utilizzare una GPU sarebbe importante modificare l'implementazione della rete in modo tale da poter sfruttare la scheda grafica alleggerendo i calcoli che deve fare la CPU e ottenendo delle prestazioni migliori.

Un ulteriore sviluppo futuro potrebbe essere quello di realizzare l'architettura descritta in 5.2, o una simile, in modo tale da poter integrare a Trauma Tracker questo sottosistema in maniera rapida senza doverlo modificare.

6.3 Commenti generali

Oltre a migliorare i due sottosistemi un possibile sviluppo futuro può essere quello di integrarli in Trauma Tracker. Infatti, nonostante tutto sia stato realizzato con l'idea di poter integrare i sottosistemi facilmente, questi sono ancora staccati da Trauma Tracker e non operativi presso l'ospedale Bufalini.

Sarebbe infatti interessante poter validare i due sottosistemi utilizzandoli nell'ambiente ospedaliero in modo tale da avere un feedback da tutti i membri dell'equipe medica e poterli perfezionare in base alle loro necessità.

Complessivamente i sottosistemi sviluppati hanno ottenuto dei risultati interessanti; inoltre implementando gli sviluppi precedentemente proposti si potrebbero ottenere dei risultati migliori che siano effettivamente d'aiuto durante la prima fase di trattamento del trauma.

Appendice A

Materiale informatico

Il progetto è stato implementato nel linguaggio Java in modo tale da renderlo uniforme al resto del sistema Trauma Tracker.

L'intera implementazione del progetto è disponibile nella repository pubblica raggiungibile tramite l'indirizzo `bitbucket.org/Enrico_Cagnazzo/traumatrackersimilarpatients` .

I dati utilizzati nel corso della tesi non sono disponibili nel rispetto della privacy dei pazienti ricoverati presso il centro traumatologico dell'ospedale Bufalini di Cesena.

Bibliografia

[1] M. Aczon, D. Ledbetter, L. Ho, A. Gunny, A. Flynn, J. Williams, and R. Wetzel. Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks. *arXiv preprint arXiv:1701.06675*, 2017.

[2] L. E. Brooke and M. R. M. The golden hour: Scientific fact or medical "urban legend"? *Academic Emergency Medicine*, 8(7):758–760.

[3] I. Civil. What is quality care in trauma? *Injury*, 38(5):525 – 526, 2007.

[4] A. R. Cohen and P. M. B. Vitányi. Normalized compression distance of multisets with applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1602–1614, Aug 2015.

[5] A. Croatti, S. Montagna, and A. Ricci. A personal medical digital assistant agent for supporting human operators in emergency scenarios. In G. Sukthankar and J. A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 228–244, Cham, 2017. Springer International Publishing.

[6] M. Dupanovic. *ABCDE of Trauma Care*, pages 1–6. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[7] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[8] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[9] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, 240:98–109, 2017.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] D. Isern, D. Sánchez, and A. Moreno. Agents applied in health care: A review. 79:145–66, 03 2010.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] P. Le and W. Zuidema. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*, 2015.

[14] Z. C. Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015.

[15] Y. ming Cheung and H. Jia. Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number. *Pattern Recognition*, 46(8):2228 – 2238, 2013.

[16] A. D. Narasimhalu, M. S. Kankanhalli, and J. Wu. Benchmarking multimedia databases. *Multimedia Tools and Applications*, 4(3):333–356, 1997.

[17] R. K. Pathinarupothi, R. Vinaykumar, E. Rangan, E. Gopalakrishnan, and K. Soman. Instantaneous heart rate as a robust feature for sleep apnea severity detection using deep learning. In *Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference on*, pages 293–296. IEEE, 2017.

[18] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, Sep 1999.

[19] A. Sathyanarayana, S. Joty, L. Fernandez-Luque, F. Ofli, J. Srivastava, A. Elmagarmid, T. Arora, and S. Taheri. Sleep quality prediction from wearable data using deep learning. *JMIR mHealth and uHealth*, 4(4), 2016.

[20] S. Sharma and M. Singh. Generalized similarity measure for categorical data clustering. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 765–769, Sept 2016.

[21] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 802–810. Curran Associates, Inc., 2015.

[22] E. D. D. Team. Deeplearning4j: Open-source distributed deep learning for the jvm, apache software foundation license 2.0. `http://deeplearning4j.org`.

Ringraziamenti

Dopo un periodo di intenso lavoro questa tesi e il mio percorso di studente sono giunti al termine.

Le figure più importanti, durante questo lavoro, sono state per me il professor Alessandro Ricci, la dottoressa Sara Montagna e il dottore Emiliano Gamberini dell'ospedale Maurizio Bufalini di Cesena.

Sono infinitamente grato per il loro sostegno durante la mia tesi, il loro aiuto è stato fondamentale per ottenere un buon lavoro finale.

Vorrei ringraziare anche la mia famiglia che mi ha sempre assistito durante tutto il mio percorso da studente che in questi giorni sta giungendo a termine.

Non posso esimermi dal ringraziare anche tutti gli amici, sparsi per l'Italia e anche fuori, che mi sono sempre stati affianco.

In queste poche parole vorrei raggiungere tutte le persone che incrociato finora e che mi hanno permesso di crescere e maturare. Confrontarsi con gente dalle opinioni diverse e ascoltare strambe storie di vita vissuta permettono sempre di apprendere qualcosa che i libri e i manuali non riescono ad esprimere.

Grazie a tutti.

Enrico Cagnazzo