ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA TRIENNALE IN INGEGNERIA E SCIENZE INFORMATICHE

DIALOGO UOMO-MACCHINA NELLA PIATTAFORMA ROBOTICA NAO ALDEBARAN

Tesi in

Architetture degli elaboratori

Relatore:
Prof. Davide Maltoni

Presentata da:
Chiara Volonnino

Correlatore:
Dott. Simone Costanzi

Sessione II
Anno Accademico 2016/2017

A mia nonna,
che da lassù so che è fiera di me
e, a nonno, mamma, babbo, Nadia e le couc,
che non hanno mai smesso di sostenermi.

Elenco delle figure

Elenco dei codici

Indice

Introduzione

In un mondo che è sempre più incentrato e aperto al progresso tecnologico, dove ad oggi è possibile raggiungere traguardi che fino a pochi decenni fa erano inavvicinabili, risulta inopportuno non porsi una domanda: i robot, potranno mai acquisire una personalità autonoma che li renda in grado di pensare e sapersi interfacciare con l'uomo e con suoi altri simili? Questo ha portato con sé l'esplosione di uno studio generale sempre più incentrato su un approccio di tipo uomo-centrico, dove viene posto l'uomo come modello principale da seguire per poter implementare una macchina a sua immagine e somiglianza. I risultati, anche se in netto miglioramento, risultano però ancora lontani, questo è alimentato dall'essere intrinseco dell'uomo.

L'intelligenza artificiale può essere definita come un cambiamento sia tecnologico che culturale, che ha avuto un forte impatto sociale ed di conseguenza l'aumento e la diffusione dell'utilizzo di macchine sempre più improntate a compiere specifiche azioni in base ai comandi che gli vengono impartiti, dalla supervisione diretta dell'uomo o autonomamente, basandosi su linee guida generali.

Figura 1: Nao robot

La tesi si concentrerà sullo sviluppo di un'applicazione di dialogo con il robot umanoide Nao [Figura 1] che renda l'interazione uomo-macchina il più naturale possibile e, quindi atta ad essere utilizzata da chiunque non possieda particolari competenze informatiche o abilità di programmazione. Nao è uno fra i più famosi e migliori social robot esistente a livello mondiale sviluppato nel 2004 dall'azienda francese Aldebaran Robotics.

Il lavoro svolto mira alla progettazione e allo sviluppo di un database di conoscenza, ampliato e integrato da una serie di regole più complesse che diano la possibilità di dialogare con il robot su un argomento a "piacere" e quindi rendano capace quest'ultimo di saper interagire e rispondere a domande a lui poste in completa autonomia. In generale l'idea dello studio consiste nell'analizzare i moduli esistenti integrati nell'SDK, nel nostro caso Python SDK for Nao, e capire se esistono soluzioni alternative rispetto a quelle base.

Il progetto di tesi in questo documento è stato strutturato considerando la componente tecnologica, quella sociale e culturale del robot.

Nel primo capitolo si vuole dare una panoramica dei cambiamenti tecnologici e sociali legati alla nascita dell'intelligenza artificiale seguita da una breve introduzione, vengono poi spiegati i principi di base dell'interazione uomo-macchina. Ci sarà quindi un'analisi della comunicazione tra agenti e dei processi che portano un'agente autonomo alla comprensione di una frase, tramite l' "analisi grammaticale" di questa. Successivamente andremo ad analizzare più nel particolare gli aspetti di nostro interesse: il riconoscimento vocale e la nascita dei modelli di autoapprendimento, quali, fra le altre cose, il ramo del deep learning. Questi aspetti cercano di rendere l'interfacciamento del dialogo con le macchine il più naturale possibile, tramite lo sfruttamento delle reti neurali profonde.

Nel secondo capitolo si vuole andare a presentare la tecnologia sulla quale sono state incentrate le analisi di questa tesi. Si parlerà dunque del robot umanoide Nao Aldebaran presentando nello specifico tutte le sue componenti e funzionalità. L'intento è quello di far comprendere al lettore tutti i principi di funzionamento e i linguaggi con i quali Nao può essere programmato. Si presenterà anche il sistema operativo sviluppato appositamente per lui. Infine daremo uno sguardo alle applicazioni già esistenti ed applicate per rendere possibile molteplici attività.

Il terzo capitolo risulterà fondamentale per capire le varie scelte implementative intraprese per lo sviluppo del progetto e, di conseguenza anche le tecnologie usate, fino ad arrivare ai risultati e alle conclusioni dello studio fatto. Vedremo inoltre esposte le principali funzionalità prestate dalla Nao SDK per svolgere un'attività di programmazione delle funzionalità di dialogo.

Per concludere, nell'ultimo capitolo andremo ad illustrare in modo più dettagliato i requisiti e il design dell'applicazione che è stata creata con lo scopo di rendere possibile a tutti gli utenti di effettuare una conversazione informale con il robot ed esporremo i suoi possibili sviluppi futuri. L'applicazione si chiamerà "DIALOG WITH NAO".

Capitolo 1

Intelligenza artificiale

> *<<L'intelligenza artificiale (IA) si può definire come l'insieme di studi e tecniche che tendono alla realizzazione di macchine, specialmente calcolatori elettronici, in grado di risolvere problemi e di riprodurre attività proprie dell'intelligenza umana.>>* [13]

L'intelligenza artificiale, risalente agli anni '40, ma che solo recentemente è stata oggetto di grandi progressi, si pone come da definizione, lo scopo fondamentale di "dare la vita" alle macchine. Essa si basa principalmente su quattro concetti:

1. pensare umanamente: studio delle scienze cognitive, ovvero lo studio del funzionamento della mente umana;

2. pensare razionalmente: con obiettivo la risoluzione logica di problemi, cercando di sviluppare macchine in grado di ragionare;

3. agire umanamente: seguendo il test di Turing, sviluppato nel 1950 da Alan Turing, il quale prevede di sottoporre ad un automa alcune domande poste da un operatore umano in forma scritta. La macchina passerà positivamente il test se, a fine questionario, l'utente umano non riconosce se si trova dinnanzi ad un automa oppure ad un suo simile [18];

4. agire razionalmente: seguendo l'approccio degli agenti razionali, ovvero prevede che la macchina agisca per ottenere i migliori risultati, facendo la "cosa giusta".

1.1 Interazione uomo-robot

Human-Robot Interaction (HRI) è la scienza multidisciplinare che studia l'interazione tra uomo e macchina. Rappresenta un tassello fondamentale per la nascita dei robot umanoidi e industriali, motivata dai più recenti sviluppi tecnologici che hanno permesso ai robot di uscire dal mondo della fantascienza per entrare nel mondo presente. Oggetto di gran discussione sono i meccanismi avanzati di interazione necessari per permettere a persone, senza specifiche conoscenze e capacità, di interagire facilmente con i robot.
Le complicazioni legate alla progettazione di interfacce oltre che l'interazione con l'uomo sono da molto tempo oggetto della ricerca nell'ambito della robotica. Se dapprima lo studio di tali questioni era concentrato su un approccio "robot-centrico" con maggior accento sulla sfida tecnologica per ottenere controllo e mobilità intelligenti; recentemente invece ci si è spostati verso una ricerca per i quali i robot necessitino di un'intelligenza di tipo "umano" e perciò le macchine devono essere capaci di interagire con gli esseri umani e tra di loro, nella medesima maniera in cui gli uomini comunicano tra di essi. Questo approccio alla robotica, di tipo "uomo-centrico", enfatizza lo studio degli esseri umani come modelli per i robot. [1]

La comunicazione tra un robot e un essere umano può avvenire tramite metodi "classici", ovvero tramite l'interazione macchina-computer, oppure tramite modalità più "naturali" (*human friendly*), ovvero tipiche dell'interazione uomo-uomo.
Per quanto riguarda quest'ultimo approccio possiamo distinguere sei principali categorie di interazione:

- parlato;
- gesti;

- espressioni facciali;
- tracciamento dello sguardo;
- prossemica e cinesica;
- aptica.

In questa tesi andremo a considerare e studiare nello specifico l'approccio human friendly per quanto riguarda il parlato.

1.1.1 Introduzioni agli Agenti

Un agente è un sistema che percepisce l'ambiente attraverso i sensori e agisce su di esso mediante gli attuatori. Per percezione (*percept*) si intendono gli input percettivi dell'agente in un dato istante.
In termini matematici, si può definire una *funzione agente* che descrive la correlazione di una sequenza percettiva a una specifica azione che verrà poi implementata internamente, da un *programma agente*.

Un Agente razionale è un agente che "fa la cosa giusta", ovvero cerca di ottenere il massimo grado di successo seguendo un dato comportamento in fase di risoluzione di un problema. Questo grado di successo verrà poi calcolato in base a misure di prestazione prestabilite dall'utente. É importante sottolineare che la razionalità di un agente dipende da quattro fattori:

- misura di prestazione che definisce il grado di successo;
- conoscenza pregressa dell'ambiente da parte dell'agente;
- azioni che l'agente può effettuare;
- sequenza percettiva dell'agente fino all'istante corrente.

Ciò ci permette di dare una più specifica definizione di agente razionale:

> *<< per ogni possibile sequenza di percezioni, un agente razionale, dovrebbe scegliere un'azione che massimizzi la sua misura di prestazione*

attesa, date le informazioni fornite dalla sequenza percettiva e da ogni ulteriore conoscenza dell'agente>>.

Oltre a essere razionale, un agente dovrebbe anche essere in grado di imparare automaticamente e autonomamente. Per questo molti programmi agente sono convertibili in agenti capaci di apprendere. [17]

1.2 Comunicazione

Come affrontato nel volume 2 del libro *Intelligenza Artificiale, un approccio moderno*, di Russell S. e Norvig P., la comunicazione rappresenta l'azione di un agente di riproduzione del linguaggio attraverso un atto linguistico (*speech act*), il quale non è legato unicamente al parlato, ma a qualsiasi modalità di comunicazione, per questo si considerano due agenti, parlante e ascoltatore, e un enunciato. Il parlante e l'ascoltatore sono agenti che compiono un atto linguistico, questi possono interrogare, informare, richiedere di eseguire azioni da altri agenti, acconsentire e promettere.

Il problema della comunicazione come definito in [16], risiede nell'ambiguità grammaticale e semantica della linguaggio naturale, ovvero del linguaggio comunemente parlato dalle persone (es. inglese o italiano), che non prevede regole precise ed universali. Questo differisce da un linguaggio formale, composto da un insieme di stringhe e definito da regole matematiche precise (es. JAVA).
Per comprendere un testo in linguaggio naturale, esso deve essere trattato come un linguaggio formale, anche se siffatto porterà difficilmente ad un risultato ottimale. Questo perché la maggior parte delle frasi presentano ambiguità. Le regole di un linguaggio formale sono definite dalla grammatica, per un linguaggio naturale queste regole non sono precise e possono verificarsi delle eccezioni. Ad ogni stringa viene associata una semantica, ovvero un significato. Molto importante risulta anche la pragmatica di una stringa, ossia il suo effettivo significato nel momento in cui viene pronunciata che

dipende dal contesto extra-linguistico in cui ci si trova, ad esempio fattori ambientali, sociali, ecc.

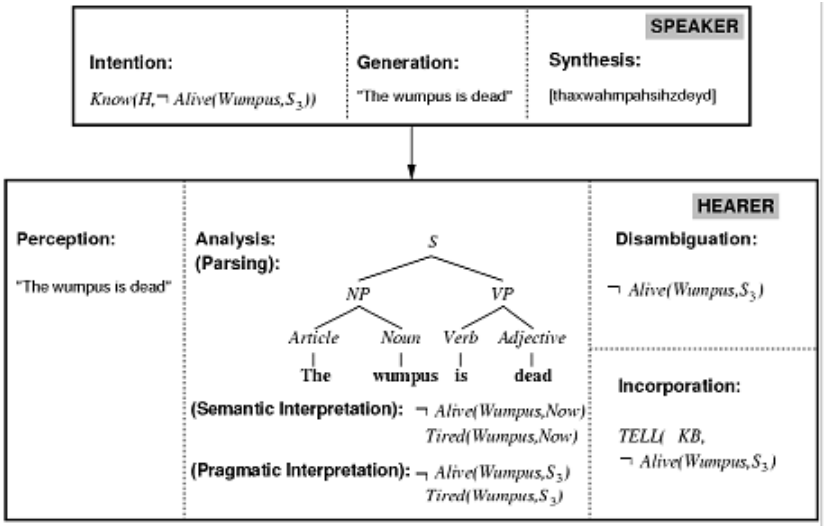

Figura 1.1: I sette processi della comunicazione [16]

Per descrivere la grammatica [16] nella maggior parte dei casi si utilizza una struttura sintagmatica, quindi le stringhe sono formate da sotto-stringhe, chiamati sintagmi, e vengono poi suddivise in categorie. Un sintagma può essere di tipo nominale (NP) o verbale (VP). Un sintagma nominale seguito da un sintagma verbale genera una frase (S).

Un tipico episodio comunicativo è composto da sette processi [Figura 1.1]:

- **Intenzione**: il parlante decide di comunicare una preposizione ad uno specifico ascoltatore.

- **Generazione**: il parlante organizza la trasformazione della preposizione in un enunciato.

- **Sintesi**: costituisce la concretizzazione fisica della parola dell'enunciato che il parlante vuole notificare.

- **Percezione**: l'ascoltatore apprende la realizzazione fisica effettuata dal parlante.
- **Analisi**: l'ascoltatore decodifica i possibili significati dell'enunciato. Questa procedura si suddivide a sua volta in tre step:
  - interpretazione sintattica o parsing, ovvero il processo nel quale viene creato un albero sintattico della stringa in input, dove i nodi interni rappresentano i sintagmi e le foglie le parole;
  - interpretazione semantica, fase di studio del significato di un enunciato;
  - interpretazione pragmatica, dove si tiene in considerazione che una parola può avere significati diversi a seconda del contesto nel quale essa viene pronunciata.
- **Disambiguazione**: l'ascoltatore sceglie la migliore interpretazione tra quelle proposte.
- **Assimilazione**: fase in cui l'ascoltatore decide di "credere o meno" al parlante.

1.2.1 Analisi sintattica (parsing)

Analizzando più nel dettaglio l'analisi sintattica deduciamo che il parsing può essere considerato un processo di ricerca di un albero sintattico e, lo spazio di ricerca può essere analizzato seguendo principalmente due politiche: parsing top-down, inizia dal simbolo S (root dell'albero) e percorre un albero che contenga le parole che compongono l'enunciato creando nodi depth-first. Il parsing bottom-up invece, parte dalle parole e ricostruisce un albero, seguendo le regole ricorsive a sinistra, ovvero analizza l'input da sinistra verso destra, cercando di arrivare alla radice S [16].
Tuttavia entrambi i metodi possono essere drasticamente inefficienti a causa dell'elevato numero di combinazioni. Infatti si considerino, per esempio due

frasi, un comando e un domanda, esposte nella medesima forma e discordanti solo nella punteggiatura, in caso di errore l'analisi dovrà assere ripetuta dal principio, impiegando così un'elevata mole di tempo.
Per ovviare questo problema [16] e creare un algoritmo di parsing efficiente si è deciso di rifarsi alla programmazione dinamica, producendo un parser in grado di memorizzare i risultati certi, ottenuti durante l'analisi, in una struttura denominata chart e quindi, in caso di errore non dover più riesaminare l'intero enunciato.

1.2.2 Natural language processing

Il Natural Language Processing, NLP (o elaborazione del linguaggio naturale, in italiano) è il campo di studi che si concentra nell'interazione del parlato tra l'uomo e la macchina. Sostanzialmente è un processo di trattamento automatico per mezzo di un calcolatore elettronico delle informazioni scritte e parlate in lingua naturale.
Utilizzando NLP, gli sviluppatori possono organizzare e strutturare la conoscenza della macchina, permettendole di eseguire attività come:

- sintetizzazione automatica dei concetti;
- traduzione automatica delle varie lingue da lui conosciute;
- analizzare i sentimenti;
- riconoscimento vocale;
- segmentazione e ristrutturazione degli argomenti per una migliore comprensione.

> *"Oltre alle comuni operazioni di word processor che, trattano il testo come una semplice sequenza di simboli, la NLP considera la struttura gerarchica del linguaggio: parecchie parole formano una frase, più frasi formano un discorso, un discorso forma una sentenza e, in fine, una sentenza esprime un'idea."*

Scrisse nel suo blog, John Rehling, ex ingegnere di software e esperto di NLP per Meltwater Gruop. La NLP è usata per analizzare il testo, permettendo alla macchina di assimilare ciò che l'uomo sta dicendo.

Oggigiorno gran parte degli algoritmi del linguaggio naturale sono basati su tecniche di machine learning, il che permette di non codificare manualmente grandi serie di regole, ma di avvalersi dell'apprendimento della macchina per imparare automaticamente queste regole analizzando un insieme di esempi e facendo un analisi statica di questi. [6]

Tuttavia, rappresenta ancora un processo difficile dalle caratteristiche intrinseche, dovute alle particolari ambiguità del linguaggio umano. Per questo il processo di elaborazione viene generalmente suddiviso in quattro fasi, simili al processo di elaborazione di un linguaggio di programmazione.

1. **Analisi lessicale**: scomposizione di un espressione linguistica in token di parole;

2. **Analisi grammaticale**: associazione delle parti del discorso a ciascuna parola nel testo;

3. **Analisi sintattica**: arrangiamento dei token di parole in una struttura sintattica;

4. **Analisi semantica**: assegnazione di un significato alla struttura sintattica e, equivalentemente, all'espressione linguistica. Ovvero il processo di *disambiguazione*.

1.3 Riconoscimento vocale

Il riconoscimento vocale (*speech recognition*) è un processo mediante il quale il linguaggio orale viene riconosciuto e, successivamente elaborato, da un apposito sistema di riconoscimento vocale sulla base di specifiche esigenze. Questi sistemi vengono impiegati per una vasta gamma di applicazioni, in quanto atti ad un facile utilizzo, naturale e che dovrebbe possedere un'interfaccia vocale flessibile in modo tale da permettere a chiunque di lavorare con

una macchina.
A questo scopo, dopo l'analisi dell'onda sonora della voce, la parola può essere elaborata attraverso programmi che tramite particolari algoritmi sono in grado di ricostruire il discorso seguendo la frequenza delle associazioni tra le parole, sulla base di un vocabolario fonetico inserito in un database di conoscenza, dove poi viene decodificata. Metodo tuttavia poco affidabile in quanto il segnale, oltre a essere diverso per ogni interlocutore, è suggestionato dal trasduttore impiegato per catturarlo, dal canale usato per trasmetterlo e dal rumore che può essere presente nell'ambiente. Oppure, un miglior approccio [11], si basa sulla manipolazione del segnale come un modello stocastico che, attraverso calcoli di probabilità, riesce a decifrare se la sequenza acustica risulta corretta.
Il riconoscimento vocale [11] viene diviso in tre livelli [Figura 1.2]

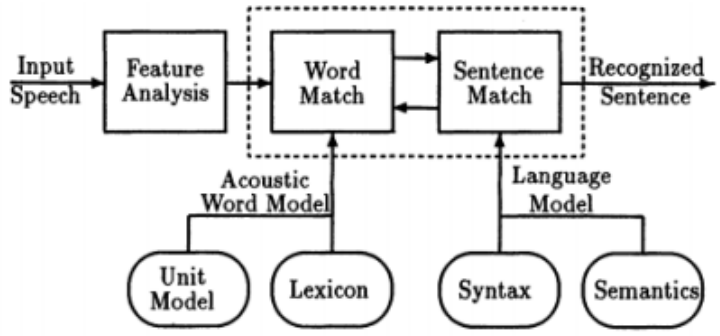

Figura 1.2: I 3 livelli di Speech Recognition [11]

- Livello 1: riceve il segnale e dà origine ad un vettore contenente le caratteristiche acustiche. Questo verrà usato per definire le proprietà spettrali del suono grazie all'analisi di Fourier, ovvero una rappresentazione di una funzione periodica mediante una combinazione lineare di funzioni sinusoidali (seno e coseno)

- Livello 2: valuta il matching tra la sequenza in input e le parole contenute nel vocabolario.

- Livello 3: in base ad un modello di linguaggio ottimale, sintattico e semantico, stabilisce quale frase viene pronunciata con maggior probabilità.

1.3.1 Deep learning

Il *deep learning* è un ramo dell'apprendimento automatico e dell'intelligenza artificiale che si incentra su un insieme di algoritmi che cercano di modellare strutture complesse o diverse trasformazioni non lineari, di concetti ad alto livello tramite l'apprendimento di quelli di basso livello, sfruttando diversi livelli di elaborazione incentrati in gerarchie di concetti o fattori.
Molti algoritmi di deep learning vengono applicati a attività di apprendimento senza supervisione (Unsupervised learning), ovvero deducendo una funzione per descriverne poi la struttura nascosta dai dati non "etichettati". Questo rappresenta un'importante beneficio, in quanto i dati non etichettati sono di solito più ridondanti dei dati etichettati. Tra le architetture interessanti troviamo:

1. Deep Neural Network (DNN): rete neurale artificiale composta da più livelli nascosti di input e output. Possono modellare relazioni non lineari complesse. Qui può essere applicato il riconoscimento vocale e l'elaborazione del linguaggio naturale.

2. Recurrent Neural Network (RNN): a livello di input "impara" a prevedere il successivo input sulla base della cronologia di immissioni precedenti. In questo modo le connessioni tra le unità formano un ciclo diretto, creando così uno stato interno della rete che consente un comportamento dinamico temporale.

Le RNN, a differenza delle DNN, possono usare la memoria interna per processare in modo arbitrario sequenze in input. Questo le rende idonee per il riconoscimento vocale. [20]

1.3.2 RNN per il riconoscimento vocale

Come dimostrato [11] le reti neurali profonde (RNN) Long Short-Term Memory (LSTM) presentano prestazioni migliori delle reti neurali profonde (DNN) feed-forward come modelli acustici per il riconoscimento del parlato sul larga scala. LSTM è una particolare architettura RNN con propensione nel processare, classificare e predire serie temporali. Le RNN modellano le serie in input in maniera unidirezionale e bidirezionale. Nelle reti unidirezionali [Figura 1.3 (in alto)] viene stimano l'etichetta a posteriori, usando solo il contesto sinistro dell'input corrente e processandolo quindi sinistra verso destra avendo uno stato nascosto nella direzione in avanti. Questo è preferibile nella applicazioni che richiedono un basso livello di latenza tra gli input e gli output, migliorandone l'accuratezza di classificazione.
Se invece ci si può permettere una latenza, le reti RNN bidirezionali [Figura 1.3 (in basso)] stimano le etichette a posteriori impiegando stati separati per per processare gli input nelle due direzioni.

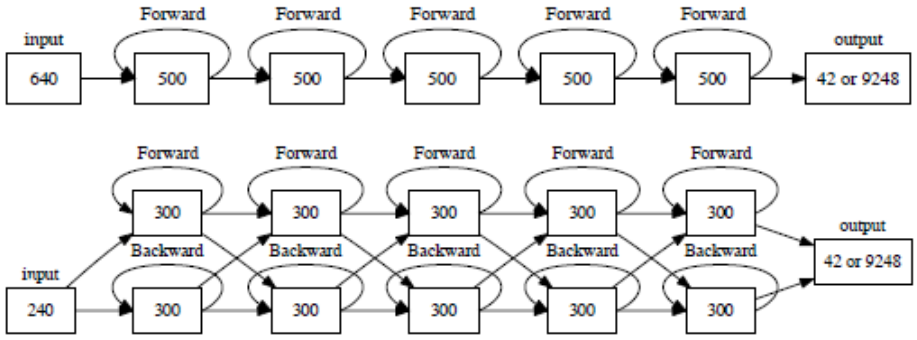

Figura 1.3: Connessione tra gli stati nelle RNN LSTM a 5 stati di tipo unidirezionale (in alto) e bidirezionale (in basso) [11]

In [11] viene studiato un approccio basato sull'architettura LSTM RNN, costituita con la sovrapposizione di più livelli LSTM, e dimostra che il riconoscimento vocale viene svolto in modo migliore rispetto ad altri modelli preesistenti. In particolare i modelli bidirezionali utilizzano due livelli LSTM

per ogni profondità collegati fra loro e, quindi l'output è collegato alla parte finale di entrambi i livelli in ambedue le direzioni.

L'addestramento Connectionist-Temporal-Classification (CTC) è una tecnica di etichettatura delle sequenze RNN in cui non si conosce l'allineamento tra etichette input ed etichette target (o destinazione). Questo può essere implementato con uno stato output che sfrutta un'unità addizionale per le etichette "blank", usata poi per stimare la probabilità che entro un certo slot di tempo non sia stata emessa nessuna etichetta. La probabilità di etichetta output della rete definiscono una distribuzione di probabilità per tutte le possibili etichettature delle sequenze in input incluse quelle di tipo blank. La rete può quindi essere implementata in modo da ottimizzare la probabilità totale di correttezza che, tramite gli output e algoritmi di tipo forward-backward definisce l'etichettatura. Una corretta etichettatura per una sequenza in input è data dall'insieme delle possibili etichette (nella giusta sequenza e consecutive) ed eventuali ripetizioni o sempre dall'insieme delle possibili etichette separate ed inserimento di etichette blank fra le due, per evitare perdita di frame in caso di incertezza. [11]
Quindi il criterio CTC è valido per il Word Error Rate (WER), ovvero nel minimizzare la percentuale di parole non riconosciute nel riconoscimento automatico del parlato.

Per migliorare la precisione dei modelli acustici RNN inizializzati con il criterio CTC, che risulta poco stabile a causa dalla mancata convergenza di alcune esecuzioni, si utilizza il criterio Sequence Discriminative training che incorpora il lessico e i vincoli del modello di linguaggio usati nel discorso decodificato. Dopo Sequence Discriminative training la differenza tra CTC e un modello convenzionale è quello dell'uso del simbolo blank. CTC utilizza modelli indipendenti dal contesto, ma è noto che gli stati dipendenti dal contesto rendono più performanti i sistemi di riconoscimento vocale. La dipendenza dal contesto è un vincolo importante per la decodifica che fornisce una vantaggiosa etichettatura per gli stati dell'output. Quindi la combinazione della memoria LSTM RNN e le abilità del modello CTC ad allineare l'etichetta e le

sequenze di fotogrammi acustici consente l'uso di unità di modellazione con maggiore durata, alleggerendo la rete del compito di etichettare ogni frame con l'introduzione dell'etichetta blank [11].

Capitolo 2

Piattaforma robotica NAO

Nao, robot umanoide di taglia media, frutto di un'unica combinazione tra ingegneria meccanica e sviluppo software, è composto da una molteplicità di sensori, motori e software pilotati da un sistema operativo fatto su misura: *NAOqi OS*.
Il progetto (Project Nao) fu lanciato nel 2004 da Bruno Maisonnier, che fondò, durante l'anno successivo, l'azienda robotica *Aldebaran Robotics* con sede principale a Parigi, in Francia. Nel 2008 l'azienda decise di lanciare la *NAO Academics Edition*, destinata alle università ed ai laboratori di ricerca per scopi di istruzione e ricerca; mentre dal 2014 è disponibile Nao Evolution. Ad oggi, secondo le ultime stime effettuate dalla società stessa [14], sono in uso oltre 5.000 unità Nao in circa 70 paesi diversi. Dal 2013, inoltre Aldebaran è una società del gruppo *SoftBank* che detiene il 78,5% del suo capitale.

2.1 Specifiche

Nao [12] è alto 57.3 cm, largo 27.3 cm e pesa 4.3 kg. Il corpo è realizzato con uno speciale materiale plastico ed ha una batteria agli ioni di litio con autonomia stimata di 90 minuti.

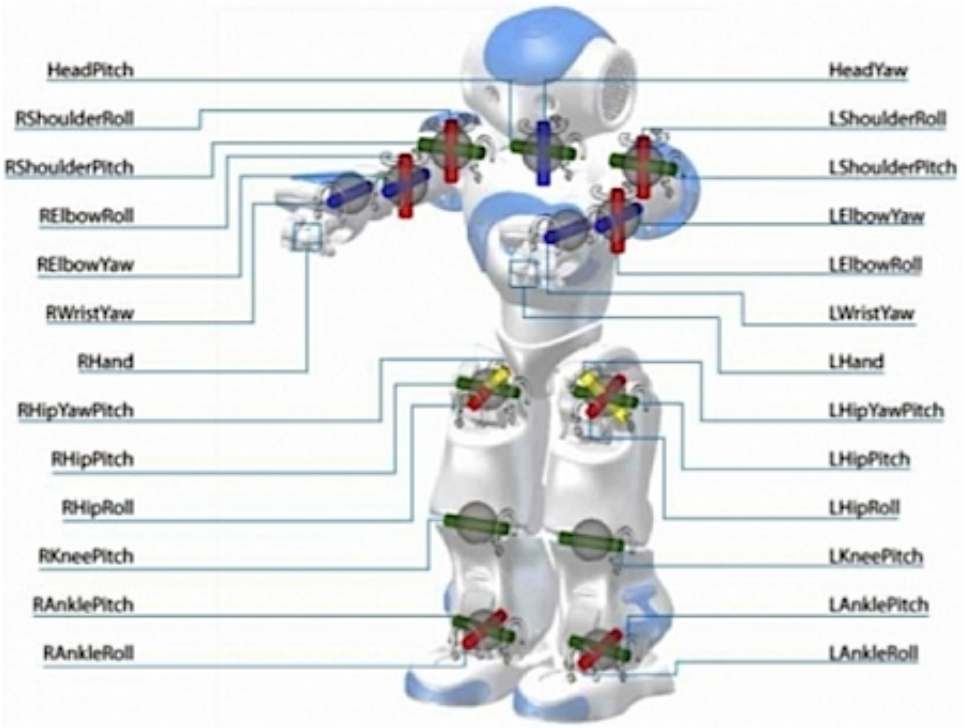

Figura 2.1: Specifiche Nao

Dispone [12] [Figura 2.1] di una centrale inerziale a cinque assi che gli gli permette di capire in quale posizione si trova, di sensori di prossimità a ultrasuoni rivolti in diverse direzioni e di sensori di pressione sotto i piedi. Inoltre, è dotato di un sistema multimediale composto da: quattro microfoni, due altoparlanti e due video camere CMOS, per la sintesi vocale, la localizzazione nello spazio e, per il riconoscimento facciale e degli oggetti. Possiede anche dei sensori di interazione come, tre zone tattili al disopra della testa, due LED infrarossi, due sensori di contatto respingenti nella parte anteriore dei piedi e due sensori touch ai lati di ogni mano.

Viene distribuito con un software di programmazione e di manipolazione visuale, *Choregraphe* [Figura 2.2], sviluppato specificamente per lui; é possibile, inoltre, programmarlo in Python [15] e C++. Ulteriormente, sono incluse nella sua SDK: webots, che permette di testare i programmi realizzati con choregraphe prima di istallarli all'interno del robot; e monitor, applicazione

per valutare come il robot vede e percepisce gli stimoli esterni e monitorare i dati provenienti dai sensori.

Il sistema operativo integrato in Nao è Linux, ma è comunque compatibile con Mac OS e Windows.

Possiede connettività a Internet mediante protocollo Ethernet oppure mediante la tecnologia Wi-Fi.

Grazie ai suoi sette sensi gli è permessa una "naturale" interazione in merito a: movimento, percezione dello spazio, udito e parlato, vista, connessione, pensiero.

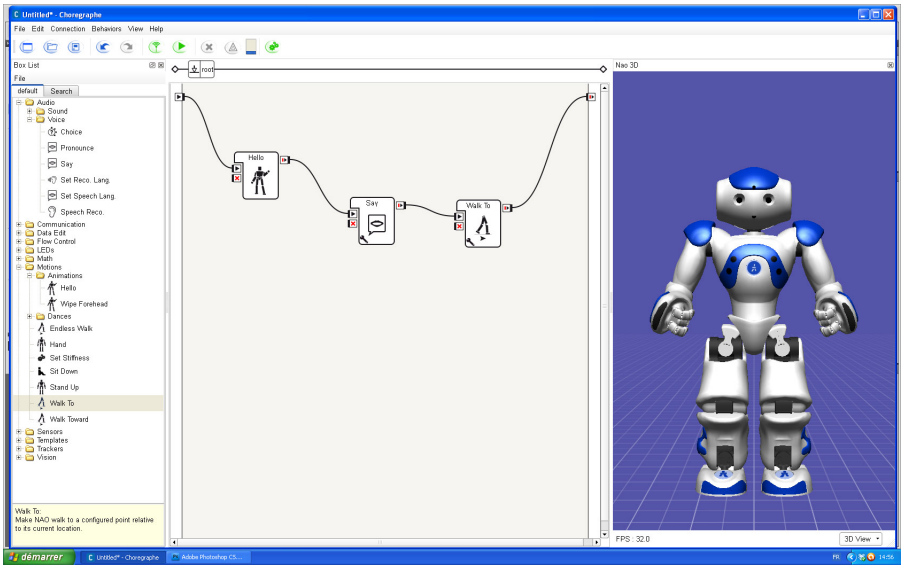

Figura 2.2: Esempio progettazione Choregraphe

Possiede 25 gradi di libertà *(Degrees of Freedom - DOF)*, parametri utilizzati per capire quanto i movimenti di un robot siano sofisticati; ovviamente più il numero è altro più il robot possiede possibilità di movimento evolute. Nao infatti cammina in tutte le direzioni ed evita gli ostacoli, sia fissi sia in movimento; se "perde l'equilibrio" è capace di rialzarsi da solo. Riconosce volti, oggetti e voci ed è capace di dialogare in ben ventuno lingue, con gradi di conoscenza diversi. Ed è grazie a tutte queste abilità che lo rendono uno fra i migliori social robot (o robot da compagnia) esistenti fino ad oggi.

2.2 NAOqi OS

NAOqi OS è un sistema operativo creato su misura per il robot, basato su Gentoo, distribuzione source-based di GNU/Linux, che prevede la compilazione di programmi mediante codici sorgenti, piuttosto che attraverso pacchetti precompilati [9]. Questa particolarità permette di ottimizzare il software basandosi sull'hardware utilizzato e configurandolo secondo le proprie esigenze.
Questo sistema dà al robot la sua personalità di base e gli permette di "prendere vita" dopo l'accensione dello stesso, attivando così all'interno del proprio ambiente i sensori per l'ascolto e per il movimento.
Il sistema attuale [12] si rifà alla release 2.1 dove il processo che va in run sul robot è un *broker*, ovvero un oggetto che fornisce:

- servizi di directory, consentendo di trovare moduli e metodi
- network access, consentendo ai metodi di essere chiamati al di fuori del processo.

Quindi in fase di avvio parte un processo che carica un file AUTOLOAD.INI, il quale definisce quali librerie devono essere caricate, dove ogni libreria contiene uno o più moduli, che mediante il broker, rendono pubblici i loro metodi.

Risulta molto interessante l'esecuzione dei programmi, che può essere eseguita da computer e processata dal robot ma, è anche possibile eseguire i programmi direttamente sul robot stesso, rendendolo del tutto autonomo da comandi esterni.

2.3 Applicazioni esistenti

Molte sono le applicazioni già prodotte nel corso degli anni, che vede Nao attore principale di svariate mansioni. Fra le più importanti troviamo:

- **Insegnante**: progetto partito in Francia nel 2016, dove il robot si affianca ai docente di informatica delle scuole primarie e secondarie.

- **Receptionist**: aiuto clienti nella catena alberghiera Hilton negli Stati Uniti. Il nome del robot viene cambiato in Connie ed è una versione più "intelligente" dello stesso, questo grazie ad un software redatto da IBM, Watson, il quale migliora la sua capacità di capire il linguaggio umano, così può comprendere meglio le domande che gli vengono poste e interagire meglio con i clienti dell'albergo. Quindi Connie può dare informazioni sull'hotel nel quale si sta soggiornando ma anche fornire una serie di indicazioni su centri di informazione turistica, ristoranti, collegandosi via Internet con la piattaforma di viaggi WayBlazer.

- **RoboCup**: dal 2008 nao partecipa alla competizione robotica RoboCup, con lo scopo di riunire diversi team di programmatori da tutto il mondo, ampliando e consolidando le potenzialità del robor per sfidarsi su un mini campo da calcio.

- **Ballerino**: spettacolo di ballo della coreografa Bianca Li con l'intento di interfacciare le macchine agli essere umani, in cui a danzare sono otto ballerini e sei robot umanoidi.

- **ASK Nao**: acronimo di Autism Solution for Kids, è un software disponibile per Nao, elaborato in collaborazione con il Centro autismo dell'Università di Birmingham. Nao stimola l'attenzione di bambini affetti da autismo con una serie di giochi ed esercizi didattici sullo sviluppo della comunicazione verbale e non. Secondo quanto affermato da Ian Lowe, insegnante di una scuola di Birmingham che ha utilizzato Ask Nao in via sperimentale:

  > *<<I robot interagiscono meglio con i bambini autistici perché "non hanno emozioni", sono "meno minacciosi" e più carismatici. Con loro, i tempi di reazione si accorciano e instaurare un dialogo diventa più facile.>>*

Capitolo 3

Nao SDK dialogo e Audio

A livello hardware Nao è dotato di un sistema di trasmissione stereo composto da due altoparlanti nelle sue orecchie e da quattro microfoni, posti sui quattro lati della faccia, con una banda di: [300Hz - 8kHz] [12]. A livello software, invece, la gestione audio del robot è organizzata da NAOqi Audio, che ha il compito di gestire il suono, di localizzarlo e rilevarlo e di gestire il linguaggio. Queste rappresentano le componenti che permettono l'effettivo interfacciamento del robot con gli umani.

3.1 ALAudioDevice

Il modulo ALAudioDevice, appartenente alla categoria di gestione audio in NAOqi Audio, fornisce le API necessarie agli altri moduli per accedere agli input audio tramite microfono e, agli output audio tramite altoparlanti. Questo si basa sulla biblioteca standard Linux ALSA (Advanced Linux Sound Library), la quale fornisce funzionalità audio e MIDI (Musical Instrument Digital Interface) al sistema operativo Linux permettendo un supporto efficiente per tutti i tipi di interfacce audio, quindi sfruttata per comunicare con il driver audio di Nao e, successivamente per comunicare con i microfoni e gli altoparlanti. Qualsiasi modulo NAOqi può inviare dati agli altoparlanti del robot tramite una chiamata; mentre per quanto riguarda i dati prove-

nienti dai microfoni possono essere processati solo dai moduli che elaborano correttamente tali dati, ovvero che rispettano determinati range richiesti.

Appartengono alla categoria di gestione del suono NAOqi Audio anche il modulo ALAudioPlayer che permette di riprodurre file audio sul robot e, ALAudioRecorder che permette di registrare file audio in real-time.

3.2 ALSpeechRecognition

Il modulo **ALSpeechRecognition**, appartenente alla categoria di gestione del linguaggio in NAOqi Audio, consente al robot di riconoscere parole e frasi predefinite in diversi linguaggi. Esso si basa su sofisticate tecnologie di riconoscimento vocale fornite da:

- ACAPELA GROUP per Nao versione 3.x, inspiring provider che creano soluzioni "voci" capaci di leggere, informare, guidare, educare, raccontare storie, aiutare e comunicare, allarmare, informare ed intrattenere. Create e sviluppate per fornire un risultato sonoro piacevole ed intelligente.[10]

- NUANCE per Nao versione 4. Tecnologia embedded cloud-based e linguaggio espressivo di text-to-speech attraverso programmi di sviluppo NDEV. Questo permette al robot di avere conversazioni più naturali ed espressive con le persone, consentendo una voce custom-to-speech personalizzata. [5]

3.2.1 Principio operativo

Il funzionamento di ALSpeechRecognition si suddivide in quattro step [12]:

- A - prima di iniziare, il modulo deve essere alimentato dall'elenco delle frasi che devono essere riconosciute

- B - una volta iniziato, il modulo pone nella chiave *SpeechDetected* un valore booleano che specifica se l'altoparlante è attualmente attivo o meno.

- C - se l'altoparlante è attivo e viene sentito qualcosa, l'elemento dell'elenco che meglio corrisponde a quello ascoltato dal robot, viene inserito nella chiave **WordRecognized** e gli viene attribuito un grado di confidenza [Figura 3.1].

```
[phrase_1, confidence_1, phrase_2, confidence_2, ..., frase_n, confidence_n]
```

Figura 3.1: Struttura WordRecognized

- D - se l'altoparlante è attivo, l'elemento dell'elenco che meglio corrisponde a quello ascoltato dal robot, viene inserito nella chiave **WordRecognizedAndGrammar** ed, oltre ad essergli attribuito un grado di confidenza, viene specificata la grammatica utilizzata [Figura 3.2].

```
[frase_1, confidenza_1, grammatica_1, frase_2, fiducia2, grammatica_2, ...,
```

Figura 3.2: Struttura WordRecognizedAndGrammar

Appartengono alla categoria di gestione del linguaggio NAOqi Audio anche il modulo ALTextToSpeech il quale permette al robot di parlare, ALAnimatedSpeeh che combina discorso con gesti, ALDialog per creare un database base di conoscenza e ALVoiceEmotionAnalysis che permette al robot di identificare le emozioni espresse dalla voce ascoltata.

3.3 ALDialog

Il modulo *AlDialog* permette di creare un database di conoscenza per dotare Nao con abilità di conversazione, utilizzando un elenco di "regole" scritte e classificate in modo appropriato. Queste regole permettono di gestire il flusso

della conversazione tra l'uomo e il robot e, sono essenzialmente di due tipi: regole utente e regole di proposta [12].

- una regola utente collega un input utente specifico all'eventuale output del robot [Listing 3.1].

```
# regola utente
u:(Ciao Nao come stai oggi?) Ciao uomo, sto bene
                                grazie e tu?
```

Listing 3.1: Regole utente

- una regola di proposta innesca un'uscita specifica del robot senza alcun "anticipo" da parte dell'utente [Listing 3.2]

```
# regola di proposta
proposal: Hai visto quel ragazzo in TV ieri?
      u1: (si) Mi sembrava un pazzo, non credi?
      u1: (no) Mi dispiace ma non capisco
```

Listing 3.2: Regole di proposta

Per rendere le regole più chiare e la gestione più corretta, queste vengono raggruppare in argomenti, chiamati nel gergo *Topic*, solitamente figurato tramite uno script box o un file nel seguente modo [Listing 3.3]:

```
# header del topic contenente il nome e la lingua
topic: ~greetings
language: it

 # regola utente
u:(Ciao Nao come stai oggi) Ciao uomo, sto bene
                              grazie e tu?
```

```
# regola di proposta
 proposal: Hai visto quel ragazzo in TV ieri?
        u1: (si) Mi sembrava un pazzo, non credi?
        u1: (no) Mi dispiace ma non capisco
```

Listing 3.3: Esempio di un semplice topic

Questo modulo può essere integrato con un insieme di regole più complesse, QiChat, che seguono sintassi specifiche.

ALDialog utilizza il modulo ALMemory per memorizzare e recuperare i dati che poi potranno essere visti come variabili o eventi che scatenano il verificarsi di determinate regole e/o azioni.

3.4 Introduzione a QiChat

QiChat rappresenta un'insieme di regole più complesse che possono essere introdotte all'interno del database di conoscenza di Nao tramite i topic. Le regole associano ad un input umano, ovvero ciò che viene detto dall'uomo, con un pertinente output del robot. Tramite delimitatori, caratteri speciali, funzioni di regola e proprietà è possibile creare potenti regole di linguaggio [12]. QiChat permette di definire tre tipi di regole: user rule, user subrule e proposte. L'input umano, delimitato tra parentesi tonde ( ) quindi contiene il messaggio che deve essere compreso e riconosciuto dal robot; quando l'input è compreso allora la regola ad esso associata viene innescata e, di conseguenza viene eseguito l'output del robot, incentrando il focus sull'argomento (topic) contenente la relativa regola. Mentre l'output del robot è parte di una regola che specifica ciò che il robot deve dire e/o fare quando questa viene attivata. Importante è definire il concetto *(concept)*, ovvero una lista di parole e/o frasi che si riferiscono ad una idea. Questi possono essere inseriti sia nell'input umano sia nel output del robot e possono essere di tipo statico oppure dinamico [12].

Fanno parte dei concetti statici tutti quei concetti la cui visibilità è globale all'interno del robot e che non possono essere modificati a run-time (es. parti opzionali e funzioni) .

```
concept:(volere) ^rand{``io *" ``io vorrei
                        {``un po' di "}"}
```

Listing 3.4: Concetti statici

Invece i concetti dinamici sono formati da una lista di parole e/o frasi contenute all'interno di una singola scelta (choice [ ]). Questi hanno una visibilità locale all'interno del topic e, possono essere modificati a run-time.

```
dynamic: mp3
u:(``ascoltiamo" [``vorrei cantare"] _~ mp3)
                                ok, ora cerco $1
```

Listing 3.5: Concetti dinamici

```
proxy=ALProxy(``ALDialog")
proxy.setConcept(``mp3",``it",[``pink floyd",
                                ``jimi hendrix"])
```

Listing 3.6: Richiamo in python topic di concetti dinamici

Sia regole che concetti necessitano di attivazione per essere presi in considerazione e, anche se è possibile attivare diversi topic contemporaneamente, solo uno di questi detiene il focus, ovvero l'attenzione primaria da parte del robot. Per questo viene seguito un ordine di priorità in modo da non creare collisioni fra le regole:

- focus: il solo topic che in un determinato istante detiene l'attenzione;
- tutti i topic attivi;
- recover: sezione di ricovero, contenente tutte le regole che voglio essere utilizzate in caso di output errato da parte del robot;

- fallback: tag che definisce la minor priorità della regola all'interno del topic e quindi deve essere attivata per ultima.

Il progetto di questa tesi si è prefissato come scopo finale quello di comporre un database di conoscenza applicando tutte le regole rese disponibili dal modulo ALDialog integrando i singoli topic con le regole analizzate nel modulo QiChat.

Capitolo 4

Applicazione: Dialog with NAO

In questa sezione andiamo ad analizzare le metodologie mediante le quali si è deciso di ampliare il dizionario di conoscenza, implementando una serie di regole scritte all'interno del database del robot servendosi del modulo ALDialog e delle regole definite in QiChat. Per comprendere appieno l'analisi fatta, il capitolo inizia focalizzandosi sull'applicazione sviluppata durante il tirocinio presso il laboratorio DISI della sede del corso di Ingegneria e Scienze Informatiche. Per poi passare ad un descrizione di tutti gli aspetti dell'applicazione di dialogo in questione: requisiti, design e sviluppi futuri. Si è deciso di chiamare l'applicazione *"DIALOG WITH NAO"*, per dare in idea della stessa.

4.1 Analisi del problema

L'obiettivo raggiunto consisteva nello sviluppare una applicazione dimostrativa per la piattaforma robotica Nao Aldebaran, che permettesse ad un utente medio di interfacciarsi con il robot in maniera del tutto autonoma. In particolare, inizialmente il robot rimane seduto in attesa di ricevere un input di tipo touch nella tecla posta sulla testa, a seguito di questo evento il robot si alza e si presenta, attivando il riconoscimento facciale, rivolte a persone a lui note e ricercando un viso qualsiasi attorno a lui; parallelamente attiva il microfono

di ascolto in attesa di un input vocale da parte dell'utente per poter avviare una conversazione e/o dimostrare una delle varie cose che "sa fare". L'applicazione è stata progettata sfruttando la piattaforma Eclipse Neon mediante l'utilizzo dell' IDE pyDev e python SDK, fornite dal produttore, seguendo il paradigma stati-eventi

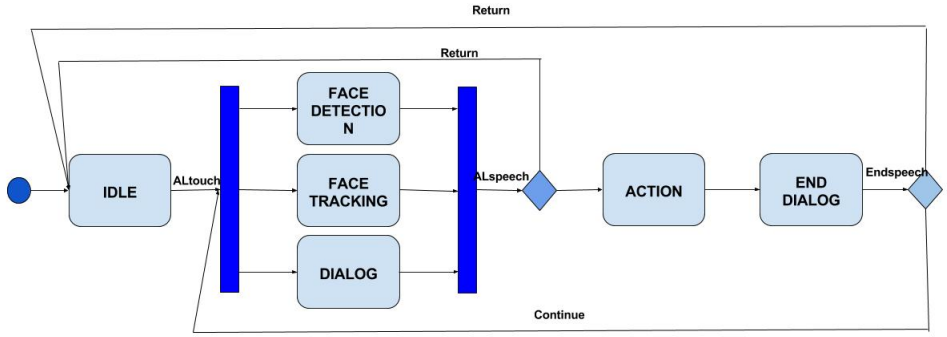

Figura 4.1: Diagramma stati-eventi di una applicazione su Nao

Il diagramma [Figura 4.1] illustra i vari stati che saranno attivi nel robot a seguito di un determinato evento.
Stati:

1. IDLE: il robot si trova in posizione di rest in attesa di ricevere l'evento di ALtouch;

2. DIALOG: viene attivato il modulo di speechRecognition che rende possibile il dialogo utente-robot seguendo un dizionario impostato dinamicamente;

3. FACE DETECTION: attiva il modulo di ALFaceDetection che rende possibile il riconoscimento da parte di NAO di persone i cui dati metrici (distanza naso, occhi, bocca) sono preventivamente salvati all'interno del suo database;

4. FACE TRACKING: attiva il modulo di ALTracker, così facendo il robot ricerca con la testa un viso o un oggetto da seguire;

5. ACTION: a seguito dell'evento di ALspeech in questo stato il robot esegue un'attività richiesta: balla la macarena, esegue una presentazione di tai-chi, prende un oggetto qualsiasi nella mano e lo esamina, ti porge la mano per camminare con te;

6. END DIALOG: Nao chiede un riscontro sull'attività appena svolta e si rimette in ascolto per potersi nuovamente interfacciare con l'umano.

Eventi:

1. ALtouch: l'utente deve toccare uno dei sensori di interazione presenti nella testa del robot. A seguito di questo Nao si alza in piedi, si presenta ed attiva gli stati DIALOG, FACE DETECTION e FACE TRACKING;

2. ALspeech: l'utente richiede, tramite comandi vocali, che Nao esegua una delle azioni che conosce;

3. Endspeech: attivato alla terminazione dello stato di END DIALOG, il robot si mette in stato di waiting aspettando un risconto di gradimento da parte dell'umano.

4. Continue: l'utente richiede di avere ancora un interazione con il robot, quindi si riattivano gli stati di DIALOG, FACE TRACKING e FACE DETECTION;

5. Return: Nao ritorna allo stato di IDLE a seguito di una richiesta vocale di terminazione da parte dell'utente.

In questo caso, il dialogo è stato implementato sfruttando i moduli ALSpeechRecognition e ALTextToSpeech che danno al robot la possibilità di capire un numero limitato e specifico di parole e/o frasi inoltre segue una metodologia di sviluppo poco flessibile e presenta un enorme mole di codice. Focalizzando la nostra attenzione sul dialogo, riscontriamo che il dizionario

risulta privo di regole discorsive e presenta una struttura fortemente statica. Dando uno sguardo ad un esempio di codice [Listing 4.1] questo può essere subito notato.

```
thingsAbleToDo_user = ["Cosa sai fare", "fare",
                       "cosa", "capace"]
thingsAbleToDo_nao = ["Direi che me la cavo bene con: "
                      "il saggio di tai-ci, "
                      "ballare la macarena, "
                      "camminare insieme a te, "
                      "e infine posso afferrare
                      oggetti. "
                      "Vuoi vedere qualcosa?"]
```

Listing 4.1: Dialogo implementato con ALSpeechRecognition e ALTextToSpeech

Una volta implementato questo "vocabolario" dovrà essere richiamato all'interno del codice tramite una funzione che permetta di accedere ad ogni definizione. Per questo ci si è chiesti se era possibile sviluppare un applicazione che permettesse di:

- implementare un dizionario di conoscenza base per il robot;
- fissare dei "concetti" specifici ai quali rifarsi per seguire delle regole generali;
- valutare la distribuzione dei focus specifici relativi ad argomenti diversi;
- effettuare un switch di contesto in tempi ragionevoli

4.1.1 Requisiti

La modellazione dei requisiti funzionali dell'applicazione viene rappresentata attraverso il diagramma degli casi d'uso. Si tratta di un diagramma che esprime un comportamento, offerto o desiderato, sulla base dei suoi risultati

osservabili; l'oggetto esaminato è solitamente un sistema o una sua parte. Esso infatti individua chi o che cosa ha a che fare con il sistema, ovvero l'attore e, che cosa viene fatto, il caso d'uso.

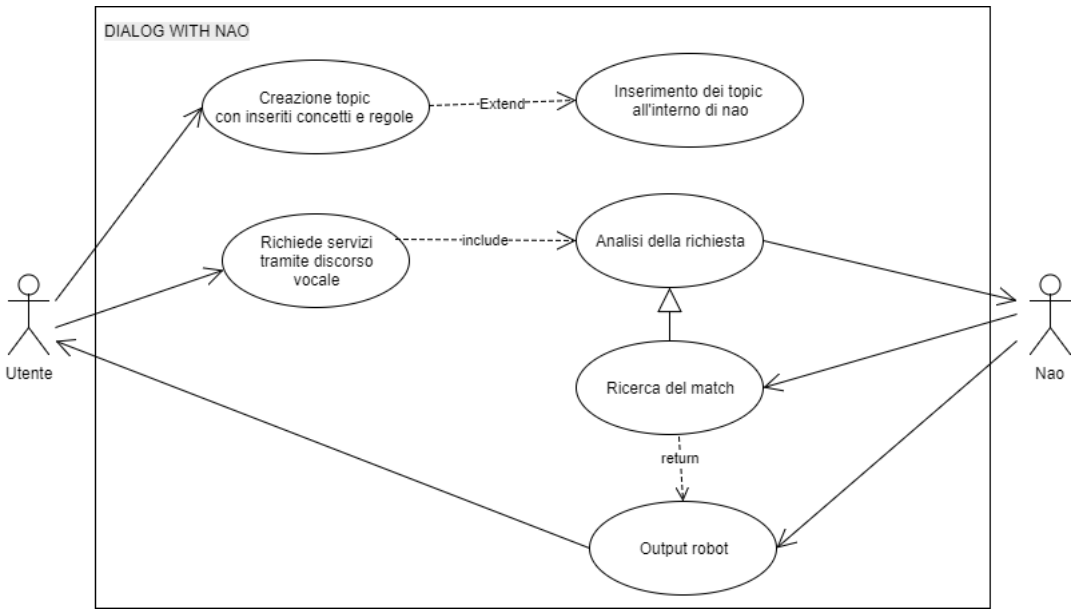

Figura 4.2: Diagramma dei casi d'uso - DIALOG WITH ME

Troviamo [Figura 4.2] due attori, nel nostro caso l'utente e il robot che hanno lo scopo di comunicare tra di loro. Come si nota nel diagramma, tutti i casi d'uso sono legati attraverso associazione all'utente, il solo in grado di attuarli, tranne l'output del robot, che sarà appunto attivato dal robot e ritornerà la risposta a ciò che gli era stato precedentemente richiesto. Invece i casi d'uso sono legati fra loro attraverso «include» sottointendendo una dipendenza tra i casi d'uso, implicando che un caso faccia parte al comportamento di quello che lo include e, inoltre l'inclusione non è una dipendenza opzionale. L'utente infatti ha il compito di creare ed inserire i propri topic all'interno del robot così poi da permettere tutte le funzionalità ed è per questo che le inclusioni, inserite nel diagramma, convergono su quel caso d'uso specifico. Una volta inserito il topic all'interno del robot, l'utente può richiede servizi tramite comandi vocali al robot. Questi comandi saranno analizza-

ti dal robot e, successivamente, dopo aver trovato il match corrispondente, ritornerà l'output corretto in base alla domanda postagli.

4.2 Design dell'applicazione

Per quanto riguarda il design dell'applicazione è stato sfruttata l'IDE Eclipse Neon [8], ambiente di sviluppo integrato multi-linguaggio e multipiattaforma per la gestione dei topic e per quanto concerne l'aspetto funzionale. Mentre il cuore dell'applicazione, ovvero la stesura del database di conoscenza, si è sfruttato un semplice editor di testo per creare file con estensione .top, documento di progettazione TopSolid Design, sono file di dati utilizzati per memorizzare oggetti CAD e disegni con una o più sezioni.

4.2.1 Implementazione

La tecnologia base dell'implementazione della logica dell'applicazione consiste, come prima preannunciato, nell'utilizzo di file con estensione .top. Non sono serviti applicativi per poter eseguire la stesura del programma, ci si è serviti solamente dell'editor di testo Notepad++. Mentre, per quanto riguarda l'esecuzione di questi file si è reso necessario l'utilizzo di Eclipse, accessoriato di plug-in e strumenti atti a seguire una buona programmazione. L'applicazione può essere eseguita sia in locale, ovvero lanciando il programma Python da computer per rendere possibile l'esecuzione del testo, oppure può essere inserita nel boot del robot e, quindi salvata al suo interno. Procedendo con quest'ultima modalità l'applicazione verrà eseguita quando richiesto senza dover eseguire codice da computer. Si è scelto comunque di lavorare in locale, tale decisione è stata determinata dal fatto che questa applicazione risulta ancora un prototipo, che in futuro può ambire ad ampliamenti e miglioramenti.

Partendo dall'implementazione Eclipse, ricordiamo redatta in Python, permette il caricamento e l'attivazione dei vari topic. Questo può avvenire a seguito della creazione del proxy `ALProxy()`, oggetto che da dà accesso a

tutti i metodi o al modulo a cui ci si si vuole collegare, che prende in input il nome del modulo da utilizzare e i parametri indispensabili per collegarsi al robot: il numero di porta, di default 5995, e l'indirizzo IP del robot, che può cambiare a fronte di nuove connessioni. Quindi in questo contesto bisogna creare un proxy che renda possibile l'acceso al modulo di ALDialog [Listing 4.2].

```
import argparse
from naoqi import ALProxy



def main(robot_ip, robot_port, topf_path):
    dialog_p = ALProxy('ALDialog', robot_ip, robot_port)
    ....
```

Listing 4.2: Dialog with Nao. Creazione ALProxy

Lo step successivo consiste nel settare il linguaggio nella lingua preferita `setLanguage()`. Si ricorda che sul robot è possibile usufruire di due idiomi contemporaneamente, inglese di default più una istallata a piacere, a patto che non siano utilizzati nel medesimo istante; l'unico modo esistente per cambiare lingua in real-time durante l'esecuzione del programma è inserendo un codice di settaggio che attivi la lingua desiderata. Successivamente si può procedere con il caricamento `proxy.loadTopic()` e l'attivazione `proxy.activateTopic` dei topic, poi successivamente in modo analogo questi verranno disattivati, `proxy.deactivateTopic`. Per caricare e gestire il topic, si è deciso di implementare un argomento, in cui viene passato un parametro di tipo str, ovvero stringa, contenente il percorso assoluto del topic che viene posto in una apposita directory e successivamente caricato nel modulo. [Listing 4.3].

```
        ....
    dialog_p.setLanguage("Italian")
```

```
    topf_path = topf_path.decode('utf-8')
    topic = dialog_p.loadTopic(topf_path.
                                encode('utf-8'))
    dialog_p.subscribe('myModule')
    dialog_p.activateTopic(topic)

        ....

    dialog_p.deactivateTopic(topic)
    dialog_p.unloadTopic(topic)
    dialog_p.unsubscribe('myModule')
    ...
```

Listing 4.3: Dialog with Nao. Gestione Topic

Al termine il modulo si chiude, e di conseguenza distrugge tutte le variabili di sessione memorizzate.

```
proxy.unsubscribe('myModule')
```

Esistono diverse metodologie per interfacciarsi con Nao. Principalmente abbiamo usufruito del protocollo Secure SHell (SSH), che permette di stabilire una sessione remota cifrata tramite interfaccia a riga di comando con un altro host di una rete informatica, tramite PuTTY [4]. Oppure, più semplicemente ci si è affidati all'interfaccia user-friendly di FileZilla [7], soluzione File Transfer Protocol (FTP), che è un protocollo per la trasmissione di dati tra host basato su TCP e con architettura di tipo client-server.

Infine definiamo il `main` dove andremo materialmente ad inserire i dati riguardanti gli argomenti da passare alla sezione attiva. Ovvero andremo ad inserire il path assoluto dei topic che vogliamo attivare, la porta e l'indirizzo IP del robot. Questi vengono poi gestiti con un eccezione che verrà lanciata se gli argomenti risultano errati.

4.2.2 I topic

Come già detto, i topic definiscono una serie di regole e concetti che devono essere appresi dal robot. In questa sottosezione andremo ad esplicare nel dettaglio le principali regole definite. In questo progetto si sono redatti tre differenti topic file con lo scopo finale di avere un interazione semplice e il più naturale possibile. Per fare questo abbiamo deciso di stabile con lui un dialogo in cui parlare di musica, più nello specifico di musica rock. Necessaria risulta trascrivere un'intestazione dove definirne il nome e la lingua utilizzata.

`topic: ~ topicName()`

Nel nostro progetto si è deciso di implementare il linguaggio in italiano, anche se ancora grezza a livello fonetico. Fondamentale è risultato l'uso dei conetti, `concept`, per definire un dizionario di possibili combinazioni di frasi e/o parole esprimibili in un determinate situazione. Questi possono apparire sia nell'input utente che nell'output del robot, permettendo quindi una decisione random sul modo di esprimersi. Rendono il dialogo più pratico e funzionale, infatti permettono di snellirne la struttura e, come da definizione, permettono la creazione di concetti e quindi vengono inseriti all'interno del contesto definendo una serie più amplia di possibili soluzioni. Quindi permette di richiamare una variabile piuttosto che definire una frase nuova per situazioni simili. Per apprendere più nel dettaglio ciò che si è appena affermato basti guardare il Listing 4.4.

```
concept:(saluto) ^rand[ciao "ehi salve a voi tutti"
                       buongiorno "{ehila} ciao"]


u:(ciao {nao}) ~ saluto
```

Listing 4.4: Dialog with Nao. Definizione di concetti

Andando avanti con l'analisi, ci si è accorti che è possibile attivare molte animazione, già pre-esistente all'interno del robot tramite una semplice chiamata ad essa, `^ run(animations\type\tagName)`. Oppure intere applicazioni tramite chiamata a switch di contesto, `^ switchFocus(animation/.)`.

Grazie a queste Nao può assumere comportamenti tipici umani, quali in questo caso, "gesticolare" o ballare per ampliare la definizione di un particolare concetto, come il saluto, oppure un'enfasi di felicità quando gli viene detto qualcosa di divertente. Risultando molto più "simpatico" e "vivo".
Notevole è anche la possibilità di definire variabili locali, che permetto al robot di creare associazioni e quindi, memorizzare una determinata parola come può ad esempio essere il nome di una persona e/o cosa. Questa rimarrà memorizzata finché non verrà deallocata tramite funzione `^ clear()`.
Per dare un ulteriore grado di autonomia al robot si sono redatte una serie di regole proposta, che il robot esprime in completa autonomia quando viene specificato nel testo tramite chiamata a `^ nextProposal`. Questa funzione può essere decorata da una chiamata randomizzata a determinate proposte, inserendo al principio di ognuna di queste un tag, % tagName, implicando un jump a determinate proposte. Similmente è possibile far rieseguire al robot la proposta appena menzionata o quella espressa in precedenza, applicando così il concetto di ripetizione o non comprensione tipico dell'atteggiamento umano.
Si è deciso di aggiungere funzionalità che permettono a Nao di riproporre la medesima domanda appena postagli utilizzando una chiamata a dialog, `$ Dialog/LastInput` che quindi, permetterà al robot un ulteriore libera interazione. Analogamente, anche il robot può, richiedere di ripetere nuovamente una nozione non capita, `$ Dialog/notunderstood`.
Interessante è anche la possibilità di scatenare eventi che vengono attivati a seguito dell'attivazione di un determinato argomento e successivamente disattivato quando si cambia argomento. Queste variabili sono memorizzate nel modulo AlMemory, e possono innescare eventi di qualsiasi genere, indipendentemente dal dialogo, quali attivazione di un movimento, attivazione di una brano musicale, ecc.

Funzionalità principali

Questo prototipo, una volta mandato in esecuzione, lascia il robot in stato di ascolto in posizione di rest, ovvero seduto. Una volta iniziata l'interazione con lui questo si alzerà in piedi e attiverà la funzione che gli permetterà di prendere "vita". Iniziando così uno scambio di informazioni con l'utente che intende approcciarsi con lui. L'applicazione permette un processo naturale, ovvero dal momento che il prototipo è stato "lanciato" non necessita di alcun interfacciamento con il computer, ma basta rivolgersi a Nao, tramite comandi vocali per poter continuare la discussione.
Una volta terminato lo script, simpaticamente Nao vi dirà che non vuole più continuare la comunicazione con voi e una volta ricevuta una risposta affermativa da parte dell'utente, ritornerà nella sua posizione tipica di attesa.

4.3 Note di sviluppo

È importante far presente che questa applicazione di dialogo risultata un prototipo e, come tale ha l'obiettivo di far intendere e mettere in pratica i requisiti del sistema. Esso infatti ha lo scopo di trattare un'attività di tipo discorsivo con Nao. Può risultare approssimativo, in quanto, conosciuto l'obiettivo della realizzazione del prototipo, ci si basa maggiormente sui requisiti, piuttosto che sull'efficienza assoluta del sistema.
Come infatti è noto, l'applicazione non presenta un complesso modello del dominio, poiché, in questo stadio di progettazione, non è nel nostro interesse sviluppare algoritmi capaci di apprendere automaticamente ciò che gli viene "insegnato", poiché come si è detto fin dall'inizio l'obiettivo è solo quello di dare ad un utente medio la possibilità di interfacciarsi con il robot, senza che esso necessiti di una particolare conoscenza riguardante linguaggi di programmazione.
Il punto critico dell'applicazione, che deve essere sviluppato in possibili sviluppi futuri, è il modello del dominio dei dati. Il modello attuale non prevede una particolare gestione di database, ma sicuramente è buona norma intro-

durre il concetto di database, Data Base Management System (DBMS) e del linguaggio SQL utilizzato per interagire con lo stesso. Utilizzare un database per la raccolta di informazioni è molto utile ed efficace. Infatti oltre a poter immagazzinare una grande quantità di dati in modo ordinato e persistente, è possibile utilizzarlo con più funzionalità. Quindi l'inserimento di un DBMS permetterebbe il riconoscimento e l'inserimento di determinate parole chiave rendendo in tal modo l'approccio più naturale.
Inolre con lo sviluppo di un modello più consistente si può pensare ad ampliare il concetto di rete semantica sfruttando applicazioni esistenti quali WordNet [19], database lessicale in lingua inglese o BabelNet [2], rete semantica multilingue.

4.4 Corpus principale del codice

```
#header topic
topic: ~introduction()
language: iti

#inclusione topic da richiamare
include response.top
include: music.top

#inserimento dei diversi concetti utili per I/O
#flessibile
concept:(saluto) ^rand[ciao "ehi salve a voi tutti"
                buongiorno "{ehila} ciao"]
....

#Il robor si sveglia alzandosi in piedi e saluta
#alzando la mano destra
u:(ciao {nao}) ^pCall(ALMotion.wakeUp()) ~saluto
```

```
                ^run(animations/Stand/Gestures/Hey_1)

# _* : operazione che permette di registrare il
#nome pronunciato un quella sezione
u:(["mi chiamo" "il mio e'"] nome _*) piacere $name

....

#permette di memorizzare la domanda posta al robot
#in maniera tale anche lui ti rifara' la medesima
#domanda
u:({come} [stai va "tutto bene?"] ) ~stare
                                $Dialog/LastInput

....

#tramite ^nextProposal il robot in autonomia ti pone
#una serie di proposte discorsive decorato da
#^gotoRandom(go) che permette una decisione
#random di tutte le proposte contenenti il tag %go
u:({"non so"} tu di cosa [vuoi "ti va di"] parlare?)
                        ... ^nextProposal
u:(~next) ^gotoRandom(go)

....

proposal: %go che dici di intelligenze artificiali, ..
proposal: %go che dici, di musica

....
```

```
# ^previousProposal permette di far ripetere al robot
#la proposta appena esplicata
u:(prima [cosa] hai detto) ^previousProposal

# %musica tag che permette di collegarsi al topic
#music
u:(["ottima idea" ...) perfetto ^clear(name) %musica

s:(*) ^addword("e tu", end, 1)
```

Listing 4.5: Codice topic introduttivo DIALOG WITH NAO

```
#header topic
topic: ~music()
language: iti

include: response.top

....

u:({che tipo} di musica ti [piace preferisci])
                        ^topicTag(music,musica) ~musica

#il tag ^switchFocus() permette l'esecuzione di un
#programma memorizzato entro nao, in questo caso
#il robot ballera'
u:(~piacere {musica} rock) oh si molto
                ^switchFocus(chacha_music/.)

....

u:(piacere) ascolti sicuramente buona musica
```

```
                ^run(animations/Stand/Gestures/You_4)
u:(!piacere) ~disaccordo


....

#esempio di uso subrules con import topic contenente
#proposte di risposta si/no/forse
proposal: sai come sono nati i pink floyd?
        u1:(~conferma) ...
        u1:(~negazione) ...
                ....
                u2:(~repet) ok ti ripeto ^sameProposal
        ^stayInScope

proposal: sia sono un po' in ritardo al mio appuntamento
        u1:(ok, ciao nao) ciao ^pCall(ALMotion.rest())
        u1:(e:Dialog/NotSpeaking5)) ~essere
        u1:(~repet) ho detto ^sameProposal ^stayInScope
```

Listing 4.6: Codice topic music DIALOG WITH NAO

Conclusioni

La centralità delle macchine intelligenti, l'evoluzione tecnologica e lo studio di nuovi algoritmi di comprensione sono gli elementi fondamentali per il miglioramento dei sempre più crescenti robot umanoidi. È ormai noto quanto questa concezione di macchina sia entrata sempre di più nella vita di tutti i giorni delle persone. Creando grosse aspettative e miglioramenti anche dal punto di vista dello stile di vita.

Questa tesi è stata incentrata sullo sviluppo dell'applicazione DIALOG WITH NAO per permettere a tutti gli utenti di comunicare con NAO tramite interazione vocale.
Lo scopo del nostro progetto è stato quello di far conversare il robot umanoide Nao con il più alto numero di umani possibile. Per un risultato positivo, l'umano non dovrebbe "accorgersi" della differenza tra il robot e una qualsiasi altra persona. Per ovvi motivi, quali le difficoltà fonetiche del robot nell'esprimersi in lingua italiana e la materia intrinseca del Natural Language Processing, questo obbiettivo risulta ancora difficile se non impossibile al giorno d'oggi. Per questo si è riscontrato un esito positivo solo per quanto riguarda il funzionamento base di tale applicazione. Infatti il dialogo con il robot è stato possibile e facendo varie prove si è riscontrato che la percentuale di comprensione si avvicina al 50% circa. Il tempo di risposta da parte del robot risulta sempre molto veloce, a patto che il focus venga inteso.

Il programma è stato sviluppato per tenere un discorso riguardante la musica, ma potrebbe essere utile modificarlo per poterlo poi impiegare in svariati ambienti. Per esempio per impiegarlo sempre di più all'interno delle

scuole per “stuzzicare” la creatività e la curiosità dei più piccoli.

Potrebbe essere anche interessante, ampliare il progetto di tirocinio con una modalità di dialogo di questo tipo, rendendolo così più flessibile e atto a essere utilizzata da chiunque né voglia usufruire.

Bibliografia

[1] Amedeo Cappelli, E. G. L'interazione uomo-robot. *Robocare Technical Report N. 1* (2003).

[2] BabelNet. Babelnet documentation.

[3] Chiari, I. *Introduzione alla linguistica computazionale*. 2007.

[4] comunity, P. Putty documentation.

[5] Dirth, M. Aldebaran robotics and nuance revolutionize human-machine interaction.

[6] Federico Cella, P. O. Machine learning. così i computer diventano intelligenti. *Corriere della sera* (2017).

[7] FileZilla. Filezilla documentation.

[8] foundation, T. E. Python documentation.

[9] Gentoo Foundation, I. Gentoo linux documentation - making the distribution, part 1, 2005.

[10] Group, A. Let't talk.

[11] Has¸im Sak, Andrew Senior, K. R. F. B. Fast and accurate recurrent neural network acoustic models for speech recognition. *" arXiv preprint arXiv:1507.06947* (2015).

[12] Ki-sung, S. *Using Nao: Introduction to Interactive Humanoid Robots.* 2013.

[13] Mauro, T. D. *Grande dizionario italiano dell'uso.* 2000.

[14] Robotics, A. Unveiling of nao evolution: a stronger robot and a more comprehensive operating system, 2014.

[15] software foundation, P. Python documentation.

[16] Stuart Russel, P. N. *Intelligenza artificiale. Un approccio moderno. Volume 2. 2ª edizione.* 2005.

[17] Stuart Russel, P. N. *Intelligenza artificiale. Un approccio moderno. Volume 1. 3ª edizione.* 2010.

[18] Turing, A. Computing machinery and intelligence.

[19] University, P. Wordnet. a lexical database for english.

[20] Wikipedia. Deep learning.