ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso di Laurea in Ingegneria Informatica

TESI DI LAUREA

in

FONDAMENTI DI INFORMATICA T-1

Un'applicazione di elaborazione del linguaggio naturale per il gioco di "Indovina chi?" su robot umanoide

CANDIDATO:

Laura Bosso

RELATORE:

Chiar.ma Prof. Paola Mello

CORRELATORE :

Ing. Thomas Bridi

Anno Accademico 2014/15

Sessione III

Indice

Introduzione 9

1. Il progetto “Indovina chi?” 13

1.1 Il test di Turing 14

1.2 Indovina chi? Il problema 15

2. Riconoscimento vocale 19

2.1 Deep learning 21

2.1.1 RNN per il riconoscimento vocale 21

2.2 Google Speech Recognition 24

3. Il linguaggio naturale 27

3.1 La comunicazione 27

3.2 Natural Language Processing 31

3.3 Lavori correlati 34

3.4 Introduzione Cogito 39

4. Il robot umanoide Nao 41

5. Cogito 47

5.1 La piattaforma 49

5.2 Categorizzazione ed estrazione 50

5.3 GSL ESSEX Export 58

5.4 Il progetto “Indovina Chi?” 59

6. Data Base 63

6.1 SQLite 69

7. Script Python 71

7.1 Turno 1 e algoritmo di matching 72

7.2 Turno 2 75

8. Conclusioni e sviluppi futuri 77

Bibliografia 79

Ringraziamenti 81

Indice delle figure

Figura 1 : Nao 10
Figura 2 : Personaggi Indovina Chi 13
Figura 3: Schema a blocchi 17
Figura 4 : Livelli Speech Recognition 20
Figura 5 : Connessioni monodirezionali e bidirezionali LSTM RNN 22
Figura 6 : Campionamento frame 23
Figura 7 : Esempio API Google Speech Recognition 25
Figura 8 : Analisi 29
Figura 9 : Architettura NLP con database 33
Figura 10: Babelnet 35
Figura 11: Esempio 1 Babelnet 36
Figura 12: Esempio 2 Babelnet 36
Figura 13: Esempio NLTK 37
Figura 14: Albero sintattico 38
Figura 15: Regola NLTK 38
Figura 16: Demo Stanford CoreNLP 39
Figura 17 : Caratteristiche Nao 41
Figura 18 : Box Say e Speech Recognition 43
Figura 19 : Box Swich Case 43
Figura 20 : Creazione Box 45
Figura 21 : Implementazione Box 45
Figura 22 : Cogito 47
Figura 23: Sensigrafo 48
Figura 24: Semantic Rules 49
Figura 25 : Workflow Cogito Studio 49
Figura 26: Layout di Cogito Studio 50
Figura 27: Sintassi categorizzazione 51

Figura 28: Operatore AND 52
Figura 29: Operatore OR 52
Figura 30: Operatore AND NOT 53
Figura 31: Operatore & 54
Figura 32: Categorizzazione 54
Figura 33: Sintassi estrazione 55
Figura 34: Attributo TYPE 55
Figura 35: Attributo PATTERN 56
Figura 36: Operatore + ... 56
Figura 37: Operatore - 57
Figura 38: Operatore ENTRY 57
Figura 39: Estrazione 57
Figura 40: GSL ESSEX Export 58
Figura 41: Esempio Request.xml 61
Figura 42: Esempio Response.xml 62
Figura 43: Modello ER 65
Figura 44: Tabella Personaggi 66
Figura 45: Tabella PersonaggiIndovinati 67
Figura 46: Esempio turno 1 (risposta positiva) 74
Figura 47: Modifica tabella PersonaggiIndovinati 74
Figura 48: Esempio turno 1 (risposta negativa) 74
Figura 49: Modifica campo "Escluso" 75
Figura 50: Robot tenta di indovinare 75
Figura 51: Esempio turno 2 76
Figura 52: Umano tenta di indovinare 76
Figura 53: Esempio tabella "Partite" 76

Parole chiave

Indovina chi?

Robot Nao

Speech recognition

Natural language processing

Artificial intelligence

Introduzione

Il viso è il componente umano più significativo per il riconoscimento di un individuo e generalmente ognuno di noi per individuare una persona la associa all'immagine del suo volto. Basti pensare alla carta di identità come mezzo più comune di identificazione.

Di conseguenza, un'immagine che ritrae il volto è uno dei metodi più semplici ed efficaci per distinguere un individuo, perché piena di caratteristiche diverse. Da essa possiamo ricavare molte informazioni come etnia, sesso, età di un soggetto.

Le varie particolarità fisiche possono essere descritte verbalmente attraverso aggettivi che descrivono una parte del corpo in particolare.

Per ogni parte del corpo ci sono una lista di aggettivi che solitamente vengono associati ad essa. Per esempio per descrivere il colore degli occhi si possono usare aggettivi come "marroni", "azzurri", "verdi", o per la loro forma aggettivi come "a mandorla", "tondi", "grandi".

Quello su cui ci si vuole soffermare è che ogni descrizione contiene una parola chiave che identifica la parte del corpo alla quale si può affiancare una serie di parole che la descrivono.

Come descritto in [1], l'insieme delle parti del corpo con i rispettivi aggettivi può identificare una persona e, in particolare, più ampia è la descrizione ottenuta e più sarà facile riconoscerla in un insieme di individui.

Supponiamo l'esistenza di un database al cui interno sono riportate le caratteristiche tramite le quali una serie di soggetti potrebbero essere identificati. Se attraverso una macchina si volessero elaborare i dati in modo da ricavare una o più persone con determinati requisiti si potrebbero estrarre i dati attraverso una query.

Nella vita quotidiana, normalmente, queste informazioni vengono espresse verbalmente.

Il problema che si vuole introdurre è proprio quello di identificare un individuo da una descrizione verbale in modo automatico.

L’obiettivo della tesi, quindi, è la progettazione di un sistema in grado di capire ed elaborare dei dati comunicati usando un sottoinsieme del linguaggio naturale, estrapolarne le informazioni chiave e ottenere un riscontro con informazioni date in precedenza.

Questi problemi sono stati affrontati e discussi attraverso il robot “NAO” ed un gioco di società chiamato “Indovina chi”.

La figura 1 raffigura il robot Nao su cui è stato studiato il progetto descritto in seguito.

Figura 1 : Nao

Una persona deve interagire con il robot Nao, il quale deve essere in grado di giocare autonomamente ad una partita del gioco in questione.

Il robot è messo in condizione di poter ascoltare la persona, la quale gli fornisce le informazioni e gli fa delle domande riguardo il gioco.

Le parole ascoltate vengono poi trasformate in testo ed elaborate in modo da poter catturare i dati fondamentali.

Nel caso di una domanda da parte del giocatore, Nao deve essere in grado di rispondere in modo adeguato.

Nel caso in cui fosse il turno di Nao di fare una domanda all'avversario, il robot deve essere capace di immagazzinare nel modo corretto le informazioni ricevute.
Attraverso calcoli probabilistici Nao cercherà di vincere il gioco.
Per memorizzare i dati è stato costruito un database con una tabella contenente tutte le caratteristiche dei personaggi e una che successivamente sarà popolata con le informazioni ricavate dal robot, una riga per ogni partita. Infine, è stata creata una terza tabella in cui verranno salvate tutte le partite giocate.
Per l'audio, il quale deve essere trasformato in forma testuale, è stato necessario utilizzare API di terzi, in questo caso Google speech recognizer [15].
Per la comprensione del testo è stato utilizzato un programma fornito dall'azienda Expert System [13] chiamato Cogito Studio. Questo programma dà la possibilità di scrivere delle regole per estrarre i dati necessari a capire la domanda posta dal giocatore umano o le informazioni fornite in risposta alle domande poste dal robot. Una volta stabilita la caratteristica di cui si sta parlando sarà possibile, tramite script sql, popolare la tabella con le nuove informazioni o riuscire a rispondere correttamente facendo un interrogazione al database.
Il linguaggio di programmazione utilizzato è Python [12] e il database è stato creato con l'utilizzo di SQLite [11].
Gli script per la programmazione del robot Nao sono stati realizzati con il linguaggio Python, con l'integrazione di script in linguaggio SQL per l'interazione con il database.
I risultati ottenuti dal nostro progetto sono stati soddisfacenti. Su 20 partite giocate la percentuale di vincita del robot è risultata pari al 50%.
I calcoli probabilistici sono stati implementati in Python, creando una funzione che restituisce il valore di probabilità di vittoria se il robot tentasse di indovinare. Le domande del robot non sono generate in modo totalmente casuale, ma vengono prese in considerazione le domande precedenti, con i relativi risultati, così da non proporre domande inutili all'umano successivamente. Se ad esempio alla domanda "è

femmina?” il robot riceve esito positivo, saranno eliminate tutte le domande relative alle caratteristiche maschili come la barba o i baffi.

Grazie al programma Cogito Studio è stato possibile realizzare il gioco di “Indovina Chi?” verosimile al classico gioco scrivendo le regole di analisi del testo come descritto precedentemente.

La tesi è organizzata come segue.

Nel capitolo 1 si descrive il progetto che si vuole realizzare e le aspettative, quindi è stato introdotto il problema in linea generale spiegando come si intende risolverlo.

Nel capitolo 2 si studia la parte del riconoscimento vocale, utile per capire il funzionamento di Google Speech Recognition utilizzato nel nostro progetto.

Nel capitolo 3 si affronta l’elaborazione del linguaggio naturale e il tema del linguaggio naturale e della comunicazione attraverso esso.

Nel capitolo 4 si vede una descrizione del robot Nao e una introduzione di come funziona e come può essere programmato attraverso il programma Choreographe.

Nel capitolo 5 si parla della piattaforma Cogito Studio, delle sue funzionalità e di come è stata utilizzata per il nostro progetto.

Nel capitolo 6 si vede come è stato realizzato il database, utile per contenere tutte le informazioni del gioco.

Infine nel capitolo 7 si studia come è stato implementato l’algoritmo di matching in modo che il robot possa giocare correttamente ad “Indovina chi?”.

1. Il progetto “Indovina chi?”

Il gioco “Indovina chi?” è un classico gioco per bambini.

Consiste in due giocatori i quali hanno un tabellone uguale con 24 figure di personaggi che raffigurano il viso di ognuno.

Nella figura 2 sono raffigurati i personaggi utilizzati nel progetto.

Figura 2 : Personaggi Indovina Chi

Ogni partecipante sceglie un personaggio che deve essere indovinato dall’avversario.

Vince chi per primo riesce ad indovinare il personaggio scelto dall’avversario.

A turno i giocatori devono porre una domanda riguardo un aspetto fisico e l’avversario deve rispondere in modo veritiero.

Nel gioco originale le domande sono a risposta affermativa o negativa ma, nel caso che si vuole studiare, sono possibili anche domande a risposta aperta.

Il gioco quindi consiste nell’abilità di proporre le giuste domande al fine di arrivare per primi alla soluzione tramite esclusione.

Anche se non si è certi, si può provare ad indovinare il personaggio se si è convinti di avere una buona probabilità di vittoria.

1.1 Il test di Turing

Alan Turing, come scrisse nell'articolo [2], si chiese "Le macchine possono pensare?" .

Il test di Turing si basa proprio su questa domanda, come possiamo sapere se una macchina è in grado di pensare?

Il problema da lui descritto è basato su un gioco chiamato "Il gioco dell'imitazione".

Il gioco ha tre partecipanti: un uomo, una donna ed un interrogatore. L'interrogatore viene separato in un'altra stanza e pone delle domande ai due soggetti che chiameremo A e B.

Alla fine del gioco l'interrogatore deve affermare chi è l'uomo e chi è la donna.

Però il giocatore A può trarre in inganno l'interrogatore, mentre B ha il compito di aiutarlo a non sbagliare. Ma l'interrogatore non sa chi dei due ha i rispettivi compiti e quindi viene messo in difficoltà.

Turing si chiese: cosa accade se una macchina prende il posto di A?

L'interrogatore sarà portato a sbagliare tante volte quanto prima?

Questa domanda riporta alla domanda principale: le macchine possono pensare?

La comunicazione può avvenire solo in forma scritta, o meglio ancora dattiloscritta. I partecipanti a seconda delle domande possono vantarsi delle loro capacità, per esempio di aver vinto una gara di corsa, ma l'interrogatore non può chiedere una prova pratica.

Se l'umano volesse imitare la macchina, sarebbe subito messo in difficoltà da una domanda di aritmetica. La macchina risulterebbe sempre più veloce.

La migliore strategia che può usare la macchina è invece quella di fornire risposte che sarebbero normalmente fornite da un uomo.

Turing con la parola "macchina", che dovrebbe risultare "pensante", nel gioco intende solo computer digitali e quindi sono escluse tutti gli altri tipi di macchine. L'idea di base è che le macchine digitali siano in grado di effettuare tutte le operazioni che potrebbe fare un calcolatore umano.

Se l'interrogatore ha una percentuale di errore circa uguale a quella prima che si sostituisse A, secondo Turing si può dire che la macchina è pensante.
Il test di Turing è stato introdotto per comprendere meglio lo scopo del nostro progetto, infatti se la percentuale di vincite tra umano e robot è simile a quella che si otterrebbe tra due umani si può dire che il robot può sostituire un umano.
Per rendere il robot simile ad un umano si è fatto in modo che esso comprenda il più possibile il linguaggio naturale. L'umano ha la possibilità di esprimersi con parole proprie per quanto riguarda le caratteristiche del personaggio e il robot è in grado di comprendere e gestire le informazioni ricevute.
In modo simile al test di Turing l'umano può trarre in inganno l'interrogatore che in questo caso è il robot.
Nel nostro progetto per inganno si intende quando l'umano per descrivere la caratteristica di un personaggio inserisce informazioni non inerenti ad essa. Un esempio di inganno alla domanda "Di che colore ha i capelli?" è la risposta "Il personaggio ha i capelli biondi ma io li ho rossi". Il robot deve saper riconoscere le informazioni da considerare e quelle da scartare.
Nel nostro progetto l'analisi e la comprensione del linguaggio naturale sono state sviluppate con la tecnologia di Cogito e quindi sono state realizzate condizioni affinché il robot comprenda al meglio ciò che vuole comunicare l'umano.
Data la possibilità di ingannare il robot è stato necessario introdurre un vincolo nell'analisi delle risposte ottenute dall'umano, infatti il robot prende in considerazione solo ciò che è legato alla parola "personaggio". In questo modo è in grado di riconoscere le giuste informazioni anche se viene imposto il vincolo di utilizzare sempre il soggetto "personaggio" quando ci si riferisce ad esso.

1.2 Indovina chi? Il problema

Il gioco "Indovina chi?" apparentemente potrebbe sembrare banale, ma nel contesto in cui si vuole sviluppare si evidenziano molti aspetti importanti su cui soffermarsi.

Il problema principale è fare in modo che il robot Nao si sostituisca ad uno dei due giocatori e riesca ad interagire con l'avversario per giocare e cercare di vincere la partita.

L'ostacolo più importante e complesso è quello della comprensione da parte del robot del linguaggio usato dall'interlocutore.

Il primo passo consiste nel riuscire a catturare tutte le parole pronunciate dalla persona per poi, tramite elaborazione, trascriverle in forma testuale. In questo caso si utilizza l'API di Google per il riconoscimento vocale, la quale è in grado di svolgere in modo preciso questo compito.

Nonostante la trasformazione del segnale vocale in testo, il robot non ha ancora dato un senso logico a ciò che la persona sta tentando di comunicargli. Con il primo passaggio Nao ottiene soltanto una sequenza di parole, le quali non hanno ancora assunto un significato.

La semantica è la complicazione più significativa di questo problema.

Nel caso particolare di questo gioco, il contesto delle frasi pronunciate dall'umano è sempre la descrizione di un personaggio o la risposta a una specifica domanda. È quindi necessario riuscire ad interpretare le frasi nel modo corretto e raccogliere le informazioni per poi memorizzarle ed elaborare i dati ottenuti.

Questo importantissimo compito è stato svolto con l'ausilio della tecnologia di "Cogito", sviluppata dall'azienda Expert System.

Per ottenere una buona probabilità di vincita si è programmato il robot implementando algoritmi di strategia.

È stato necessario un database per la raccolta di tutti i personaggi del gioco con le rispettive caratteristiche. Il robot deve sempre interfacciarsi con esso per poter rispondere alle domande dell'avversario o per fare interrogazioni in base ai dati ricevuti. Esiste inoltre una tabella dove il robot inserirà tutte le informazioni ricevute per indovinare il personaggio.

Nella figura 3 uno schema a blocchi che descrive come è strutturata la tesi.

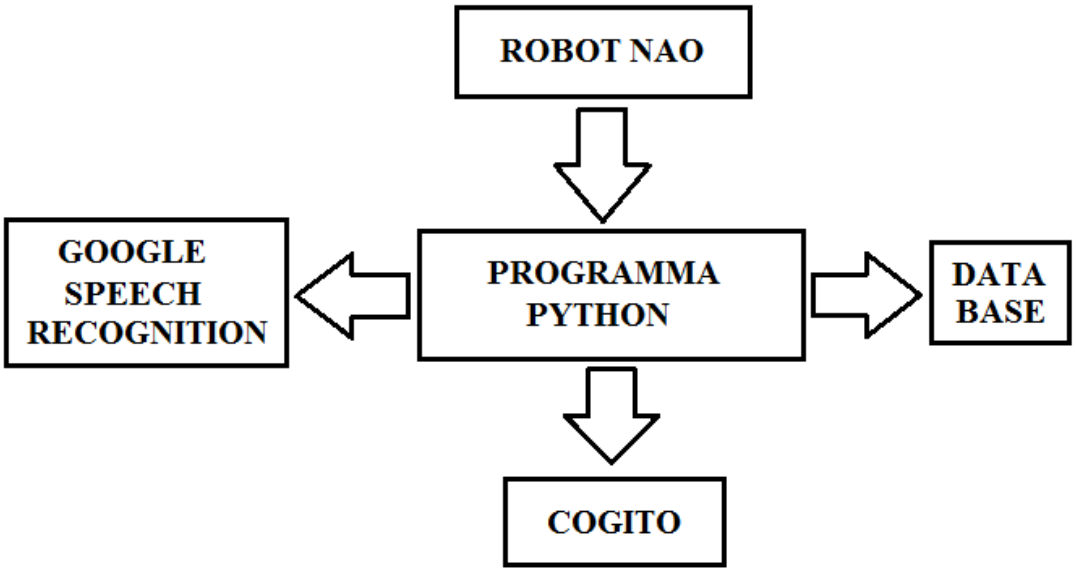

Figura 3: Schema a blocchi

2. Riconoscimento vocale

Un sistema di “speech recognition”, ovvero di riconoscimento vocale, è un sistema il cui funzionamento consiste nel riuscire a capire le parole ricevute attraverso un linguaggio orale e successivamente elaborarle a seconda delle esigenze.

Questi sistemi sono stati sviluppati per una grande varietà di applicazioni e perché siano validi devono essere facili da usare, naturali e devono avere un’interfaccia vocale flessibile tra la persona e la macchina.

Il segnale vocale è uno dei più complicati su cui dover lavorare, infatti si può osservare che il segnale risulta sempre diverso anche se vengono pronunciate le stesse parole dalla medesima persona.

Nel nostro progetto è stato necessario utilizzare il riconoscimento vocale in modo tale da rendere il gioco il più possibile verosimile rispetto a quello tra due persone. Per questo è stata utilizzata l’API di Google Speech Recognition.

In questo capitolo si introduce il funzionamento di un sistema di riconoscimento vocale e dei metodi che possono essere utilizzati per realizzare questa funzione.

Il segnale vocale, oltre a essere diverso da un interlocutore ad un altro, è influenzato dal trasduttore usato per catturarlo, dal canale usato per trasmettere il segnale e dal rumore che può essere generato dall’ambiente.

Uno dei metodi, come studiato nel libro [3], per il riconoscimento vocale può essere eseguito identificando i suoni corrispondenti, quindi tramite l’ausilio di un dizionario viene decodificata la lista di parole.

Questo metodo solitamente non è usato per la difficoltà nel distinguere i suoni in un discorso, quindi il riconoscimento risulta inaffidabile.

Un approccio di maggior successo, descritto in [3], è quello di trattare il segnale vocale come un modello stocastico: utilizzando un modello di riconoscimento statistico. Attraverso calcoli di probabilità potrà essere deciso se la sequenza di parole ascoltate è corretta. Sarà valutata la possibilità che la sequenza acustica dia una particolare sequenza di parole e che possa essere stata generata nella lingua stabilita.

Il riconoscimento vocale nel libro [3], da cui è tratta la figura 4, è suddiviso in tre livelli.

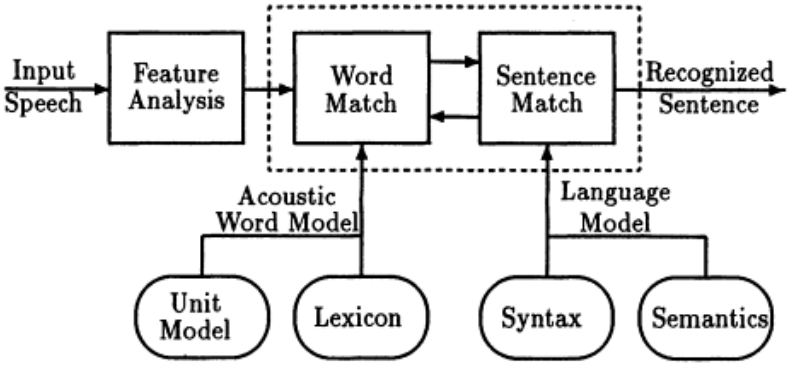

Figura 4 : Livelli Speech Recognition

Il primo riceve in input il segnale e genera il vettore con le caratteristiche acustiche usato per determinare le proprietà spettrali, le quali vengono generalmente estratte usando l'analisi di Fourier.

Il secondo valuta la somiglianza tra la sequenza in input e le parole nel vocabolario utilizzando un modello acustico per stabilire quali di esse sono state più probabilmente pronunciate. Esso si basa su un insieme di modelli di sottoparole e un lessico.

I modelli di sottoparole sono composizioni di blocchi per formare parole e frasi e corrispondono alle classi fonetiche più usate nella maggior parte dei sistemi di riconoscimento vocale. Nel caso più semplice le unità di sottoparole sono indipendenti dal contesto.

Il lessico fornisce una descrizione delle parole in termini di insieme di unità di sottoparole. Nella maggior parte dei sistemi di riconoscimento vocale il lessico è estratto da un dizionario standard e ogni parola è rappresentata da una singola voce del vocabolario nella forma base. La variabilità lessicale è caratterizzata solo indirettamente attraverso i modelli di sottoparole.

La forma base identifica una guida di come dovrebbe essere pronunciata una parola, il numero delle pronunce alternative indica la misura di variabilità di quella parola.

L’ultimo livello utilizza un modello di linguaggio (sintattico e semantico) per determinare quale frase è stata espressa con maggior probabilità.
La grammatica imposta, un insieme di regole sintattiche e semantiche, fa in modo che la frase sia quella ottimale.

2.1 Deep learning

Il “Deep learning”, come spiegato in [4], è un ramo dell’apprendimento automatico basato su un insieme di algoritmi che tentano di modellare, ad un alto livello di astrazione e usando multipli livelli di elaborazione, strutture complesse o diverse trasformazioni non lineari.
Una delle architetture del deep learning è la “Deep Neural Network” (DNN) e uno dei campi in cui può essere applicata è il riconoscimento vocale e l’elaborazione del linguaggio naturale.
Per “Deep Neural Network” si intende una rete neurale artificiale creata da più livelli di unità nascosti tra input e output. Esso, può modellare complesse relazioni non lineari.
Le DNN sono tipicamente designate come reti feed-forward .
Un’altra architettura è il “Recurrent Neural Network” (RNN), in essa le connessioni tra le unità formano un ciclo diretto. Questo crea uno stato interno della rete che consente un comportamento dinamico temporale.
Le RNN, a differenza delle reti feed-forward, possono usare la memoria interna per processare arbitrariamente sequenze in input. Questo le rende idonee per il riconoscimento vocale.

2.1.1 RNN per il riconoscimento vocale

Recentemente, secondo il testo [5], è stato dimostrato che le RNN possono superare le reti di tipo feed-forward per il riconoscimento vocale, in particolare le Long Short-Term Memory (LSTM RNN).

LSTM RNN è un'architettura RNN adatta a classificare, processare e predire serie temporali.

Le reti RNN modellano le sequenze in ingresso in modo unidirezionale e bidirezionale.

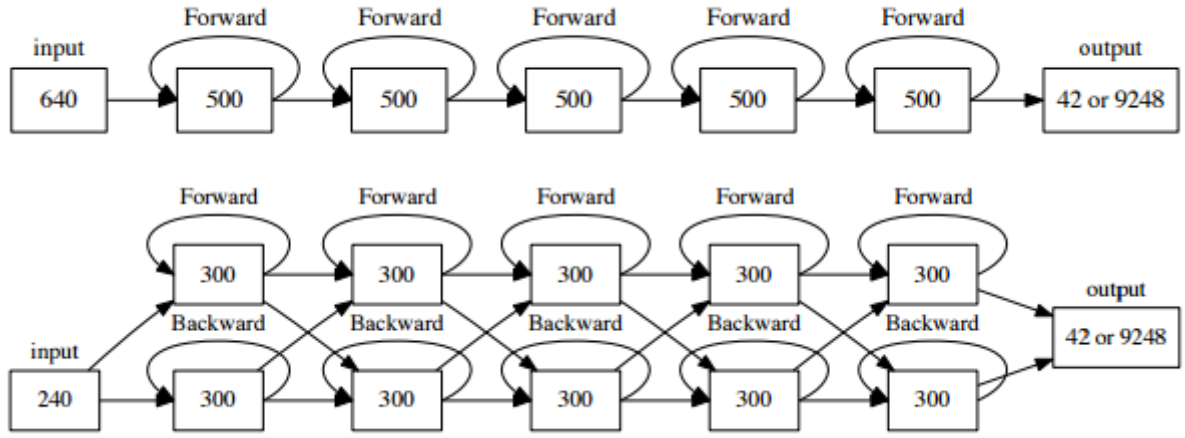

Figura 5 : Connessioni monodirezionali e bidirezionali LSTM RNN

Nel primo caso rappresentato dalla figura 5 viene utilizzato solo il contesto a sinistra rispetto all'input corrente. Viene quindi processato l'input da sinistra verso destra, con uno stato nascosto in avanti.

Questo, come descritto in [5], è consigliabile per applicazioni che richiedono bassa latenza tra ingresso e uscita.

Se invece un applicazione può permettersi la latenza può usare le RNN bidirezionali (secondo caso della figura 5), le quali hanno livelli separati per processare l'input in entrambe le direzioni.

Lo studio di [5] dice che usando l'architettura LSTM RNN, costruita con la sovrapposizione di più livelli LSTM, si dimostra che il riconoscimento vocale viene svolto in modo migliore rispetto ad altri modelli. In particolare i modelli bidirezionali utilizzano due livelli LSTM collegati tra loro, il livello di output è collegato alla parte finale di entrambi.

L’approccio CTC (Connectionist-Temporal-Classification), descritto nel testo [5], è una tecnica per l’etichettatura delle sequenze che utilizza le RNN quando la corrispondenza tra input ed etichetta di destinazione è sconosciuta.

CTC può essere implementato con un livello di output che utilizza un’unità addizionale per l’etichetta “blank”, la quale viene usata per stimare la probabilità che entro un certo tempo non sia stata emessa nessuna etichetta.

Le probabilità di etichetta dell’output definiscono una distribuzione di probabilità su tutte le possibili etichettature delle sequenze in input incluse quelle di tipo “blank”.

La rete può essere realizzata in modo da ottimizzare la probabilità totale di corretta etichettatura usando gli output e algoritmi di tipo forward-backward.

La corretta etichettatura per una sequenza in input è definita come l’insieme delle possibili etichette nella giusta sequenza, eventualmente con ripetizioni ed etichette “blank” tra due separate.

Il criterio CTC è valido per il WER (Word Error Rate).

Per migliorare la precisione dei modelli acustici RNN inizializzati con il criterio CTC si utilizza il criterio di “Sequence Discriminative training”.

Un numero di livelli di “Sequence Discriminative training” incorporano il lessico e i vincoli del modello di linguaggio usati nel discorso decodificato.

Dopo il criterio di “Sequence Discriminative training” la differenza tra CTC e un modello convenzionale è quello dell’uso del simbolo “blank”.

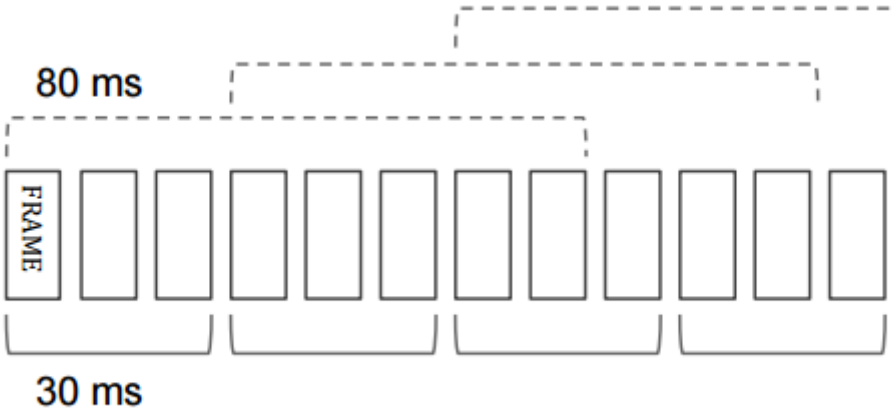

Figura 6 : Campionamento frame

Nella figura 6 è mostrato un esempio di impilamento e sottocampionamento di frame. Le caratteristiche acustiche sono generate ogni 10ms ma sono concatenate e ricampionate per l'ingresso alla rete, otto frame sono impilati per il modello unidirezionale e tre per quello bidirezionale.

CTC utilizza modelli indipendenti dal contesto, ma è bene sapere che gli stati dipendenti dal contesto li rendono più performanti nel caso di 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 e quindi per i modelli CTC.

La combinazione della memoria del 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".

2.2 Google Speech Recognition

Attualmente Google dice che la sua tecnologia per il riconoscimento vocale ha un tasso di errore di solo 8% (WER) mentre nel 2013 era del 23%. Infatti Google ha acquistato diverse società di deep learning nel corso degli anni.

Il deep learning, come già introdotto, comporta un'ingestione di grandi quantità di dati per la formazione delle reti neurali.

Sundar Pichai, il vice presidente senior di Android, Chrome e Apps di Google ha annunciato che la loro rete neurale ha 30 livelli.

Nel nostro progetto è stata utilizzata l'API di Google Speech Recognition per la trasformazione delle frasi pronunciate vocalmente dal giocatore umano in forma testuale.

In particolare questa API è stata integrata nello script in Python, come mostrato in [16], utilizzando la funzione : recognize_google(audio,language="it-IT").

Testando più volte la funzionalità di Google Speech Recognition (in Italiano) si è vista la precisione con cui vengono catturate le parole espresse dalla persona. L'unico

punto su cui ragionare è che non è rilevata la punteggiatura. Nel caso specifico del progetto è importante identificare una domanda, ma l'API non identifica la presenza del punto interrogativo alla fine della frase. Questo problema, nel nostro caso, può essere ovviato dalla conoscenza dei turni e quindi non sarà necessario stabilire se è stata posta una domanda attraverso il riconoscimento del punto interrogativo perché avendo turni prestabiliti diventa implicito.

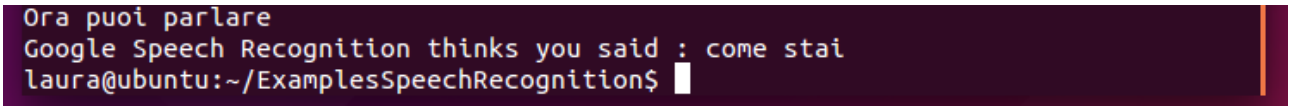

Figura 7 : Esempio API Google Speech Recognition

In figura 7 un piccolo esempio di come funziona l'API e della mancata rilevazione del punto interrogativo.

3. Il linguaggio naturale

Il linguaggio naturale è una delle parti più complesse da affrontare in questo contesto. Il focus del progetto consiste nel giocare ad "Indovina Chi?" utilizzando il riconoscimento vocale e successivamente, una volta tradotta in forma testuale la frase pronunciata, riuscire a comprendere ciò che è stato comunicato.

In questo capitolo si affronterà il tema del linguaggio naturale e della comunicazione attraverso esso.

3.1 La comunicazione

Il libro [6] spiega che qualsiasi modalità di comunicazione può essere divisa in tre parti: un parlante, un ascoltatore e un enunciato. Un agente è qualcosa che agisce, che fa qualcosa. Il parlante e l'ascoltatore sono agenti che compiono un atto linguistico. Gli agenti possono interrogare, informare (anche attraverso risposte a delle domande), richiedere di eseguire azioni da altri agenti, acconsentire e promettere.

Nel caso in esame gli agenti sono il robot e il giocatore umano. Le azioni che devono compiere i due agenti sono interrogare ed informare. L'azione di interrogare viene effettuata quando il giocatore chiede all'avversario una caratteristica del personaggio da indovinare mentre l'azione di informare è effettuata dal giocatore in risposta all'interrogazione.

Il problema della comunicazione, come descritto in [6], è la comprensione degli atti linguistici e l'agente deve procedere all'indietro per decifrarli. La comunicazione è un atto pianificato e la sua comprensione quindi prevede il riconoscimento di piani.

Un linguaggio formale è un linguaggio composto da un insieme di stringhe che possono essere simboli o parole ed è definito da regole matematiche precise. Un esempio di linguaggio formale è il linguaggio di programmazione Java. Un linguaggio naturale, come l'italiano o l'inglese, differiscono dal linguaggio formale perché non hanno regole precise e sono spesso ambigui.

Per riuscire a comprendere un testo in linguaggio naturale bisogna trattarlo come un linguaggio formale anche se il risultato finale difficilmente sarà perfetto. Questo perché per la maggior parte delle frasi esistono 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 attribuita una semantica ovvero un significato. È molto importante anche la pragmatica di una stringa cioè il suo significato effettivo nel momento in cui viene pronunciata.
Per descrivere la grammatica il libro [6] dice che nella maggior parte dei casi si utilizza una struttura sintagmatica, le stringhe sono formate da sottostringhe (sintagmi) e suddivise in categorie.
Un sintagma può essere di tipo nominale (NP) o verbale (VP). Un sintagma nominale seguito da uno verbale genera una frase (S).
La comunicazione, nel libro [6], viene divisa in sette step: intenzione, generazione, sintesi, percezione, analisi, disambiguazione e assimilazione:

- L'intenzione si verifica quando il parlante decide di comunicare una certa proposizione all'ascoltatore.
- Per generazione si intende la pianificazione da parte del parlante per trasformare la proposizione in un enunciato.
- La sintesi è la realizzazione fisica delle parole dell'enunciato che il parlante vuole comunicare.
- La percezione avviene da parte dell'ascoltatore quando riceve le parole dal parlante.
- L'analisi è la procedura con cui l'ascoltatore decodifica i possibili significati dell'enunciato. Essa si suddivide in tre parti: parsing, interpretazione semantica e interpretazione pragmatica. Il parsing consiste nella costruzione di un albero in cui i nodi sono sintagmi, le foglie sono le parole e la radice è la frase S.
    - L'interpretazione semantica ricerca il significato dell'enunciato con diverse interpretazioni.

- L'interpretazione pragmatica considera i diversi significati che possono assumere le parole in contesti differenti.

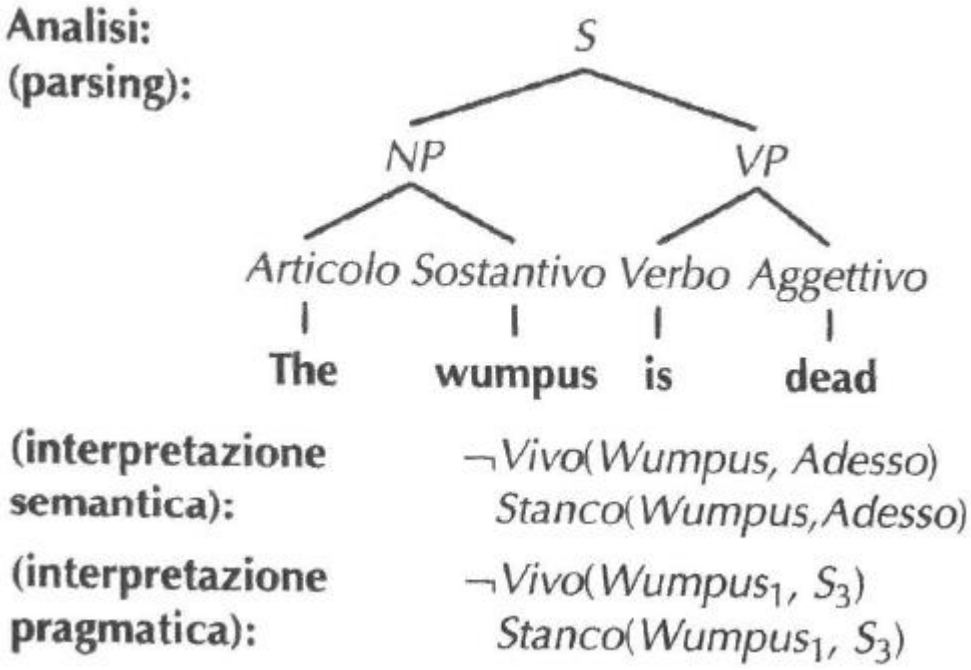

Figura 8 : Analisi

La figura 8 mostra un esempio di analisi di una frase.

- La disambiguazione ha il compito di scegliere la migliore interpretazione tra quelle proposte dall'analisi.
- L'assimilazione è l'ultimo passo della comunicazione e consiste nella decisione dell'ascoltatore di credere o meno al parlante.

Nel contesto del gioco, bisogna definire la grammatica formale per esprimere domande o risposte che possono essere generate dai due giocatori. Quindi non è indispensabile definire regole per tutta la grammatica del linguaggio italiano, che risulterebbe estremamente complicato, ma solamente quelle necessarie per poter giocare.

Come descritto in [6] il lessico è l'insieme delle parole accettabili per la nostra grammatica. Le parole sono suddivise in categorie come i nomi propri, verbi o articoli. Le categorie possono appartenere ad una classe di tipo aperto o chiuso. Le

classi aperte sono quelle che contengono un numero indeterminato di parole e che possono essere aggiornate con nuovi elementi. Le classi chiuse sono invece degli insiemi finiti di parole come le preposizioni o gli articoli.

Per descrivere le regole della grammatica sono definiti cinque simboli non terminali: frase (S), sintagma nominale (NP), sintagma verbale (VP), sintagma preposizionale (PP) e proposizione relativa (RelClause).

Usando questi simboli è possibile determinare le regole con cui essi possono combinarsi per generare frasi diverse.

Il problema affrontato nel libro [6] è che nonostante le regole imposte, la grammatica può sia sovragenerare, dando la possibilità di generare frasi non corrette, sia sottogenerare, rifiutando frasi corrette nonostante non rispettino le regole stabilite.

Nel progetto il robot ha un insieme di domande prestabilite, le quali possono essere poste in ordine casuale, e deve generare delle risposte utilizzando il database che ha a disposizione. Quindi nel caso specifico il problema si pone solamente nel riconoscimento delle domande o delle affermazioni poste dalla persona.

Nell'analisi sintattica, descritta in [6], il parsing può essere eseguito in due modi. Il primo metodo è quello top-down, partendo dalla radice si cerca un albero che corrisponda alle parole dell'enunciato finale. Il secondo metodo è quello bottom-up, partendo dalle parole si cerca di ricostruire l'albero iniziando dalla parte sinistra dell'enunciato e ricavando le regole che riconducono alla radice S.

Entrambi i metodi di parsing non si possono considerare efficaci perché l'algoritmo non può determinare il giusto risultato fino alla fine della frase. Ad esempio si possono confondere domande con affermazioni distinte solo dalla punteggiatura alla fine dell'enunciato. In caso di errore si dovrà ripetere la procedura rendendo l'operazione molto lunga.

Per eliminare questo problema ed avere un parsing efficiente, i parser devono memorizzare i risultati ottenuti in modo da non doverli analizzare nuovamente.

Le regole della grammatica che abbiamo creato per le nostre esigenze consentono comunque di generare frasi scorrette. Questo può accadere perché alcuni sintagmi, nonostante siano della stessa categoria, non sono uguali ad altri.

Un esempio tratto dal nostro progetto può essere la rivelazione del colore dei capelli del personaggio. La regola riportata sotto impone che la frase abbia una certa sequenza, ma come si nota il verbo "essere" può essere coniugato anche in modo errato. Per esempio la frase "I capelli del personaggio è biondi" nonostante sia grammaticalmente scorretta il sistema consente di generarla. In questo caso si è scelto di avere un maggiore grado di libertà rispetto ad una regola più precisa che però ha lo svantaggio di essere più restrittiva.

```
LEMMA("capelli") <> LEMMA("personaggio") > LEMMA("essere") >
@Colore[ANCESTOR(142282,224070,132153,294195,131371,131380)]
```

Con questo esempio si vuole far notare la possibilità di scelta in base alle priorità del progetto a cui si sta lavorando.

Il libro [6] propone due possibilità per risolvere questo problema. La prima possibilità è aggiungere nuove categorie, ma così molte regole saranno duplicate, la seconda è quella di aumentare la nostra grammatica, ovvero parametrizzare le categorie. In questo modo rimarranno le regole di base e si aggiungeranno nuove regole a seconda del parametro che indica il caso.

La comunicazione del gioco è caratterizzata da turni, quindi si ha il vantaggio di conoscere in anticipo se ci si aspetta di ricevere una domanda o una risposta. Con il supporto di Cogito Studio sono state formulate le regole con cui estrarre le informazioni necessarie per rispondere o immagazzinare informazioni. Se in una frase sono riportate ulteriori informazioni non rilevanti il sistema dovrà essere in grado di non considerarle.

3.2 Natural Language Processing

Per Natural Language Processing (NLP) s'intende un sistema che analizza, cerca di comprendere o produrre una o più lingue, come l'inglese o l'italiano, in base al compito che deve eseguire.

È difficile definire il modo in cui un sistema dovrebbe comprendere un linguaggio, ma possiamo provare tramite dei test quanto sembra in grado di capirlo.
Il classico modello è il test di Turing, già descritto precedentemente. Se durante il test il sistema è indistinguibile da un umano allora si può dire di aver ottenuto con successo lo scopo desiderato.
Mentre il test di Turing non prevede una valutazione intermedia del lavoro compiuto dalla macchina, per quanto riguarda il NLP in [7] è sostenuto che si stanno sviluppando modelli più sensibili per valutarne le prestazioni.
Per produrre i test di valutazione solitamente vengono verificate le specifiche funzionalità all'interno di un dominio limitato.
La difficoltà principale dell' elaborazione del linguaggio naturale è l'ambiguità che si può trovare in tutti i livelli del problema.
Un elenco di tipi di ambiguità che possono verificarsi, come descritto in [7], sono:

- ambiguità lessicale (un nome può essere anche un verbo)
- ambiguità sintattica (interpretazione scorretta della frase in base al contesto)
- ambiguità semantica (una parola può avere significati diversi)
- ambiguità pragmatica (il significato della parola in un determinato contesto)
- ambiguità di riferimento (un riferimento incerto).

Inoltre, ogni forma di ambiguità può interagire con le altre, rendendo il processo di interpretazione estremamente complicato. La presenza di ambiguità è proprio ciò che contraddistingue un linguaggio naturale da uno artificiale e rende la maggior parte delle tecniche utilizzate per l'analisi dei linguaggi di programmazione inefficaci.
I sistemi di NLP di maggior successo utilizzano database. Questi sistemi possono comprendere solo domande isolate.
Un tipico esempio di architettura per il database di query è raffigurato nella figura 9.

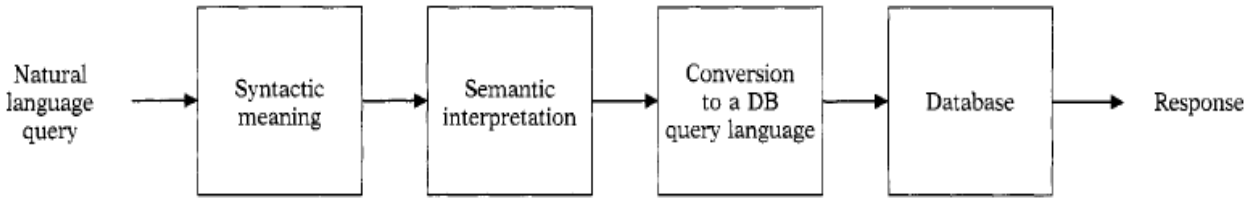

Figura 9 : Architettura NLP con database

La figura 9 rispecchia in modo particolare anche lo schema del progetto in studio. Infatti una domanda posta al robot deve essere trasformata in una query che utilizzi il linguaggio SQL. I primi due blocchi devono essere realizzati con il sistema di Cogito per poi riuscire a collegare la domanda posta in linguaggio naturale ad un dato da fornire come risposta, ottenuto da un database.

Un'altra importante evoluzione nel settore della grammatica e parsing per il linguaggio naturale è lo sviluppo di definite clausole grammaticali basate in Prolog [14]. Prolog offre un efficiente meccanismo per l'analisi grammaticale context-free scrivendo ogni regola come una clausola.

Come detto in [7] il componente per la semantica non è altro che un programma di traduzione dall'output del parser al linguaggio per interrogare il database. Nella maggior parte dei sistemi di ricerca la fase di interpretazione semantica produce una mappatura dall'output ad una rappresentazione conosciuta che supporta le fasi successive di interpretazione pragmatica.

Gli interpreti semantici in [7] sono divisi in due classi principali: compositivi e non compositivi.

Gli interpreti non compositivi permettono trasformazioni arbitrarie dall'output del parser alla forma finale. Quelli compositivi richiedono invece che le regole siano applicate in base alla struttura dell'output del parser, è prevista una singola regola di interpretazione semantica per ogni regola sintattica e possono supportare simultaneamente elaborazioni sintattiche e semantiche durante il parsing.

Inizialmente gli schemi non compositivi erano più comuni a causa della loro maggiore flessibilità, ma ora l'approccio compositivo è usato più frequentemente perché è più modulabile ed estendibile.

Il nostro progetto sotto questo aspetto è abbastanza semplice perché il contesto non cambia e deve soltanto riconoscere le parole chiave da inserire nel database. Quindi non si necessita di una interpretazione dato che il gioco si basa su semplici frasi riferite sempre ad un unico soggetto. Nel caso in cui la persona formulasse frasi inaspettate e non inerenti al gioco il robot si limita a comunicare che non ha compreso e chiede di ripetere.

I sistemi che si occupano della comprensione di un testo esteso come un giornale, un libro o degli articoli necessitano di conoscenze. In particolare, c'è una forte componente pragmatica. È necessaria una significativa conoscenza delle parole anche per singole frasi. Un esempio, tratto da [7], può essere la frase "Jack non può guidare la macchina perché ha perso le chiavi", il sistema deve essere a conoscenza che la macchina per funzionare ha bisogno delle chiavi.

Nei sistemi che si impegnano in un dialogo in linguaggio naturale si presentano molte difficoltà in aggiunta a quelle già descritte. Essi devono riconoscere le intenzioni dei parlanti e produrre risposte ragionevoli. Secondo il lavoro [7] il riconoscimento di piani è un'importante tecnica per la comprensione. Questo modello può essere utilizzato per generalizzare sistemi di domanda-risposta, tuttavia non è ancora in grado di spiegare dialoghi più lunghi di qualche frase. In aggiunta sono stati sviluppati modelli strutturali che appaiono promettenti.

Il sistema che è stato realizzato per il gioco "Indovina Chi?" è di tipo domanda-risposta. Ogni domanda o risposta è riferita al personaggio di cui si sta cercando di individuare l'identità, quindi per quanto riguarda la pragmatica si hanno tutte le informazioni perché il sistema comprenda il significato delle parole nel contesto specifico.

3.3 Lavori correlati

Un esempio di rete semantica è la piattaforma Babelnet [17].

BabelNet è un dizionario enciclopedico multilingua.

Ha una rete semantica molto grande di relazioni che connette concetti ed entità. È composta da circa 14 milioni di voci chiamate Babel synsets, ognuna di esse rappresenta un significato e contiene tutti i sinonimi che possono esprimerlo in diverse lingue.

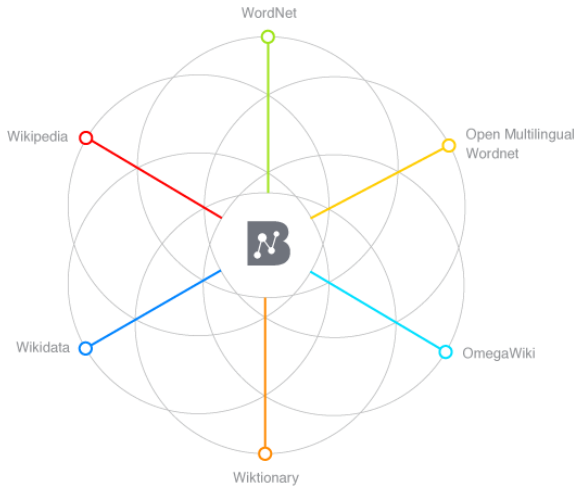

Figura 10: Babelnet

Come mostrato nella figura 10, tratta da [17], il sistema BabelNet utilizza altri sistemi creando una rete che li collega tra loro.

Questa rete semantica analizza la frase scritta e restituisce l'analisi con il significato attribuito alle parole in quel determinato contesto.

Quindi tenta di attuare il compito di disambiguazione. Anche se, mettendolo alla prova con frasi fortemente ambigue, può essere tratto in inganno e sbagliare l'analisi semantica.

Un esempio è l'utilizzo della parola "espresso", può significare a seconda del contesto il caffè, il giornale o il treno.

Se proviamo ha inserire la frase "oggi ho preso un espresso al bar" lo riconoscerà come la bevanda.

Figura 11: Esempio 1 Babelnet

Ma se scriviamo “oggi ho preso l’espresso al bar” lo riconoscerà come giornale.

Figura 12: Esempio 2 Babelnet

Queste frasi potrebbero essere ambigue anche per una persona.

Quindi con questo esempio si vuole far notare quanto è complicato realizzare un sistema di analisi semantica che funzioni correttamente, soprattutto quando vengono inserite frasi ambigue anche per un umano.

Un’altra frase scritta per essere ambigua è “l’espresso l’ho preso in stazione”, il sistema considera l’espresso il giornale e non il treno. Quest’ultima frase è ambigua anche per una persona che potrebbe sbagliare interpretazione tanto quanto il sistema.

Con queste dimostrazioni non si vuole dire che questo sistema (o altri) non siano funzionanti o che lavorino male, ma per quanto siano potenti, quando il livello di ambiguità è troppo alto c’è un forte rischio che vengano prodotti risultati scorretti.

Un altro software per lavorare sul linguaggio naturale è NLTK [18], ovvero "Natural Language Toolkit". Utilizza il linguaggio Python e fornisce più di 50 corpora e risorse lessicali come WordNet [20].

NLTK di default lavora con la lingua inglese ma ha il vantaggio di essere open source. Non effettua l'analisi semantica ma solamente tassonomica ed è in grado di rappresentare una frase con un albero che la descrive.

Un esempio, tratto da [18], di alcune funzioni della libreria "nltk" nella figura 13 . La prima funzione prende in input una frase e la suddivide in token, la seconda funzione assegna ad ogni token un tag e la terza funzione realizza un albero della frase.

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
           ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
      Tree('PERSON', [('Arthur', 'NNP')]),
           ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
           ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

Figura 13: Esempio NLTK

Infine la figura 14, tratta da [18], evidenzia la possibilità di rappresentare in forma grafica l'albero sintattico di una frase.

```
>>> from nltk.corpus import treebank
>>> t = treebank.parsed_sents('wsj_0001.mrg')[0]
>>> t.draw()
```

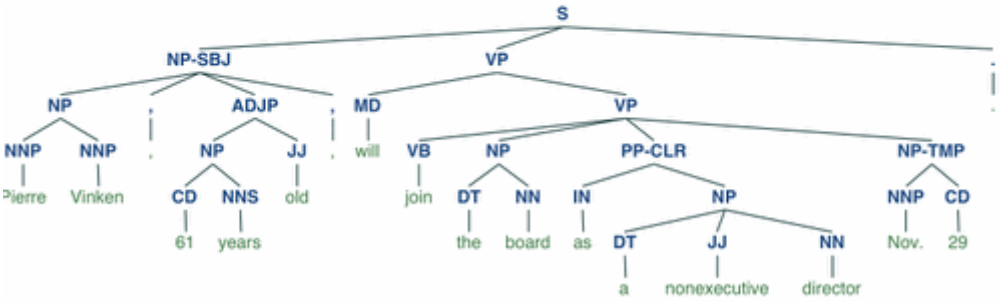

Figura 14: Albero sintattico

Al contrario Babelnet, come già anticipato, è una piattaforma multilingua che è in grado di realizzare un analisi semantica ma non è open source.

NLKT permette di realizzare regole come nella figura 15 tratta da [18].

```
>>> nouns = [("money", "NN"), ("market", "NN"), ("fund", "NN")]
>>> grammar = "NP: {<NN><NN>}  # Chunk two consecutive nouns"
>>> cp = nltk.RegexpParser(grammar)
>>> print(cp.parse(nouns))
(S (NP money/NN market/NN) fund/NN)
```

Figura 15: Regola NLTK

Babelnet invece non da la possibilità di realizzare regole personalizzate. Inoltre NLTK consente di classificare un testo.

Come si può notare Babelnet ha un'interfaccia più semplice e più facile da usare rispetto a NLKT, infatti Babelnet non necessita di installazione e qualsiasi persona sarebbe in grado di utilizzarlo agevolmente. NLKT invece è più flessibile ma più difficile da utilizzare se non si hanno conoscenze pregresse in ambito di programmazione.

Un ultimo esempio di software sono quelli della Stanford NLP Group [19] che fornisce tool statistici, di deep-learning e regole per il natural language processing. Il linguaggio di programmazione utilizzato è Java. Anche il loro software, come NLTK, è open source ma non supporta la lingua italiana.

Stanford NLP Group mette a disposizione vari software tra cui Stanford CoreNLP che fornisce un insieme di tool per il linguaggio naturale.

Nella figura 16 un esempio di come lavora questo software, tratto dal sito http://corenlp.run/ che propone una demo.

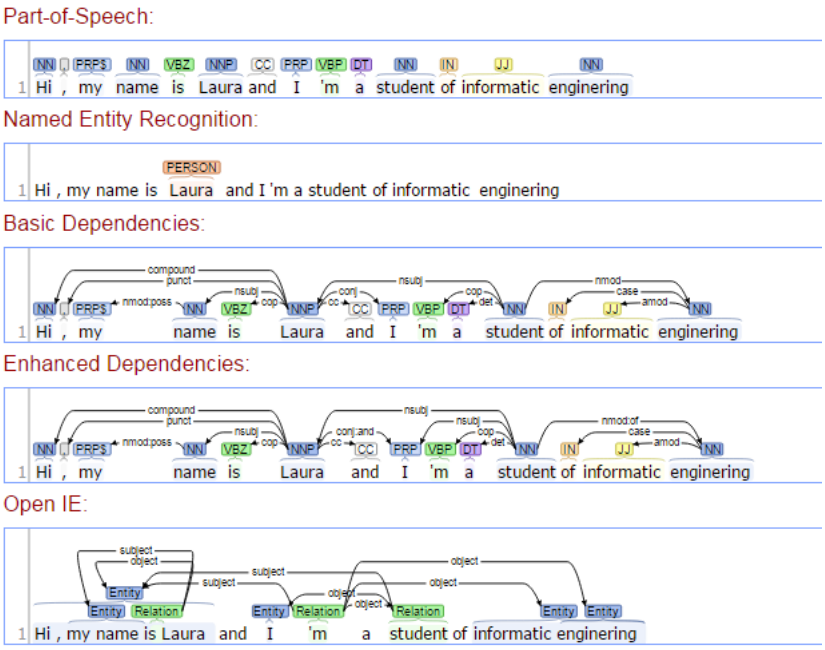

Figura 16: Demo Stanford CoreNLP

A differenza di NLKT, il quale contiene tutte le funzioni necessarie, se si vuole utilizzare le funzioni di classificazione bisogna scaricare il tool Stanford Classifier. Anche quest'ultimo risulta più difficile da utilizzare per un normale utente rispetto a Babelnet per le stesse ragioni precedentemente spiegate.

3.4 Introduzione Cogito

Cogito è una tecnologia in grado di svolgere l'analisi semantica di documenti testuali. Attraverso il programma Cogito Studio, che utilizza questa tecnologia, è possibile creare regole che categorizzino gli argomenti trattati o che estraggano informazioni in base al nostro interesse. Questo ambiente di sviluppo è stato ideato per realizzare progetti in cui si può analizzare il testo implementando delle regole. Le regole realizzate si possono testare attraverso la finestra di "Test". Una volta terminato il progetto è possibile esportare la soluzione in modo da poterla utilizzare al di fuori di Cogito Studio. La soluzione comprende un file eseguibile da avviare se si vuole utilizzare il progetto creato precedentemente. Il passaggio di informazioni avviene

tramite file XML. Inoltre il progetto per funzionare ha bisogno di essere collegato ad un servizio chiamato “dispatcher”.

Nel gioco, ad esempio, si vuole sapere quando l’umano sta parlando di una caratteristica del personaggio. Attraverso Cogito Studio è possibile stabilire delle regole che identifichino di quale caratteristica si sta parlando o che estraggano le informazioni fornite dall’umano se rilevanti.

Nel capitolo 6 si approfondirà il funzionamento dell’ambiente di Cogito Studio con le sue caratteristiche e come è stato utilizzato per il nostro progetto.

4. Il robot umanoide Nao

NAO di Aldebaran Robotics è un robot di taglia media con un architettura aperta. Esso viene usato a scopo educativo e di ricerca in molte università. La tecnologia di NAO è stata realizzata per poter essere usata sia da utenti principianti che da più esperti.

Sulla base dei dati riportati in [8] è alto 57.3 cm, largo 27.3 cm e pesa 4.3 kg. Il corpo è realizzato con un materiale plastico speciale ed ha una batteria agli ioni di litio che permette un utilizzo di 90 minuti.

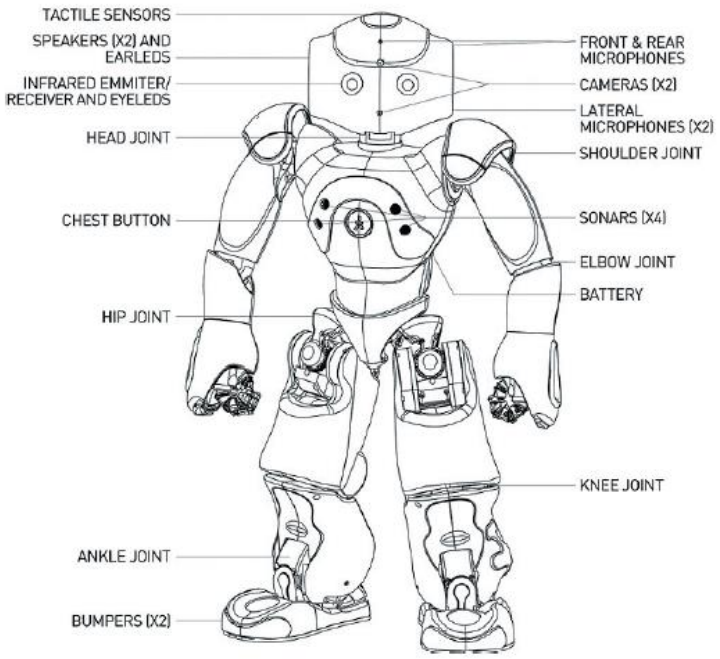

Figura 17 : Caratteristiche Nao

La figura 17 raffigura Nao e le sue caratteristiche.

Nao supporta alcuni linguaggi di programmazione tra cui Python, utilizzato per il nostro progetto.

Nao può comunicare con il PC attraverso un cavo o una rete wireless.

I Robot Nao possono interagire tra loro usando: sensori infrarossi, rete wireless, telecamera, microfono e altoparlanti.

Nel nostro caso non è necessario far comunicare più robot tra loro, ma Nao deve comunicare con un umano.

In questo capitolo si introduce il robot Nao e si affrontano i suoi limiti per quanto riguarda il riconoscimento vocale e la comunicazione.
Nao offre come software di sviluppo Choreographe, un SDK Python o C++ e un software per il monitoraggio chiamato Monitor.
Choreographe è un programma di sviluppo basato sulla grafica ed è facile da usare, utilizza dei box i quali forniscono diverse funzioni prestabilite in cui è possibile modificare i parametri. È possibile collegare vari blocchi in serie e in parallelo e mette a disposizione dei blocchi di controllo come "if" e "while".
NAOqi SDK è il framework di programmazione per programmare Nao ed è realizzato per soddisfare i requisiti necessari in robotica.
Infine Monitor è un software per monitorare lo stato di alcuni componenti del robot.
La programmazione in Choreographe si attua collegando i box tra loro e combinandoli per generare una serie di azioni desiderate. Essi possiedono un input, un output e un pulsante per configurare i parametri.
La struttura del box è formata da Script, Timeline e Diagram:

- Script è la parte di implementazione delle funzionalità del box ed usa il linguaggio di programmazione Python.
- Timeline programma i movimenti di Nao nel tempo.
- Diagram programma le azioni di Nao in base agli eventi.

Sono possibili due tipi di programmazione, quella basata sul tempo e quella basata sugli eventi. La prima può essere usata, per esempio, se vogliamo creare un programma che faccia ballare il robot a tempo di musica. La seconda mette in primo piano la posizione dei box e le loro connessioni.
Per la comunicazione, usando Choregraphe, è possibile utilizzare due box in particolare: "Say" fa parlare il robot con una frase impostata e "Speech Recognition" ascolta in attesa di una parola presente nella lista del box (modificabile a seconda delle esigenze).
Un esempio tratto dal libro [9] nella figura 18:

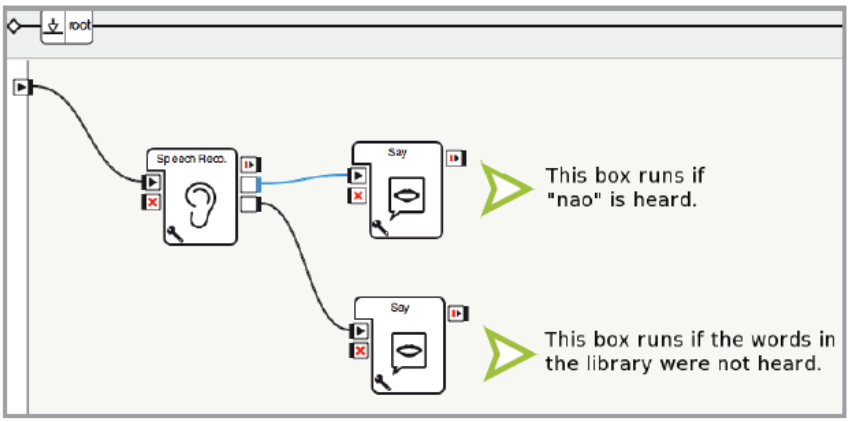

Figura 18 : Box Say e Speech Recognition

Nel caso della figura 18 come si può notare l'unica parola della lista è "nao" e la risposta data dal box "Say" dipende se viene riconosciuta questa parola, in caso contrario risponderà il secondo box.

Se invece vogliamo diversificare la risposta in base alla parola identificata possiamo utilizzare un ulteriore box "Swich case" come nella figura 19 riportata da [9] :

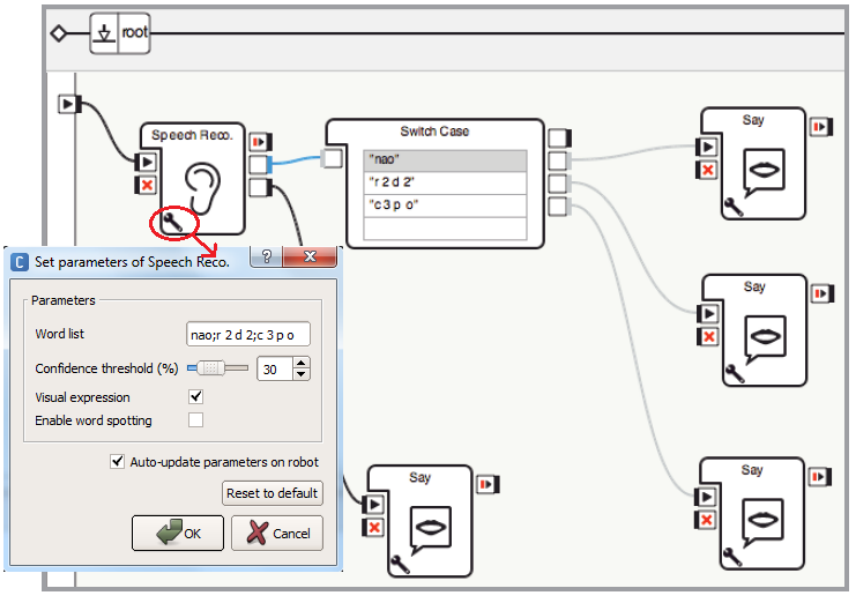

Figura 19 : Box Swich Case

I limiti della comunicazione si vedono attraverso questi esempi.

Il box "Speech Recognition" si aspetta parole ben precise e una volta ricevute può rispondere solo in un determinato modo a seconda dei casi stabiliti.

Nel gioco studiato è vero che le domande e le relative risposte sono le stesse e si potrebbe pensare di strutturare il gioco utilizzando frasi impostate che non possono

variare a seconda della domanda/risposta. In questo modo studiando la casistica si potrebbe arrivare ad una soluzione funzionante ma molto limitata e poco flessibile.
Volendo rendere il gioco il più possibile verosimile a quello giocato tra due umani la possibilità citata sopra è da scartare.
Infatti il programma deve capire e interpretare le domande e risposte formulate in diversi modi dall'umano.
Il problema di Nao è che il riconoscimento vocale è limitato al riconoscimento di parole precise e restituisce la risposta in base a quelle identificate. Infatti la frase pronunciata non è catalogata in base al suo concetto ma solo in base alle parole. In altri termini si potrebbe equiparare questa funzione ad un uguaglianza fra stringhe con l'unica differenza che prima della comparazione il suono ricevuto deve essere convertito in testo.
Come già anticipato i box di Choreographe sono composti principalmente da script in Python. Precedentemente abbiamo visto un esempio di box già pronti come "Say" e "Speech Recognition" utili per il progetto. Però da soli non sono sufficienti, quindi utilizzando il linguaggio di programmazione Python è necessario creare dei box personalizzati che abbiano le funzionalità desiderate.
È possibile sia modificare lo script di un box già esistente o crearne uno nuovo. Durante la creazione di un nuovo box si può impostare l'input, l'output e gli altri parametri. Una volta terminato si dovrà inserire il nuovo box in una libreria e sarà necessario crearne una nuova se non è già stato fatto.
Ora il box è pronto per essere implementato e per poterlo fare si deve aprire la sezione per editare lo script.

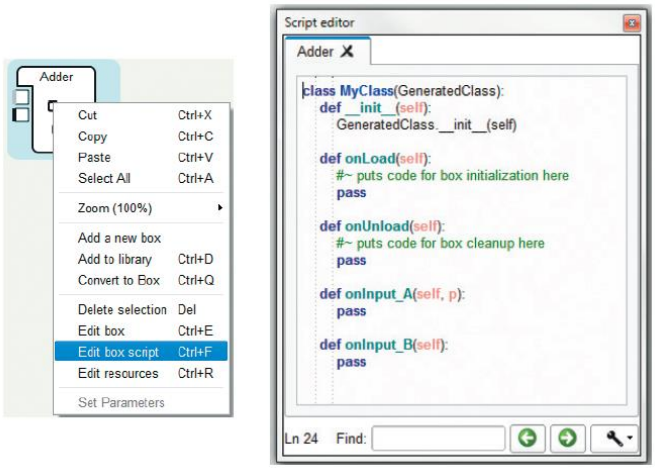

Figura 20 : Creazione Box

L'esempio mostrato nella figura 20 tratto da [8] mostra la creazione del box "Adder" in cui i parametri di input A e B devono essere sommati. Al momento lo script è vuoto ad eccezione della struttura che presenta il costruttore e le funzioni principali da implementare.

Successivamente deve essere implementato come nella figura 21 perché funzioni.

```
1.  class MyClass(GeneratedClass):
2.      def__init__(self)
3.              GeneratedClass__init__(self)
4.              self.bA=False
5.              self.bB=False
6.      def onInput_A(self, p):
7.              s self.R() #~ activate output of the box
8.              self.bA = True
9.              self.A = p
10.             if self.bA and self.bB:
11.             self.process()
12.     def onInput_B(self, p):
13.             self.bB = True
14.             self.B = p
15.             if self.bA and self.bB:
16.             self.process()
17.     def process(self):
18.             self.R(self.A + self.B)
19.             self.bA = False
20.             self.bB = False
```

Figura 21 : Implementazione Box

Sebbene il modello di Choreographe sia molto flessibile, si è preferito utilizzare l'SDK Python di NAOqi, in questo modo non dobbiamo gestire casi come: corse concorrenti ai blocchi, semafori e sincronizzazione di eventi di per sé asincroni.

L’SDK Python NAOqi è un insieme di librerie Python che “wrappano” tutte le azioni del robot (sia di basso livello che di alto livello) e consentono, eseguendo il programma su un pc, di delegare queste azioni al Nao connesso.
Inoltre è anche possibile eseguire questi programmi sul robot stesso, rendendolo del tutto libero dalla presenza di un altro pc nella rete.
Una volta importate le librerie necessarie, si deve implementare lo script utilizzando le funzioni di Google in modo che l'input venga trasformato in testo e passi successivamente attraverso le regole create con Cogito Studio, utilizzando un SDK per la tecnologia di Cogito, in modo da generare un output adeguato. Nel capitolo successivo si approfondirà il procedimento che analizza l'input generato dall'umano in modo da ricondurlo ad una delle possibili domande o risposte.

5. Cogito

Come anticipato nei precedenti capitoli per l'analisi semantica è stata utilizzata la piattaforma Cogito Studio fornita dall'azienda Expert System.

In questo capitolo sarà spiegato il funzionamento di Cogito, le sue caratteristiche principali e infine come è stato impiegato nel nostro progetto.

Cogito è in grado di svolgere un'analisi morfologica, grammaticale, logica e semantica di documenti testuali, creando una mappa concettuale e associando ogni termine con il corrispettivo lemma e il giusto concetto.

Attraverso i lemmi è possibile identificare tutte le forme dello specifico termine, come declinazioni o coniugazioni. Attraverso i concetti si riesce ad evitare fraintendimenti linguistici e associare i giusti sinonimi alla parola considerata, quindi trovare altri modi per esprimere le stesse informazioni.

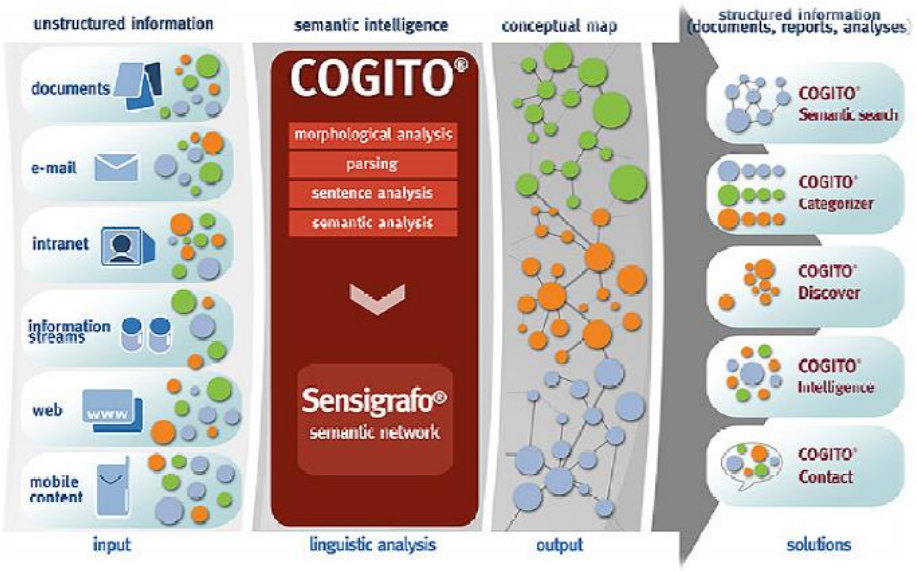

Figura 22 : Cogito

Cogito rende possibile il compito di disambiguazione attraverso una rete semantica chiamata "Sensigrafo" che contiene tutti i concetti e le relazioni tra loro e un motore di disambiguazione che sulla base del Sensigrafo è in grado di associare ogni parola al significato che rappresenta in quel contesto.

Il Sensigrafo è un multi-grafo in cui ogni nodo rappresenta un concetto della lingua modellata che è formato da uno o più lemmi.

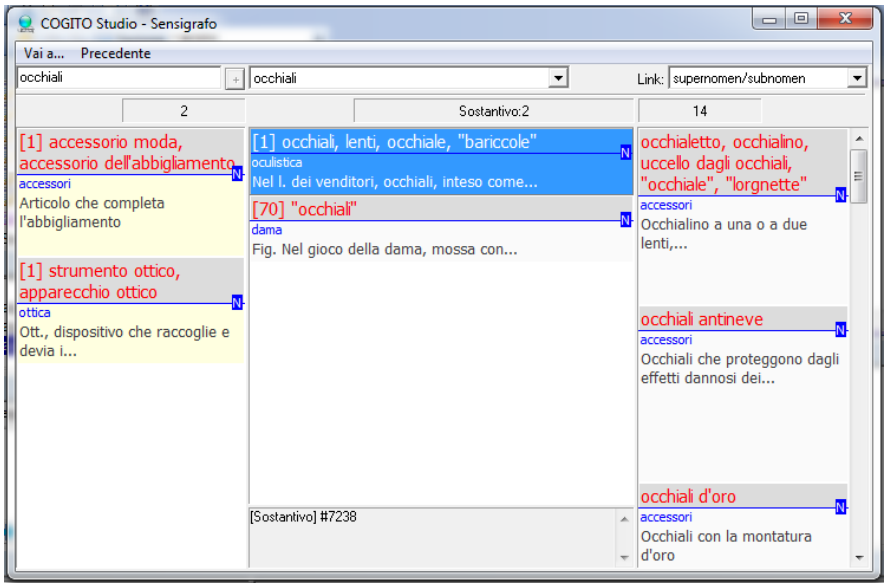

Figura 23: Sensigrafo

La figura 23 mostra un esempio di funzionamento del Sensigrafo se viene ricercata la parola “occhiali”. Selezionando in alto a destra il valore di link a “supernomen/subnomen” , si può notare che a sinistra è presente un elenco di concetti più generali e a destra un elenco di parole più specifiche rispetto a quella cercata. È possibile, in base alla ricerca che si vuole effettuare, visualizzare altri tipi di relazioni modificando il valore di link. Inoltre sotto si vede il numero identificativo del syncon. Il concetto è chiamato “syncon” ed è un insieme di parole che rappresentano lo stesso concetto lessicale, può contenere lemmi singoli o composti (es: non-stop). Un syncon coincide con un nodo della rete semantica ed ognuno di loro è collegato, da una o più relazioni semantiche chiamate “link”. Syncon e link sviluppano una struttura a grafo. Gli elementi principali di un syncon sono: classe (nome, verbo, aggettivo, avverbio), relazioni semantiche (link), spiegazione del significato, dominio e la lista dei lemmi. Cogito Studio è usato per realizzare regole di categorizzazione ed estrazione, gestire i progetti creati in modo da testarli e distribuirli.

Il progetto quindi si divide in due parti: categorizzazione ed estrazione.

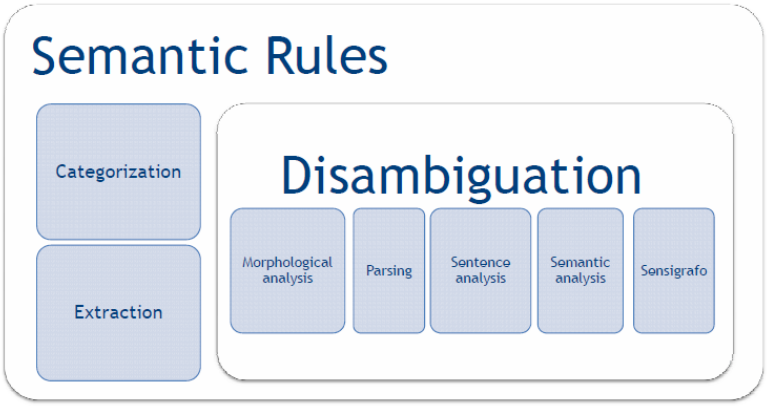

Figura 24: Semantic Rules

È possibile testare il progetto utilizzando il "Test Panel" direttamente sulla piattaforma di Cogito Studio. Una volta terminato si potrà procedere all'estrazione di cui si parlerà in seguito.

La figura 25 rappresenta in blocchi un esempio di workflow per la realizzazione di progetti semantici basati su Cogito Studio.

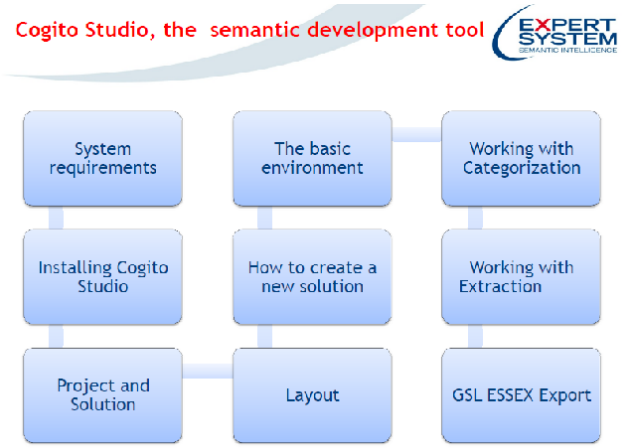

Figura 25 : Workflow Cogito Studio

5.1 La piattaforma

Cogito Studio è uno strumento di sviluppo basato sull'analisi semantica dei testi.

Con Cogito Studio è possibile scrivere, compilare e testare i file realizzati con esso.

L'ambiente di sviluppo include: un pannello di output di disambiguazione, un editor sensibile alla sintassi, il visualizzatore del Sensigrafo, un compilatore delle regole, un

pannello di output per la categorizzazione ed estrazione ed una procedura per creare test corpora (non utilizzata nel nostro progetto).

Un progetto è composto da uno o più file sorgenti contenenti le regole, un Sensigrafo, un "Semantic Disambiguator" e una tassonomia per la categorizzazione (o albero di dominio) e può essere di categorizzazione, estrazione o entrambi.

Ogni progetto è configurato nella lingua che si desidera analizzare, nel nostro caso l'italiano.

Nella finestra di editor è possibile analizzare testi attraverso il pannello <TEST>.

Il layout di Cogito Studio si presenta come nell'immagine 26

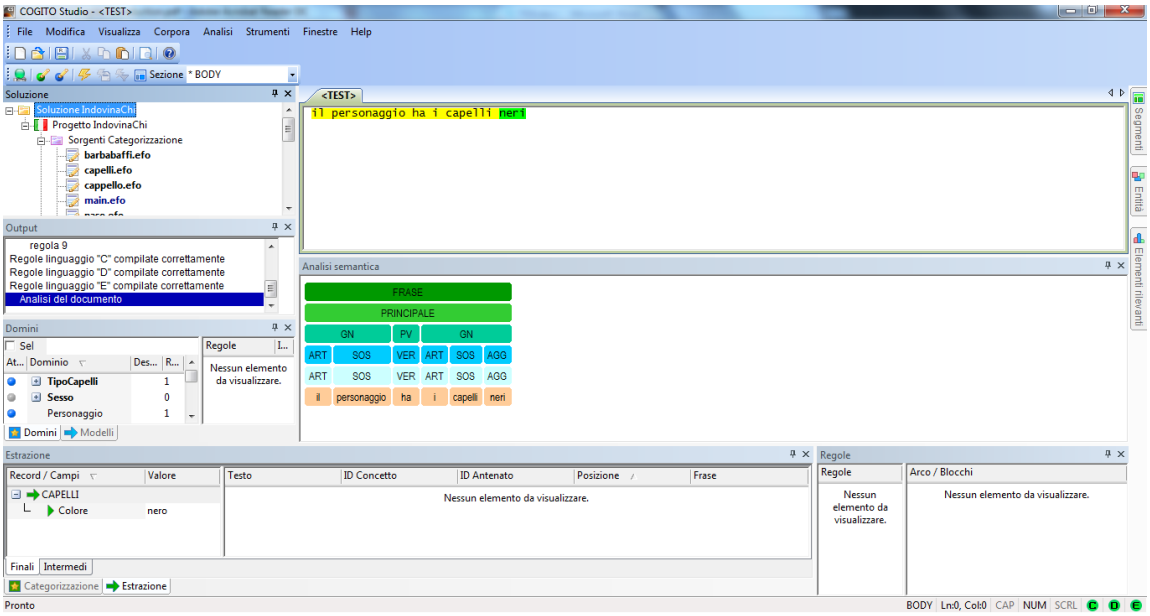

Figura 26: Layout di Cogito Studio

In alto a sinistra il pannello della soluzione, sotto di esso il pannello di output dove è possibile verificare la corretta compilazione del progetto.

Al centro in alto l'editor delle regole e il pannello di test e sotto il pannello con l'output dell'analisi semantica.

La creazione di un progetto è molto semplice, è sufficiente seguire i passi guidati dopo aver selezionato la voce "New Project" nel menu "File".

5.2 Categorizzazione ed estrazione

Per il progetto di categorizzazione la tassonomia è una parte importante di esso.

Le regole di categorizzazione sono realizzate con un linguaggio chiamato “C” , per questo definite “C Rules”, hanno la funzione di realizzare condizioni linguistiche, assegnare un punteggio e creare un collegamento tra il documento e le categorie.
La sintassi è composta da: “SCOPE”, “DOMAIN”, “SCORE” e le condizioni composte da attributi ed operatori.

```
1 //occhiali.efo
2 SCOPE SENTENCE
3 {
4     DOMAIN(Occhiali)
5     {
6         LEMMA("occhiali")
7     }
8 }
9
```

Figura 27: Sintassi categorizzazione

La figura 27 rappresenta un esempio della struttura della sintassi appena descritta.
In particolare “SCOPE” contiene uno o più domini, “DOMAIN” che rappresenta un dominio (in questo caso “occhiali”) contiene le condizioni che rendono valida la regola e ”SCORE” rappresenta il punteggio in percentuale con cui si verificano le condizioni imposte.
In questa parte, il testo viene analizzato e le regole scritte determineranno se il contenuto appartiene ad uno dei domini con il relativo punteggio in percentuale per ognuno di essi.
Gli attributi principali utilizzati nel nostro progetto sono: keyword, lemma, syncon e ancestor. Una keyword è una specifica sequenza di caratteri che possiamo ritrovare in un testo. Un lemma è la forma base di una parola e rappresenta tutte le sue possibili flessioni come il lemma “essere” rappresenta tutte le coniugazioni di esso (ad esempio: sono, sei, è). Un syncon, come definito precedentemente, è l’astrazione semantica di un concetto rappresentato da un insieme di sinonimi intercambiabili a seconda del contesto. Il syncon è rappresentato da un numero identificativo e può essere cercato attraverso il Sensigrafo. Come già visto precedentemente nella figura 19 si nota che il syncon identificato dal numero 7238 (al centro in basso) rappresenta l’insieme di lemmi utilizzati per descrivere il concetto di “occhiali”. Inoltre si

possono vedere i nodi superiori (a sinistra) ed inferiori (a destra) del syncon in questione.

Infine, un ancestor è una catena di concetti collegati tra loro attraverso specifiche relazioni semantiche.

Gli attributi possono essere combinati tra loro attraverso gli operatori logici AND, OR, AND NOT, di posizione >>, >, <> e di relazione &.

L'operatore AND fa in modo che la regola scatti solo se tutti gli attributi legati da esso siano presenti.

```
SCOPE SENTENCE
{
    //capelli
    DOMAIN(capelli)
    {
        KEYWORD("capelli")
        AND
        LEMMA("castano")
    }
}

```

Figura 28: Operatore AND

Ad esempio la regola nella figura 28 rappresentata dal dominio "capelli" scatta soltanto se la frase è composta dalla parola "capelli" e dal lemma "castano", ma se la frase fosse ad esempio "Ho i capelli biondi" la regola non risulterebbe valida.

OR impone che la condizione sia valida per almeno un attributo in modo che la regola dia risultato positivo.

```
SCOPE SENTENCE
{
    DOMAIN(sesso)
    {
        KEYWORD("maschio")
        OR
        KEYWORD("femmina")
    }
}
```

Figura 29: Operatore OR

La regola nell'esempio della figura 29 impone che nella frase deve essere presente almeno una delle due parole "maschio" o "femmina", ma possono essere presenti anche entrambi.

AND NOT impone che il risultato sia positivo solo se è presente il primo attributo ma non insieme ad altri indesiderati.

```
SCOPE SENTENCE
{
    //capelli
    DOMAIN(capelli)
    {
        KEYWORD("capelli")
        AND NOT
        LEMMA("castano")
    }
}
```

Figura 30: Operatore AND NOT

La regola riportata nella figura 30 scatta solo se nella frase è presente la parola "capelli" ma non insieme al lemma "castano", ad esempio la regola è valida per la frase "Io ho i capelli biondi".

A differenza degli operatori AND, OR e AND NOT, gli opertatori >>, > e <> impongono una determinata sequenza e quindi un ordine preciso.

L'operatore >> impone che i due attributi tra esso oltre che essere presenti nella frase devono risultare vicini. La regola "KEYWORD("capelli")>>LEMMA("castano")" scatta per esempio con la frase "ho i capelli castani" ma non è valida per la frase "i miei capelli sono castani", infatti questo operatore non ammette nessuna parola tra i due attributi.

L'operatore > impone una condizione meno stretta rispetto la precedente, tra le due parole può essere presente ad esempio un articolo o una congiunzione. Se scriviamo la regola "LEMMA("avere")>LEMMA("barba")" la regola risulta valida con una frase come "il personaggio ha la barba" anche se tra i due lemmi c'è un articolo.

L'operatore <> stabilisce che il primo e il secondo attributo devono essere presenti nell'ordine stabilito ma tra loro possono esserci altre parole. In questo caso la regola "KEYWORD("capelli")<>LEMMA("castano")" sarebbe valida anche per la frase "Laura ha i capelli biondi e io li ho castani".

Infine l'operatore & stabilisce se è presente la relazione descritta da esso (come &VO verbo-oggetto) tra i due attributi.

```
SCOPE SENTENCE
{
    DOMAIN(cucinare)
    {
        KEYWORD("io")
        &SV
        LEMMA("cucinare")
    }
}
```

Figura 31: Operatore &

La figura 31 è un esempio dell'operatore & in cui la regola è valida per le frasi come "io cucino", "io ho cucinato", "io sto cucinando" perché si aspetta un soggetto seguito da un verbo e in particolare il soggetto deve essere "io" e il verbo deve essere una coniugazione di "cucinare".

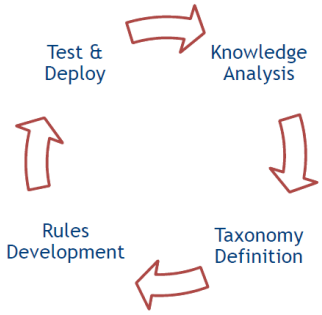

Figura 32: Categorizzazione

Il progetto di estrazione si basa sull'analisi del testo per trarre informazioni utili e quindi estrapolarle ed inserirle in un campo di un "template".

Un "template" è una raccolta di campi i quali possono essere riempiti attraverso le regole, se esse sono soddisfatte. È possibile definire uno o più template nello stesso progetto i quali, possono avere uno o più campi.

Le regole di estrazione sono scritte in un linguaggio chiamato "E" e quindi definite "E Rules".

Queste regole hanno la funzione di stabilire condizioni linguistiche, identificare i template e valorizzare i campi.

La sintassi è composta da: "SCOPE", "IDENTIFY", "TEMPLATE", "FIELD" e le condizioni, come descritto prima, composte da attributi e operatori.

```
//occhi.efe

//Colore occhi
TEMPLATE(OCCHI)
{
    @Colore
}

SCOPE SENTENCE
{
 IDENTIFY(OCCHI)
    {
        LEMMA("personaggio") > LEMMA("avere") > LEMMA("occhio") > @Colore[KEYWORD("azzurri","marroni")]
        OR
        LEMMA("occhio") <> LEMMA("personaggio") > LEMMA("essere") > @Colore[KEYWORD("azzurri","marroni")]
    }
}
```

Figura 33: Sintassi estrazione

La figura 33 rappresenta un esempio della struttura della sintassi per l'estrazione.
"TEMPLATE", in questo caso di nome "OCCHI", racchiude l'insieme dei possibili campi chiamati "FIELD" ed identificati dal loro nome preceduto dal carattere "@".
Un "FIELD" è un campo in cui viene inserita l'informazione estratta nel caso in cui la condizione è valida.
"SCOPE" contiene tutte le possibili regole identificate da "IDENTIFY".
Nell'esempio della figura 33 è presente un solo campo "@Colore" in cui è inserito il colore degli occhi se la condizione è valida, per esempio per la frase "il personaggio ha gli occhi azzurri" il campo "@Colore" avrà il valore "azzurri".
Come per la categorizzazione, l'estrazione ha gli attributi: keyword, lemma, syncon e ancestor. In più ci sono gli attributi: TYPE e PATTERN.
TYPE permette di identificare parole della stessa classe (nome, verbo).

```
TEMPLATE(TEST)
{
    @FIELD1
}

SCOPE SENTENCE
{
 IDENTIFY(TEST)
    {

        @FIELD1[TYPE(NPH)]
    }
}
```

Figura 34: Attributo TYPE

La figura 34 è un esempio in cui il campo FIELD1 assume il valore di un nome proprio nel caso in cui sia presente nella frase, se scrivo "Mi chiamo Laura" il campo FIELD1 assumerà il valore "Laura".

PATTERN permette di identificare una parola o sequenza di parole attraverso uno specifico testo che è interpretato come una espressione regolare.

```
1
2  TEMPLATE(TEST)
3  {
4      @FIELD1
5  }
6
7  SCOPE SENTENCE
8  {
9   IDENTIFY(TEST)
10      {
11
12          @FIELD1[PATTERN("ciao*")]
13      }
14 }
```

Figura 35: Attributo PATTERN

Nella figura 35 l'espressione all'interno di "PATTERN" impone che deve essere presente la parola "cia" seguita dal carattere "o" ripetuto da 0 fino ad N volte, quindi la regola vale sia per "cia" che per "ciaoooo".

Inoltre è possibile caricare una lista esterna di attributi da uno o più file.

Oltre agli operatori descritti in precedenza per l'analisi semantica è possibile utilizzare altri operatori per combinare gli attributi da inserire nel campo.

Gli operatori possibili sono: +, −,ENTRY.

Utilizzando +, l'elemento corrispondente al primo attributo deve essere presente e verificare anche la condizione seguita da +, al contrario non deve verificare le condizioni precedute da −.

```
1
2  TEMPLATE(TEST)
3  {
4      @FIELD1
5  }
6
7  SCOPE SENTENCE
8  {
9   IDENTIFY(TEST)
10      {
11          //ANCESTOR(bevanda)
12          @FIELD1[ANCESTOR(880)+TYPE(NPR)]
13      }
14 }
15
```

Figura 36: Operatore +

L'esempio nell'immagine 36 indica che il campo FIELD1 conterrà un nome proprio appartenente ad un tipo di bevanda, se scriviamo la frase "ho bevuto una Coca-Cola" il campo FIELD1 risulterà uguale a "Coca-Cola" ma la regola non scatta se viene scritta una frase come "ho bevuto un'aranciata".

```
1
2  TEMPLATE(TEST)
3  {
4      @FIELD1
5  }
6
7  SCOPE SENTENCE
8  {
9   IDENTIFY(TEST)
10      {
11          //ANCESTOR(bevanda)
12          @FIELD1[ANCESTOR(880)-TYPE(NPR)]
13      }
14 }
```

Figura 37: Operatore -

Al contrario, nell'esempio dell'immagine 37, prese le frasi precedenti scatterà la regola solo se la bevanda citata non è un nome proprio, quindi solo nella seconda frase il campo FIELD1 sarà "aranciata".

Infine ENTRY è una forma di normalizzazione dell'output, che fornisce in uscita sempre lo stesso lemma per un dato concetto estratto.

```
1
2  TEMPLATE(TEST)
3  {
4      @FIELD1
5  }
6
7  SCOPE SENTENCE
8  {
9   IDENTIFY(TEST)
10      {
11          //SYNCON(viso)
12          @FIELD1[SYNCON(11773)][ENTRY]
13      }
14 }
```

Figura 38: Operatore ENTRY

La figura 38 mostra un esempio in cui la regola per il campo FIELD1 scatta per ogni frase in cui è riportata una o più parole uguali o sinonimi di "viso". Se scrivessimo nello stesso testo la parola "volto" e "viso" il campo FIELD1 in assenza dell'opzione "ENTRY" assumerebbe i valori "volto" e "viso" mentre se fosse presente, il campo avrebbe un unico valore "faccia".

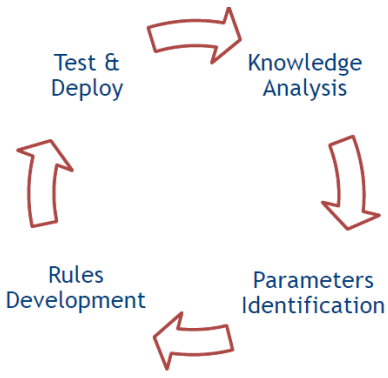

Figura 39: Estrazione

5.3 GSL ESSEX Export

Terminato il progetto è possibile esportarlo per utilizzarlo al di fuori del contesto di Cogito Studio.

L'esportazione avviene attraverso il menu "Strumenti", selezionando la voce "Esporta GSL ESSEX" e seguendo la procedura, eventualmente modificando le impostazioni di esportazione.

ESSEX è una piattaforma software che analizza i documenti.

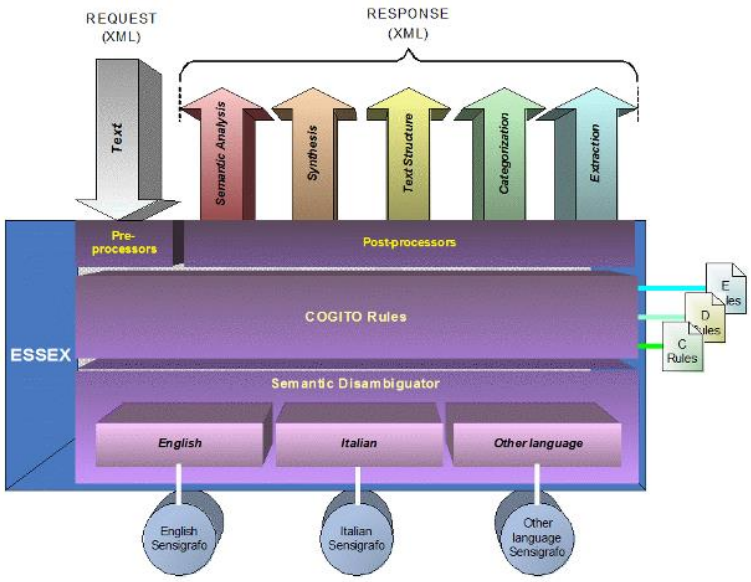

Figura 40: GSL ESSEX Export

La figura 40 rappresenta le funzionalità di ESSEX.

In realtà l'utente non utilizzerà in modo diretto la piattaforma ESSEX ma sarà collegato ad esso attraverso un servizio chiamato DISPATCHER.

Il DISPATCHER è un servizio che deve essere installato sulla macchina in cui si intende utilizzare la propria applicazione per poter comunicare con ESSEX, in questo modo è possibile effettuare anche richieste parallele e su nodi distribuiti. Una volta esportato il progetto si otterrà una cartella in cui è presente un file eseguibile "_execute.bat" il quale si collegherà al servizio del DISPATCHER e resterà in attesa delle richieste da parte dell'utente.

Come si può notare le richieste e le risposte sono formulate attraverso file xml.

5.4 Il progetto “Indovina Chi?”

Per il progetto “Indovina Chi?” è stato necessario implementare sia la parte di categorizzazione, sia la parte di estrazione.

La categorizzazione in questo contesto realizza l’analisi delle domande effettuate dall’umano. Ogni domanda può essere effettuata in modo diverso e le regole sono state scritte in modo da riuscire a comprendere ogni possibilità di formulazione e cercando di evitare che le regole siano valide anche in casi sbagliati. Ad esempio le domande “I capelli del personaggio sono biondi?” o “Il personaggio ha i capelli biondi?” devono essere entrambe valide e far scattare il dominio in cui la regola scatta se la domanda fa riferimento ai capelli biondi.

Quindi è stato necessario utilizzare diverse condizioni che limitassero la possibilità di errore. Sono stati definiti tutti i possibili domini di interesse in modo che ogni domanda, fatta da un giocatore umano, rientrasse in uno di questi.

Sotto un esempio di codice per la regola di categorizzazione la quale individua le domande che riguardano il colore degli occhi. In questo caso la domanda può essere generica come “Di che colore ha gli occhi?” oppure più specifica come “Ha gli occhi azzurri?”. Infatti sono dichiarati tre domini: “ColoreOcchi” per la domanda generica, “OcchiAzzurri” e “OcchiMarroni” per la domanda specifica sul colore degli occhi. Per ogni dominio sono scritte le regole che indicano quali parole e in quale ordine devono essere presenti nella frase affinché sia valido.

```
SCOPE SENTENCE
{
    DOMAIN(ColoreOcchi)
     {
          LEMMA("colore")
          <>
          LEMMA("occhio")
     }
     DOMAIN(OcchiAzzurri)
     {
          LEMMA("occhio")
          <>
          LEMMA("azzurro")
     }
     DOMAIN(OcchiMarroni)
```

```
      {
            LEMMA("occhio")
            <>
            LEMMA("marrone")
      }
}
```

L'estrazione è stata utilizzata per registrare i dati ottenuti dall'umano a seguito di una domanda effettuata dal robot. Anche in questo caso è stato necessario inserire condizioni in modo tale da non estrarre informazioni non inerenti alla domanda. Per esempio le frasi "il personaggio ha i capelli biondi" e "i capelli del personaggio sono biondi" devono dare come risultato di estrazione la parola "biondo" per il campo del colore dei capelli, ma deve saper distinguere quando si parla o meno del personaggio. Ad esempio per la frase "Io ho i capelli castani ma il personaggio ha i capelli biondi" deve essere prelevato solo il colore "biondo" e non anche "castano".

L'estrazione è stata la parte più complicata da implementare perché, oltre alla diversa formulazione di una frase, l'umano ha la possibilità di ingannare il robot con parole che potrebbero risultare ambigue. Per esempio alla domanda "Di che colore ha i capelli il personaggio?" l'umano potrebbe rispondere "Io ho i capelli rossi ma il personaggio li ha neri". In questo caso il riconoscimento del colore dei capelli del personaggio è corretto soltanto se è stata inserita la condizione che l'aggettivo deve essere preceduto dalla parola "personaggio".

Si è cercato di inserire più regole e condizioni possibili per soddisfare le nostre esigenze, cercando di eliminare il più possibile gli errori ma dando la possibilità all'umano di formulare frasi diverse senza che risultino invalide.

In seguito, un esempio di codice per le regole di estrazione delle risposte che riguardano il colore degli occhi. Come si nota la regola non impone solo la presenza dei colori "azzurro" o "marrone" ma anche una serie di vincoli per fare in modo che il robot non sia ingannato dall'umano. Questi vincoli però, come effetto collaterale, stabiliscono valida la frase solo se al suo interno è presente la parola "personaggio" la quale identifica che l'umano sta parlando proprio del gioco. Quindi una frase come

“Ha gli occhi azzurri” non sarà considerata valida mentre è valida la frase “Il personaggio ha gli occhi azzurri”.

```
TEMPLATE(OCCHI)
{
     @Colore
}
SCOPE SENTENCE
{
 IDENTIFY(OCCHI)
     {
          LEMMA("personaggio") > LEMMA("avere") > LEMMA("occhio")
> @Colore[KEYWORD("azzurri","marroni")]
          OR
          LEMMA("occhio") <> LEMMA("personaggio") >
LEMMA("essere") > @Colore[KEYWORD("azzurri","marroni")]
     }
}
```

Utilizzando il GSLN SDK fornito dall’azienda Expert System è stato possibile collegarsi ad ESSEX attraverso il programma implementato in Python. Esso provvede una libreria con le funzioni necessarie per fornire ad ESSEX il file xml di richiesta e quelle per ricevere il file xml di risposta.

```
<REQUEST VERSION="13.1">
  <DOCUMENT TYPE="PLAIN">
    <![CDATA[il personaggio ha i capelli biondi]]>
  </DOCUMENT>
  <ANALYSIS NAME="CLEANDOCUMENT"/>
  <ANALYSIS NAME="CATEGORIZATION"/>
  <ANALYSIS NAME="EXTRACTION"/>
</REQUEST>
```

Figura 41: Esempio Request.xml

Nella figura 41 si può vedere una possibile richiesta di analisi della frase “il personaggio ha i capelli biondi”.

Nella figura 42 si nota che nella risposta la parte di categorizzazione evidenzia due possibili domini, di cui consideriamo solo il primo perché più probabile.

Nella parte di estrazione è stato catturato il colore dei capelli “biondo”.

```
<RESPONSE VERSION="13.6.4" DDIS="13.6.2" LANGUAGE="16" SYNC="CLEAN">
<DOCUMENT><![CDATA[il personaggio ha i capelli biondi
]]></DOCUMENT>
<CATEGORIZATION TYPE="WINNERS">
<DOMAIN NAME="CapelliBiondi" ID="CapelliBiondi"/>
<DOMAIN NAME="TipoCapelli" ID="TipoCapelli"/>
</CATEGORIZATION>
<EXTRACTION>
<RECORD TEMPLATE="CAPELLI" TRACK="0">
<FIELD NAME="Colore" BASE="biondo"/>
</RECORD>
</EXTRACTION>
</RESPONSE>
```

Figura 42: Esempio Response.xml

Come detto nel paragrafo su Google Speech Recognition, non è possibile individuare la presenza del punto interrogativo e quindi capire se si tratta o meno di una domanda. È il programma ad avere il compito di stabilire il turno del gioco.

L'esempio della figura 41 e 42 mostra infatti che non è distinguibile una domanda da una risposta se non implementando uno specifico algoritmo che in base al turno decide quale parte della risposta ricevuta da ESSEX deve essere utilizzata per lo scopo del gioco.

6. Data Base

Il nostro giocatore umanoide di "Indovina Chi?" per poter funzionare ha bisogno del supporto di un database ben strutturato in modo che contenga tutte le informazioni utili e che queste siano di facile reperibilità.

In questo capitolo si introdurrà il concetto di database, di DBMS (Data Base Management System) e del linguaggio SQL utilizzato per interagire con il database stesso.

Un database è semplicemente una collezione di dati, in particolare si riferisce ad un insieme organizzato di informazioni. I database possono avere strutture differenti ma nel nostro caso, come per la maggior parte, è utilizzato il modello relazionale. Questo tipo di modello è basato su tabelle e relazioni tra esse. Altri due tipi di modelli sono quelli di tipo gerarchico e reticolare, essi a differenza del primo utilizzano i puntatori che risultano molto efficienti a discapito però della semplicità.

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 applicazioni.

Il libro [10] dice che un DBMS è un sistema software di gestione per i database. Esso deve garantire la buona gestione delle grandi quantità di dati, fornire meccanismi per l'affidabilità dei dati, per il controllo degli accessi e della concorrenza. L’utilizzo del file system è sconsigliato perché non ha molti dei servizi offerti da un DBMS e la condivisione risulta molto limitata, come la modifica contemporanea dello stesso file.

La condivisione di un database risulta, oltre che comoda, molto vantaggiosa perché ogni applicazione può utilizzare la stessa raccolta di dati evitando un inutile spreco di risorse e ripetizioni che spesso portano ad errori dovuti a copie non aggiornate. Ad esempio nel nostro progetto il database potrebbe essere utilizzato anche in un'altra applicazione. Infatti il programma è stato implementato in Python per poter essere utilizzato con il robot Nao, ma se si decidesse di realizzare lo stesso gioco in un linguaggio di programmazione differente sarebbe possibile utilizzare lo stesso database.

Un DBMS fornisce diversi linguaggi per l'interazione con i database, in particolare si possono suddividere in tre classi: DDL (Data Definition Language), DML (Data Manipulation Language) e DCL (Data Control Language).
SQL è un linguaggio di programmazione ideato per la gestione dei database e racchiude tutti e tre i linguaggi citati.
DDL è utilizzato per definire gli schemi, ovvero la descrizione della struttura dei dati.
DML serve per modificare o interrogare le istanze del database, quindi i dati veri e propri.
DCL racchiude i comandi come le autorizzazioni e il controllo degli accessi.
Nel progetto che è stato realizzato sono stati utilizzati i linguaggi DDL e DML ma non è stato effettuato il controllo sull'accesso al database anche se è possibile implementarlo se si vuole garantire una maggiore sicurezza ai dati o per esempio consentire l'accesso al gioco solo ad utenti registrati.
Il modello relazionale, come anticipato, fa uso di tabelle.
Ogni tabella è composta da colonne chiamate "attributi" e righe chiamate "tuple". Ogni tupla rappresenta un oggetto specifico della tabella.
Una tabella nel contesto dei database è chiamata "relazione".
Prima di creare un database generalmente viene realizzato un modello chiamato ER ovvero Entity-Relationship. Questo modello è utilizzato per la progettazione concettuale del database, soprattutto quando la sua struttura è molto complessa. Come descritto in [10] i componenti base del modello ER sono: entità, associazione e attributo.
Un'entità è un'insieme di oggetti con caratteristiche in comune, è rappresentata graficamente da un rettangolo e può considerarsi equivalente ad una tabella del database.
Un'associazione è un legame tra due o più entità, graficamente è rappresentata da un rombo e anche questa può essere considerata una tabella. L'associazione si differenzia da una entità perché essa è un sottoinsieme delle istanze delle entità legate dall'associazione.

Infine un attributo è una proprietà di un'entità o di un'associazione.

Tra un'entità e un'associazione c'è un vincolo chiamato cardinalità che consiste in due numeri che specificano il numero minimo e massimo con cui un'istanza di un'entità può partecipare all'associazione.

Nel gioco "Indovina Chi?" è stato realizzato un database chiamato "IndovinaChi" nel quale sono state realizzate tre tabelle: Personaggi, PersonaggiIndovinati, Partite.

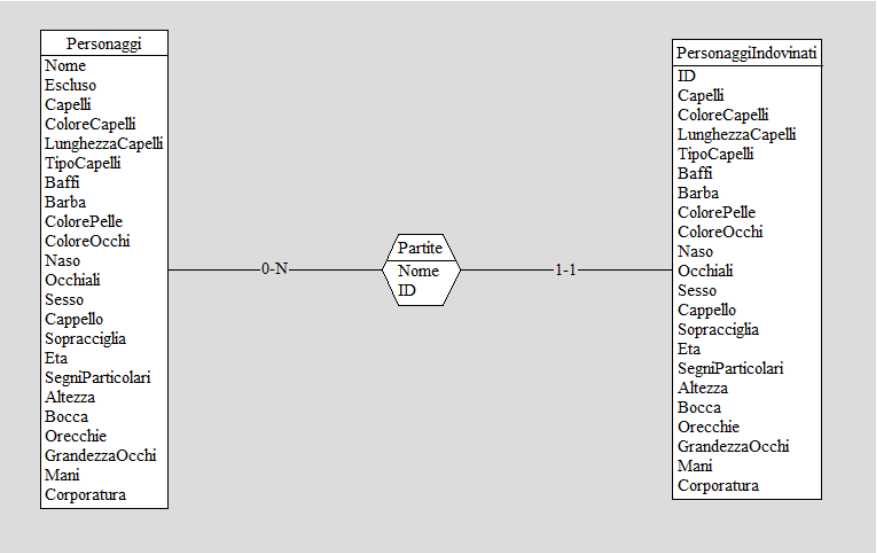

Figura 43: Modello ER

La figura 43 mostra il modello ER del database del progetto. Nel nostro caso è molto semplice in quanto sono necessarie solo tre tabelle di cui "Partite" costituisce l'associazione tra le altre due.

Ogni ID della tabella PersonaggiIndovinati dovrà comparire nella tabella Partite in modo da identificare quale partita si stava giocando.

La tabella Personaggi contiene l'elenco di tutti i 24 personaggi del gioco con le rispettive caratteristiche. La tabella è formata dagli attributi: Nome, Escluso, Capelli, ColoreCapelli, LunghezzaCapelli, TipoCapelli, Baffi,Barba, ColorePelle, ColoreOcchi, Naso, Occhiali, Sesso, Cappello, Sopracciglia, ColorePelle, Eta, SegniParticolari, Altezza, Bocca, Orecchie, GrandezzaOcchi, Mani, Corporatura.

Per completezza sono stati inseriti anche attributi non utilizzati per la finalità del gioco, quindi nel progetto per questi ultimi è stato impostato il valore nullo.

La figura 44 mostra la tabella “Personaggi” con tutti i personaggi e le rispettive caratteristiche.

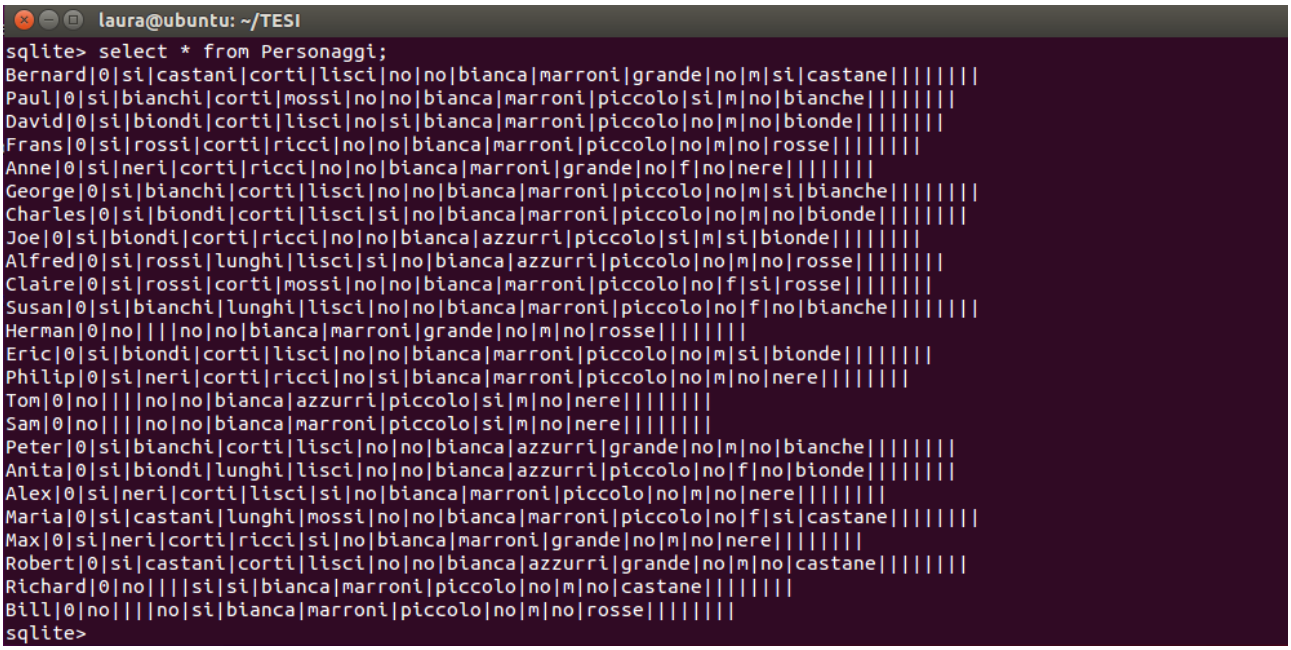

Figura 44: Tabella Personaggi

Questa tabella è stata realizzata in modo che il robot potesse scegliere, in modo casuale, uno dei personaggi disponibili e tramite interrogazioni confrontare le informazioni ricevute dall’umano con quelle dei personaggi o prelevare la risposta corretta a seconda della domanda posta dall’umano.

Il campo “Escludi” è stato inserito in modo da escludere i personaggi che il robot ha provato ad indovinare o quelli che hanno un attributo per cui la risposta dell’umano è risultata negativa.

La tabella PersonaggiIndovinati contiene gli stessi attributi della precedente ad eccezione dei primi due “Nome” ed “Escluso” e della presenza dell’attributo “ID” il quale rappresenta l’identificativo della partita giocata ed è un numero intero crescente.

Questa tabella è utilizzata dal robot per immagazzinare le informazioni ricevute dall’umano in modo da confrontarle con la tabella Personaggi ed escludere i personaggi che non soddisfano i requisiti.

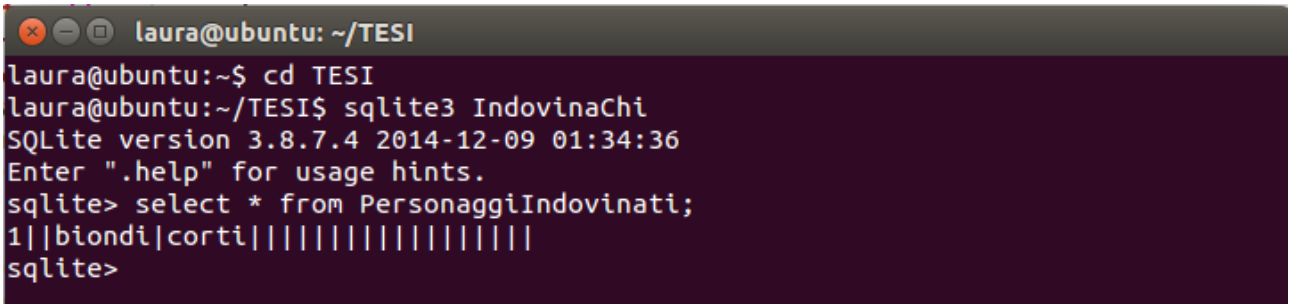

Figura 45: Tabella PersonaggiIndovinati

La figura 45 rappresenta un esempio in cui il robot è a conoscenza che il personaggio da indovinare deve avere i capelli corti e biondi, di conseguenza escluderà tutti quelli che non hanno queste caratteristiche.

Per ogni partita il robot aggiorna solamente l'ultima riga della tabella con le informazioni ricevute e confronta solo quest'ultima con i personaggi della prima tabella.

Infine la tabella Partite ha gli attributi: Nome e ID.

Questa tabella rappresenta la relazione tra le due tabelle precedenti e indica, per ogni partita, il personaggio che il robot ha scelto affiancato dall'ID della partita corrispondente nella seconda tabella.

Per interagire con le tabelle è stato necessario effettuare interrogazioni o modifiche che sono state implementate tramite il linguaggio SQL.

Come specificato in [10] il linguaggio SQL è un linguaggio dichiarativo, ovvero non impone la sequenza delle operazioni.

Le fasi del gioco che comprendono il linguaggio DML, cioè di modifica del database, sono tre. La prima fase è ad inizio partita e consiste nella modifica della tabella "PersonaggiIndovinati" in cui sarà inserita una nuova riga inizialmente con valori nulli ad eccezione del valore intero ID.

Il codice SQL per l'inserimento della nuova riga nella tabella PersonaggiIndovinati:

```
insert into
PersonaggiIndovinati(Capelli,ColoreCapelli,LunghezzaCapel
li,TipoCapelli,Baffi,Barba,ColorePelle,ColoreOcchi,Naso,O
cchiali,Sesso,Cappello,Sopracciglia) values
(null,null,null,null,null,null,null,null,null,null,null,n
ull,null)
```

La seconda fase è il momento in cui il robot deve registrare nel database le informazioni ricavate dalle domande poste all'umano. In caso di risposta affermativa o di conoscenza di una nuova informazione la tabella "PersonaggiIndovinati" dovrà essere aggiornata.

Il codice SQL per l'aggiornamento delle caratteristiche del personaggio è:

```
update PersonaggiIndovinati set "+Caratteristica+" =
'"+NuovoDato+"' where ID=(SELECT Max(ID) FROM
PersonaggiIndovinati)
```

In caso di risposta negativa il campo "Escluso" della tabella "Personaggi" dovrà essere modificato se in contrasto con la caratteristica per cui il robot ha domandato.

Il codice SQL che aggiorna il campo "escluso" è:

```
update Personaggi set escluso=1 where "+Caratteristica+"
<> '"+dato+"
```

L'ultima fase di modifica è effettuata alla fine del gioco, la tabella "Partite" dovrà essere aggiornata con l'inserimento di una nuova riga che rappresenta la partita appena conclusa.

Le fasi del gioco che usano il linguaggio DDL, in cui si richiede al database una informazione, sono tre.

La prima interrogazione al database è fatta dal robot per la scelta di un personaggio casuale tra quelli proposti nella tabella "Personaggi".

Il codice SQL per la scelta random del personaggio è:

```
SELECT Nome FROM Personaggi ORDER BY RANDOM() LIMIT 1
```

La seconda richiesta che il robot effettua è quella inerente alla domanda posta dall'umano. L'interrogazione dovrà selezionare la caratteristica per cui l'umano ha domandato e rispondere dopo averla ricavata dalla tabella "Personaggi".

Il codice SQL per ottenere la risposta da dare all'umano è:

```
SELECT "+tipoD[0]+" FROM Personaggi WHERE
Nome='"+nomeP+"
```

Dove "tipoD[0]" contiene la caratteristica del personaggio per cui vogliamo conoscere le informazioni.

La terza interrogazione serve per individuare il numero di personaggi possibili da prendere in considerazione in base alle informazioni ricevute.

Questa selezione dovrà tenere conto dell'ultima riga della tabella "PersonaggiIndovinati" e del campo "Escluso" della tabella "Personaggi". Questo numero è utile per ricavare la probabilità di successo in caso che il robot dovesse provare ad indovinare il nome del personaggio tra quelli possibili.

Il codice SQL per ricavare il numero di possibili personaggi è:

```
SELECT COUNT(P.Nome) FROM Personaggi P,
PersonaggiIndovinati I WHERE (P.Capelli=I.Capelli OR
I.Capelli IS NULL) AND (P.ColoreCapelli=I.ColoreCapelli
OR I.ColoreCapelli IS NULL) AND
(P.LunghezzaCapelli=I.LunghezzaCapelli OR
I.LunghezzaCapelli IS NULL) AND
(P.TipoCapelli=I.TipoCapelli OR I.TipoCapelli IS NULL)
AND (P.Baffi=I.Baffi OR I.Baffi IS NULL) AND
(P.Barba=I.Barba OR I.Barba IS NULL) AND
(P.ColorePelle=I.ColorePelle OR I.ColorePelle IS NULL)
AND (P.ColoreOcchi=I.ColoreOcchi OR I.ColoreOcchi IS
NULL) AND (P.Naso=I.Naso OR I.Naso IS NULL) AND
(P.Occhiali=I.Occhiali OR I.Occhiali IS NULL) AND
(P.Sesso=I.Sesso OR I.Sesso IS NULL) AND
(P.Cappello=I.Cappello OR I.Cappello IS NULL) AND
(P.Sopracciglia=I.Sopracciglia OR I.Sopracciglia IS
NULL) AND P.Escluso=0 AND I.ID=(SELECT Max(ID) FROM
PersonaggiIndovinati)
```

6.1 SQLite

SQLite è una libreria software che implementa un DBMS SQL. La documentazione riportata in [11] dice che il codice sorgente è di dominio pubblico ed è di libero utilizzo per qualsiasi scopo.

La libreria è compatta e con tutte le sue caratteristiche occupa circa 500KiB.

Di seguito sono riassunte le caratteristiche principali tratte da [11].

SQLite non ha bisogno di essere installato prima dell'uso e non dispone di un processo server separato da avviare, fermare o configurare. Quindi il processo che vuole accedere al database legge e scrive direttamente dal file. Per creare un database

non è necessario alcun amministratore o permessi speciali. Lo svantaggio di SQLite è la non previsione della gestione dei permessi di accesso.
Tutto il database con tabelle, indici, trigger e viste è contenuto in un unico file. Questo lo rende di facile utilizzo e comodo da poter spostare semplicemente copiando il singolo file. Inoltre il formato del file è multipiattaforma, ovvero può essere utilizzato in macchine con architetture differenti.
Generalmente per ogni colonna è deciso un tipo di dato prestabilito al momento della creazione della tabella. SQLite rende questa regola più flessibile, infatti è possibile assegnare ad ogni campo qualsiasi tipo di dato ad eccezione di alcune colonne come le chiavi primarie di tipo intero.
Un'altra particolarità che differenzia SQLite è quella dell'allocazione delle variabili, se ad esempio si assegna ad un campo VARCHAR(100) un solo carattere SQLite gli assegnerà solo un byte e non 100 byte come potrebbe accadere generalmente.

7. Script Python

In questo capitolo si parla dello script realizzato in Python per unire tutte le parti del nostro progetto precedentemente spiegate e in particolar modo dell'algoritmo attraverso cui il robot riesce ad indovinare il personaggio scelto dall'umano.

Sono stati realizzati tre script in Python: IndovinaChi.py, categorizzazione.py ed estrazione.py.

Il file IndovinaChi.py è lo script principale ed è quello utilizzato per lanciare il programma.

categorizzazione.py ed estrazione.py sono script in cui sono state implementate delle funzioni utilizzate da quello principale, implementano le funzioni utili per l'estrazione e la categorizzazione realizzate precedentemente e altre funzioni utili allo scopo.

Il programma può essere suddiviso in quattro parti principali: inizializzazione, primo turno, secondo turno e fine della partita.

```
Inizializzazione()
While(!fine)
    Turno1
    Turno2
Fine()
```

La parte di inizializzazione consiste nel creare le variabili utili nei passaggi successivi. La prima cosa, dopo l'importazione di tutte le librerie necessarie, è l'esecuzione del file NuovaPartita.py che aggiunge una nuova riga alla tabella "PersonaggiIndovinati". È stata creata una variabile "personaggio" in cui è stato inserito il nome del personaggio scelto dal robot. La scelta è stata effettuata in modo casuale utilizzando una query in SQL alla tabella "Personaggi". Un vettore di 29 possibili domande è stato inizializzato in modo che il robot possa proporle all'umano.

Il primo e secondo turno sono racchiusi in un ciclo "while" in modo che non vengano interrotti ad eccezione della fine della partita a causa della vittoria di uno dei due giocatori.

Nel primo turno sarà il robot a fare una domanda all'umano, nel secondo turno sarà l'umano a porre una domanda al robot.

7.1 Turno 1 e algoritmo di matching

Prima di porre la domanda il robot deve verificare che la probabilità di indovinare sia minore di 0,3. Se la probabilità è inferiore varrà selezionata in modo casuale una domanda dal vettore e proposta all'umano, altrimenti sarà selezionato un nome random tra quelli possibili e il robot proverà ad indovinare.

Il codice Python in cui il robot decide se provare o meno ad indovinare è:

```
if probabilita()<0.3:
    rand=random.randrange(0,len(domande))
    dom=d[domande[rand]]
    del domande[rand]
else:
    personaggioRand=nomeRandom()
    dom="e' "+personaggioRand+"?"
```

Dove "dom" è la stringa a cui è assegnata la domanda che il robot deve fare all'umano e "d" è il vettore con tutte le possibili domande che può fare. Una volta scelta la domanda il programma cancella il numero corrispondente dal vettore "domande" in modo che non sia ripetuta.

La funzione "probabilità" serve per determinare la probabilità di successo del robot nel caso in cui provasse ad indovinare il personaggio dell'umano. Facendo un interrogazione al database si ottiene il numero di personaggi possibili e si ricava la probabilità con la formula 1/(n° personaggi possibili).

Il codice Python è:

```
def probabilita():
     conn=sqlite3.connect('IndovinaChi')
     cursor = conn.cursor()
     query = "SELECT COUNT(P.Nome) FROM Personaggi P,
PersonaggiIndovinati I WHERE (P.Capelli=I.Capelli OR I.Capelli IS
NULL) AND (P.ColoreCapelli=I.ColoreCapelli OR I.ColoreCapelli IS
NULL) AND (P.LunghezzaCapelli=I.LunghezzaCapelli OR
I.LunghezzaCapelli IS NULL) AND (P.TipoCapelli=I.TipoCapelli OR
I.TipoCapelli IS NULL) AND (P.Baffi=I.Baffi OR I.Baffi IS NULL)
AND (P.Barba=I.Barba OR I.Barba IS NULL) AND
(P.ColorePelle=I.ColorePelle OR I.ColorePelle IS NULL) AND
(P.ColoreOcchi=I.ColoreOcchi OR I.ColoreOcchi IS NULL) AND
(P.Naso=I.Naso OR I.Naso IS NULL) AND (P.Occhiali=I.Occhiali OR
I.Occhiali IS NULL) AND (P.Sesso=I.Sesso OR I.Sesso IS NULL) AND
(P.Cappello=I.Cappello OR I.Cappello IS NULL) AND
(P.Sopracciglia=I.Sopracciglia OR I.Sopracciglia IS NULL) AND
P.Escluso=0 AND I.ID=(SELECT Max(ID) FROM PersonaggiIndovinati);"
     cursor.execute(query)
     dato=cursor.fetchall()
     if float(dato[0][0])==0:
          print "errore dato=0!"
     prob= 1/float(dato[0][0])
     return prob
```

La risposta dell'umano è analizzata dalla funzione "estrai" che restituisce un vettore con tre valori che rappresentano la caratteristica (es: capelli), quale parte di essa è analizzata (es: colore) e il valore che assume (es: biondo). La risposta appena ottenuta viene inserita nel database con la funzione "inserisci" in cui è necessario passare il vettore appena ottenuto. Il caso di una affermazione "sì" o "no" è gestito diversamente poiché prima di inserire la risposta dobbiamo essere a conoscenza a quale caratteristica fa riferimento. Quindi è stata realizzata una funzione "inserisciAffermazione" che in ingresso riceve, oltre la risposta, la domanda di riferimento precedentemente trasformata in un vettore, dalla funzione "tipoDomanda", contenente le informazioni necessarie. Questa funzione ha lo stesso compito della precedente ad eccezione che prima deve analizzare la domanda del robot per poter inserire correttamente l'informazione ricevuta nel database.

Se la risposta alla domanda in cui il robot tenta di indovinare il personaggio è positiva il programma esce dal ciclo e il robot comunica di aver vinto.

In seguito l'algoritmo del primo turno in pseudocodice.

```
if( turno1)
    if( probabilità<0.3)
        proponi domanda random
    else
        prova a indovinare il personaggio
        if( personaggio giusto)
            fine= 1
    attendi risposta
    estrai risposta
    inserisci risposta nel database
```

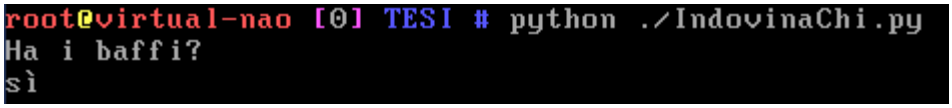

Figura 46: Esempio turno 1 (risposta positiva)

L'immagine 46 mostra un esempio del primo turno, il robot chiede se il personaggio dell'avversario ha i baffi e l'umano risponde di sì. Nell'immagine 47 si vede come cambia la tabella "PersoanngiIndovinati" dopo l'inserimento del nuovo dato, il campo "baffi" ora ha il valore "sì".

Figura 47: Modifica tabella PersonaggiIndovinati

In caso di risposta negativa, come nella figura 48, allora il campo "Escluso" della tabella "Personaggi" è modificato dal valore 0 al valore 1 come mostrato in figura 49.

Figura 48: Esempio turno 1 (risposta negativa)

Nella figura 49 si può vedere che solo Maria, Susan, Claire e Anita hanno il primo campo di valore 0 mentre gli altri sono stati esclusi.

Figura 49: Modifica campo "Escluso"

Se il robot tenta di indovinare e vince, come in figura 50, prima di terminare la partita riferisce al giocatore qual era il suo personaggio.

Figura 50: Robot tenta di indovinare

7.2 Turno 2

Nel secondo turno è l'umano a dover porre una domanda al robot. Questa parte risulta più semplice perché il robot non può ingannare l'umano, quindi attraverso la funzione "rispondi" viene analizzata la domanda inserendo come parametri il personaggio scelto dal robot e la domanda stessa e si ottiene la risposta desiderata dopo aver interrogato il database. Se l'umano tenta di indovinare il personaggio allora il robot, in caso positivo, comunica che l'umano ha vinto.

L'ultima parte del programma consiste nella fine della partita. La tabella Partite deve essere aggiornata con l'inserimento dell'ID dell'ultima riga della tabella PersonaggiIndovinati affiancato dal nome del personaggio che il robot ha scelto nella partita appena terminata.

Sotto l'algoritmo del secondo turno in pseudocodice.

```
if( turno2 )
    attendi domanda
```

```
    categorizza domanda
    rispondi
    if ( umano indovina personaggio)
        fine=1
```

La figura 51 mostra un esempio di come si svolge il turno 2.

Figura 51: Esempio turno 2

Se l'umano tenta di indovinare, nel caso di risposta corretta il robot si comporta come in figura 52.

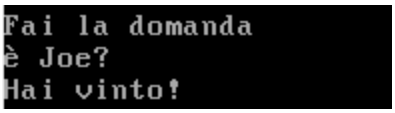

Figura 52: Umano tenta di indovinare

Infine la partita è registrata nella tabella "partite" come in figura 53.

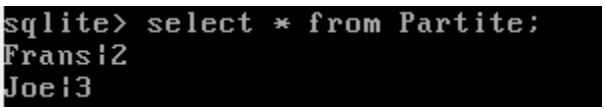

Figura 53: Esempio tabella "Partite"

8. Conclusioni e sviluppi futuri

La tesi è stata incentrata sul gioco Indovina Chi per l'identificazione da parte del robot Nao di un personaggio tramite la sua descrizione. In particolare la descrizione avviene tramite domande e risposte.

Lo scopo del nostro progetto è stato quello di riuscire a far giocare al robot umanoide Nao una partita del gioco Indovina Chi contro un umano tramite il riconoscimento vocale. Quindi, per un risultato positivo, l'umano non dovrebbe accorgersi della differenza tra il robot e una qualsiasi persona.

Il nostro progetto è stato suddiviso in parti e successivamente unite per il suo funzionamento.

Per realizzare la parte di analisi e comprensione del linguaggio naturale è stato utilizzato il programma Cogito Studio fornito dall'azienda Expert System. È stato quindi necessario approfondire il natural language processing e il linguaggio naturale per poter sfruttare al meglio le potenzialità di Cogito.

Per il riconoscimento vocale si è deciso di utilizzare Google Speech Recognition.

È stato inoltre realizzato un database, contenente tutte le informazioni necessarie, composto da tre tabelle. Una tabella con tutti i personaggi del gioco, una con le informazioni ricavate dal robot e infine una con tutte le partite giocate.

Infine, come anticipato, è stato realizzato uno script in Python in modo da unire tutte le precedenti parti e formulare l'algoritmo con cui il robot gioca e tenta di vincere il gioco. Si è sviluppata una funzione che calcola la probabilità di successo che il robot avrebbe nel caso in cui tentasse di indovinare il personaggio dell'avversario. In questo modo se la probabilità calcolata è maggiore di 0.3 il robot può provare ad indovinare, altrimenti deve proporre all'avversario un'altra domanda.

Il gioco deve poter funzionare tramite riconoscimento vocale, per questo è stata utilizzata Google Speech Recognition con cui è stato possibile trasformare le frasi formulate dall'umano in testo in modo da poter essere analizzato dalle regole scritte precedentemente con Cogito Studio.

I risultati del nostro progetto sono stati positivi poiché è stato possibile giocare ad "Indovina chi?" come tra due umani. Facendo varie prove è stato riscontrato che la percentuale di vincita del robot è circa del 50%.

Il tempo di risposta del robot rilevato è minore di 5 secondi, questo risultato è buono considerando il tempo di risposta che si avrebbe giocando con un altro umano.

Le differenze che si possono notare rispetto ad un umano sono che nelle risposte aperte è necessario inserire la parola "personaggio" in modo che il programma rilevi le giuste informazioni senza farsi ingannare e che nonostante tutti i vincoli imposti non si esclude la possibilità che l'umano formuli una frase non correttamente interpretata.

Il programma è stato sviluppato per poter giocare ad "Indovina chi?" ma potrebbe essere utile modificarlo per poterlo impiegare in diversi ambiti in cui è necessario identificare delle persone. Infatti le tabelle nel database sono state predisposte in modo da poter aggiungere caratteristiche fisiche di una qualsiasi persona anche se non sono state utilizzate nel nostro progetto, per esempio l'altezza o il colore della pelle.

Questo progetto, in futuro, potrebbe essere utilizzato per la divulgazione nelle scuole e quindi per essere studiato e modificato. Il programma è stato realizzato per il gioco "Indovina chi?" ma è possibile utilizzarlo per realizzare altre applicazioni simili ma per un diverso scopo. Il database in cui sono contenuti i personaggi con le relative informazioni potrebbe essere sostituito da uno più vasto in cui le persone sono identificate tramite una grande lista di particolarità in base a quanto si vuole rendere precisa la ricerca.

Bibliografia

[1] Alattab, Ahmed Abdu, and Sameem Abdul Kareem. "Semantic Features Selection and Representation for Facial Image Retrieval System." *Intelligent Systems Modelling & Simulation (ISMS), 2013 4th International Conference on*. IEEE, 2013.

[2] Turing, Alan M. "Computing machinery and intelligence." *Mind* (1950): 433-460.

[3] Lee, Chin-Hui, Frank K. Soong, and Kuldip Paliwal, eds. *Automatic speech and speaker recognition: advanced topics*. Vol. 355. Springer Science & Business Media, 2012.

[4] Wikipedia contributors. "Deep learning." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia.

[5] Sak, Haşim, et al. "Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition." *arXiv preprint arXiv:1507.06947* (2015).

[6] Russell, Stuart J., and Peter Norvig. *Intelligenza artificiale. Un approccio moderno*. Vol. 2. Pearson Italia Spa, 2005.

[7] Allen, James F. "Natural language processing." (2003): 1218-1222.

[8] Kisung, Seo. *Using Nao: Introduction to Interactive Humanoid Robots*. Aldebaran Robotics & NT Research, INC.

[9] Beiter, Mike, Brian Coltin e Somchaya Liemhetcharat. *An Intro To Robotics With NAO*. Aldebaran Robotics.

[10] P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone. *Basi di Dati: modelli e linguaggi di interrogazione.* Terza edizione. McGraw-Hill Italia, 2009.

[11] Documentazione SQLite: https://www.sqlite.org

[12] Documentazione Python: https://www.python.org

[13] Expert System: http://www.expertsystem.com

[14] Documentazione Prolog: http://www.swi-prolog.org/

[15] Google Speech Recognition : https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API

[16] Documentazione Speech Recognition in Python: https://pypi.python.org/pypi/SpeechRecognition

[17] Documentazione Babelnet: http://babelnet.org/

[18] Documentazione NLTK: http://www.nltk.org/

[19] Documentazione Stanford NLP Group: http://nlp.stanford.edu/software/

[20] Documentazione WordNet: http://wordnet.princeton.edu/

Ringraziamenti

Desidero ringraziare tutti coloro che mi hanno aiutato e supportato nella realizzazione della tesi e durante tutto il percorso di studi che ho affrontato.

Ringrazio la professoressa Paola Mello, relatore, e Thomas Bridi, correlatore, per avermi dato la possibilità di svolgere una tesi davvero interessante e della loro disponibilità ogni volta in cui ne avevo bisogno.

Ringrazio l'azienda Expert System che mi ha fornito tutto il materiale indispensabile e fondamentale per lo svolgimento di questo progetto e, in particolare, Andrea Asta per il suo supporto.

Ringrazio i miei genitori che mi hanno permesso di affrontare questo percorso e mi hanno sempre incoraggiato.

Grazie al mio fidanzato Jonathan che mi è stato vicino in ogni momento.

Infine ringrazio tutti gli amici e parenti per avermi sostenuto in ogni modo, con un ruolo più o meno importante ma comunque fondamentale per me.