



UNIVERSITA' DEGLI STUDI DI TORINO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Anno Accademico 2024/2025

Tecnologie del Linguaggio Naturale-Parte I Prof. A.Mazzei

Autore: Annalisa Sabatelli Matr. 866879

“Dialog System: Lara is hiring!”

Documentazione di progetto

1. Introduzione

Il progetto consiste nella realizzazione di un Dialog Speech tra Lara Croft, celebre personaggio dei videogame e cinematografico, e un utente candidato per le selezioni del nuovo assistente in campo di Lara. In particolare, durante l'esecuzione del programma, all'utente vengono sottoposte tre quesiti relativi alla vita di Lara, alle sue avventure e alla cinematografia e gli vengono assegnati 10 punti per ogni risposta corretta. Il voto complessivo dell'interrogazione è la somma dei punti ottenuti per ciascuna domanda. Solo in caso di tre risposte corrette il candidato può diventare il nuovo assistente. Le risposte fornite dall'utente sono considerate corrette o sbagliate senza possibilità di replica e/o di un secondo tentativo. Solo se la risposta dell'utente è vuota, Lara lo invita a rispondere in ogni caso prima di andare avanti con il dialogo.

2. Architettura software del Dialog System

Il Dialog System è un applicativo desktop la cui architettura software è illustrata nella figura 1.



Figura 1: Architettura software Dialog System

L'applicazione è stata sviluppata in **Python**, utilizzando la sua libreria dedicata alla realizzazione di interfacce grafiche (GUI) per l'interazione con l'utente. Il codice è stato organizzato in moduli secondo i principi dell'**architettura a tre livelli**, al fine di garantire una maggiore manutenibilità e separazione delle responsabilità. Il livello di persistenza dei dati è stato implementato tramite il **database NoSQL MongoDB**. Il database contiene le informazioni relative al set di domande previste e mantiene traccia degli utenti e delle loro risposte. Poiché il database è eseguito all'interno di un container Docker, è necessario avviare il relativo container prima di eseguire l'applicazione con il comando `docker-compose up -d`.

3. Dialog Manager

3.1. Flusso logico

Al fine di contestualizzare adeguatamente le metodologie e gli strumenti adottati per la gestione delle fasi di analisi e generazione del dialogo, si ritiene opportuno introdurre preliminarmente lo schema generale del flusso logico che caratterizza l'interazione evidenziandone le possibili diramazioni evolutive.

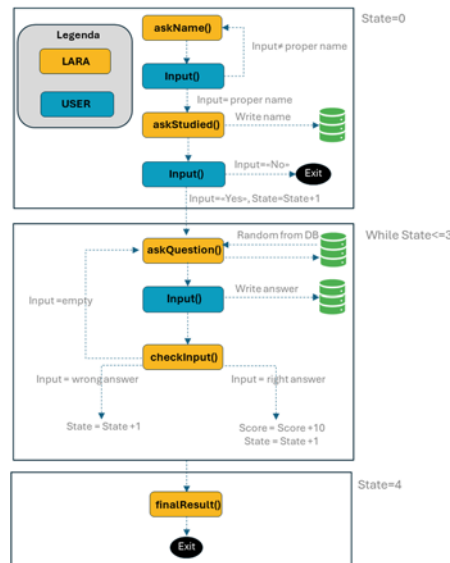


Figura 2: Flusso logico del dialogo

La figura 2 rappresenta il **flusso logico** del sistema di dialogo. L'interazione con l'utente è strutturata in diverse fasi. L'intero processo è suddiviso in tre fasi principali, a partire dallo stato iniziale (State=0) fino alla conclusione (State=4).

L'iniziativa è di LARA che all'avvio chiede all'utente di inserire il proprio nome. Se il nome inserito non è valido, la richiesta viene ripetuta fino a ottenere un input corretto. Una volta ricevuto un nome valido, questo viene memorizzato nel database e si passa alla successiva domanda preliminare, con cui LARA chiede all'utente se ha studiato. Se la risposta è negativa, Lara chiude la conversazione e il programma termina immediatamente. Se invece l'utente dichiara di aver studiato, lo stato del sistema viene incrementato e si entra nella fase successiva. In caso di risposta vuota la domanda viene ripetuta.

In questa ulteriore fase, che continua fino a quando lo stato è uguale a 3, il sistema seleziona casualmente una domanda dal database e la presenta all'utente. L'utente fornisce una risposta. Se la risposta è corretta, il punteggio dell'utente aumenta di dieci punti e lo stato viene incrementato. In caso di risposta sbagliata il punteggio non cambia ma lo stato avanza comunque. In caso di risposta vuota, Lara invita l'utente a tentare di rispondere fino ad ottenere una risposta diversa dalla stringa vuota. Al termine del ciclo di tre domande, quando lo stato raggiunge il valore 4, Lara comunica all'utente il risultato finale dell'interrogazione. Dopo questa fase conclusiva, Lara chiude la conversazione e il programma termina.

Il diagramma utilizza colori per distinguere le azioni del sistema LARA (in giallo) e gli input dell'utente (in azzurro) in modo da fornire una rappresentazione visiva chiara delle dinamiche di interazione tra utente e applicativo.

Dal punto di vista logico, la struttura del dialogo è stata modellata come una macchina a stati finiti. Gli stati da 1 a 3 rappresentano ciascuno una domanda posta all'utente. Lo stato 0 è uno stato di start in cui Lara Croft avvia il dialogo con alcune domande preliminari mentre lo stato 4 è lo stato finale in cui viene restituito all'utente il conteggio complessivo dei punti cumulati e l'esito della prova.

3.2.Frame-based, memory e backup strategy

Tutti i dati, i parametri e le informazioni utilizzate dal sistema di dialogo sono salvati in un database NoSQL. La scelta di una database documentale permette di fatto di memorizzare tali informazioni con una struttura riconducibile al formato JSON ma con prestazioni di consultazione e aggiornamento migliori rispetto ad un semplice file. Il database NoSQL realizzato con MongoDB è organizzato in due collection: “*Questions*” e “*Players*”.

La collection “*Questions*”, di cui la figura 3 ne riporta il contenuto, rappresenta la base di conoscenza del sistema di dialogo e contiene le domande con i rispettivi attributi.

La collection “*Players*” contiene un documento per ciascun utente, che funge da frame dinamico popolato durante l'interrogazione. Ogni documento presenta i seguenti campi:

- *_id*: identificativo univoco del documento generato da MongoDB;
- *name*: nome fornito dall'utente;

- *question_list_code*: lista degli identificativi numerici delle domande proposte all'utente;
- *question_answer*: risposte fornite dall'utente;
- *question_correct_answer*: risposta corretta attesa dal sistema;
- *player_total_points*: punteggio totale ottenuto.

La Figura 4 riporta la struttura della Collection “*Players*” mentre la Figura 5 ne evidenzia il contenuto nel dettaglio, che rispecchia la tipica struttura di un documento JSON.

_id	name	question_list_code	question_answer	question_correct_answer	player total points
6831dca868ed54d729049074	Bill	7,2,6	Question 1: Robert Croft, Question 2: She was born in 1968, Question 3: Tibet and England	Question 1 correct answer: Richard Croft, Question 2 correct answer: 1968, Question 3 correct answer: Greece, Tibet, England	20

Figura 3: MongoDB: Collection “*Players*”

```

1 {
2   _id: ObjectId('6831dca868ed54d729049074'),
3   name: 'Bill',
4   question_list_code: [
5     7,
6     2,
7     6
8   ],
9   question_answer: [
10    'Question 1: Robert Croft',
11    'Question 2: She was born in 1968',
12    'Question 3: Tibet and England'
13  ],
14  question_correct_answer: [
15    'Question 1 correct answer: Richard Croft',
16    'Question 2 correct answer: 1968',
17    'Question 3 correct answer: Greece, Tibet, England'
18  ],
19  'player total points': 20
20 }

```

Figura 4: MongoDB: dettaglio riga di un documento della collection “*Players*”

In sintesi, il database documentale svolge un ruolo centrale assolvendo a diversi compiti: costituisce la base di conoscenza, fornisce il frame dinamico per ogni sessione di dialogo, garantisce la persistenza dei dati e permette l’implementazione di eventuali strategie di backup.

4. Analisi

L’analisi delle risposte dell’utente è stata condotta principalmente tramite espressioni regolari, senza effettuare una vera e propria analisi sintattica delle dipendenze tra le parole. A supporto, è stata utilizzata la libreria SpaCy per il Part Of Speech Tagging (POS) e il Named Entity Recognition (NER), al fine di migliorare l’efficacia delle regole.

SpaCy è una libreria open-source per l’elaborazione del linguaggio naturale, dotata di modelli neurali per il riconoscimento delle parti del discorso, delle dipendenze sintattiche, delle entità e per la categorizzazione del testo. È stato adottato il modello più accurato disponibile, con l’obiettivo di massimizzare la qualità della parsificazione.

La seguente riga di codice permette di caricare un modello linguistico pre-addestrato.

```
spacy.load("download en_core_web_trf")
```

Un modello come en_core_web_trf (dove en = inglese, trf = transformer-based) è uno dei modelli più avanzati per l'elaborazione del linguaggio naturale in inglese.

Il database NoSQL è stato popolato con un set di 10 domande dal quale Lara attinge in modo random. Al fine di agevolare l'analisi delle risposte, ciascuna domanda è stata caratterizzata dai seguenti attributi:

- *_id*: identificativo univoco del documento generato da MongoDB;
- *code*: codice numerico intero progressivo assegnato in modo univoco a ciascuna domanda;
- *type*: categoria della domanda in base alla risposta attesa. Nel dettaglio: binary (vero/falso), year (data), number, list (elenco di oggetti), proper name (nome proprio);
- *keyWord*: elenco di nomi che ci si aspetta possano essere contenuti nella risposta.

La figura seguente riporta la struttura della Collection “Questions”.








_id	code	type	question	keyWord	param	answer
 683d7a5f68413c1be6519483	2	year	When was Lara Croft born?	year,born,Lara Croft,Lara	Lara Croft,born	1968
 683d7a5f68413c1be6519484	1	binary	Is Lara Croft an archaeologist?	archaeologist,Lara Croft,Lara,yes	Lara Croft,be,an archaeologist	true
 683d7a5f68413c1be6519485	5	number	How many films are there about Lara?	film,Lara Croft,Lara	film,Lara	3
 683d7a5f68413c1be6519486	4	year	When did the first videogame of Lara Croft releas...	year,videogame,Lara Croft,Lara,first	the first video game,Lara Croft,release	1996
 683d7a5f68413c1be6519487	6	list	What are two settings of the movie Tomb Raider: T...	places,settings	two,settings,the movie Tomb Raider: The Cradle of life	Greece,Tibet,England
 683d7a5f68413c1be6519488	7	proper name	What is Lara Croft's father's name?	father,name,Lara Croft,Lara	Lara Croft's father	Richard Croft
 683d7a5f68413c1be6519489	3	list	What are two weapons of Lara Croft?	weapons,arms	two,weapon,Lara Croft	pistols,shotgun,submachine gun,rifle,crossbow,grenade,M16 rifle,arrows,pickaxe,knife,revolver,explosives,silenced SMG,compound bow,assault rifle

Figura 5:MongoDB:Collection "Questions"

La categorizzazione delle domande e l'utilizzo delle key word sono di ausilio all'analisi delle risposte. Per esempio, si riporta il codice che effettua l'analisi delle risposte di tipo “number”.

```
def mng_question_number(user_message, correct_answer):  
    if (correct_answer in user_message.lower() or num2words(int(correct_answer), lang='en') in  
user_message.lower()):  
        answer=u.check_answer_no_list(user_message, correct_answer, key_word, type)  
    else:  
        answer=False  
    return answer
```

Per prima cosa si verifica se la risposta attesa è contenuta nell'input dell'utente (*user_message*). Tuttavia, questa condizione non basta a garantire la correttezza semantica della risposta. Ad esempio, alla domanda *"How many films are there about Lara?"*, la risposta attesa è 3, ma un input come *"I cani di Lara sono 3"* pur contenendo il numero corretto, risulterebbe semanticamente errato. Per questo, si applica un controllo aggiuntivo, implementato nel metodo seguente, per escludere risposte incoerenti con il contesto.

```
def check_answer_no_list(user_message, correct_answer, key_word, type):
    ordinal_stopwords = {"first", "second", "third", "fourth", "fifth", "last", "next",
"previous"}
    if type=="binary":
        correct_answer=str(correct_answer)
        user_message = user_message.replace(correct_answer, "")
        doc_user_message = nlp(user_message.lower())
        word_user = set()
        for token in doc_user_message:
            if token.pos_ in {"NOUN"} and token.lemma_.lower() not in ordinal_stopwords and
token.text not in ordinal_stopwords:
                word_user.add(token.lemma_.lower())
        if type!="proper name":
            proper_names = parser_proper_name(user_message)
            for name in proper_names:
                word_user.add(name)
        word_keyword = set(key_word)
        if not word_user or word_user.issubset(word_keyword):
            return True
        return False
```

L'input dell'utente viene analizzato per estrarre il set di nomi (sostantivi) presenti, escludendo eventuali stop word e i sostantivi già contenuti nella risposta corretta. Se l'insieme così ottenuto è un sottoinsieme dell'insieme delle key word attese, si inferisce che la risposta non contiene elementi estranei al contesto e quindi può essere ritenuta corretta.

5. Generazione

Per la generazione del Text-Plan e del Sentence-Plan è stata impiegata la libreria SimpleNLG, in lingua inglese. La varietà di tipologie di domande ammissibili ha rappresentato un fattore di complessità significativo, poiché ha reso molto eterogenea la struttura delle frasi interrogative da generare. Per evitare una proliferazione di metodi poco generalizzabili, ciascuno troppo focalizzato sulla struttura di una domanda specifica, si è pensato di elaborare un metodo generativo, basato sulla libreria Simple-NLG, che potesse generalizzare bene per categoria di domanda. A titolo esemplificativo, si riportano i metodi per la

generazione di domande di tipo “number” e di tipo “proper name”. Per ciascuna domanda sono stati indicati a database i parametri per generarla da fornire in input ai rispettivi metodi.

5.1.1. *generate_how_many_question*

Il metodo *generate_how_many_question(subject, entity)* genera frasi interrogative che richiedono il numero di occorrenze di un determinato evento o entità. Il metodo prende in input due parametri: “*subject*” è il soggetto principale della domanda mentre “*entity*” è l’entità di cui si vuole sapere la quantità. Ad esempio: “*How many films are there about Lara?*”. Si elencano di seguito i passi per creare la frase:

- si crea una NounPhrase per il soggetto utilizzando il parametro “*subject*” con il setting plurale;
- si crea una Clause che ha come soggetto il la NounPhrase e il verbo “*be*” come predicato;
- si imposta la forma interrogativa con il metodo *setFeature()* con i seguenti parametri:
 - *simplenlg.Feature.INTERROGATIVE_TYPE*;
 - *simplenlg.InterrogativeType.HOW_MANY*.
- si crea una seconda NounPhrase per il complemento oggetto utilizzando il parametro “*entity*” ed eventuali modificatori.

5.1.2. *generate_name_question*

Il metodo *generate_name_question(entity)* genera frasi interrogative che richiedono il nome proprio di una persona o di una entità. Il metodo prende in input il parametro “*entity*” che è l’entità di cui si chiede il nome proprio. Ad esempio: “*What is Lara Croft's father's name?*”. Anche in questo caso, si elencano di seguito i passi per creare la frase:

- si crea una NounPhrase per il soggetto utilizzando il parametro “*entity*”;
- si crea una Clause che ha come soggetto il la NounPhrase e il verbo “*be*” come predicato;
- si imposta la forma interrogativa con il metodo *setFeature()* con i seguenti parametri:
 - *simplenlg.Feature.INTERROGATIVE_TYPE*;
 - *simplenlg.InterrogativeType.WHAT_OBJECT*.

6. Valutazione del Dialog System

Si procede con l’analisi di alcuni dialoghi o di loro frammenti. All’inizio di ogni interrogazione viene richiesto all’utente il proprio nome e il dialogo non procede nello step successivo se questo non viene comunicato e riconosciuto correttamente dal sistema.

```
Lara: Hi! I am Lara Croft.  
Lara: What is your name?  
Player: Paul  
Lara: Hi! Paul. Let's start! Did you study?
```

La prima domanda mostra già alcuni limiti del sistema. Analizzando diverse risposte alla domanda “What’s your name?”, è emerso che il sistema non riconosce i nomi propri in italiano e anche alcuni nomi anglosassoni. Ad esempio, anche se la risposta “Carol” è formalmente corretta, il nome non viene identificato come entità di tipo Person dal metodo che implementa il NER Tagging, inducendo il sistema a richiedere nuovamente l’inserimento del nome.

```
Lara: Hi! I am Lara Croft.  
Lara: What is your name?  
Player: Carol  
Lara: You must tell me your name.
```

Alla domanda immediatamente successiva l’utente può rispondere in modo affermativo con locuzioni tipo “yes”, “of course”, “I have studied” o in modo negativo determinando, però, in questo secondo caso l’uscita dal gioco. Una risposta vuota, invece, causa una ripetizione della medesima domanda. Ad esempio:

```
Lara: Hi! I am Lara Croft.  
Lara: What is your name?  
Player:  
Lara: You must tell me your name.  
Player: John  
Lara: Hi! John. Let's start! Did you study?  
Player:  
Lara: Please, let me know if you have studied. Don't make me  
waste time!
```

Dato che l’analisi della risposta è basata fondamentalmente su espressioni regolari e sulla presenza di un insieme limitato di key word, locuzioni affermative diverse da quelle previste nel codice sarebbero interpretate come risposte negative.

Anche nell’interrogazione vera e propria, in presenza di risposte vuote da parte dell’utente, il sistema insiste finchè non ottiene una stringa di qualsiasi tipo.

```
Lara: Hi! I am Lara Croft.  
Lara: What is your name?  
Player:  
Lara: You must tell me your name.  
Player: John  
Lara: Hi! John. Let's start! Did you study?  
Player:  
Lara: Please, let me know if you have studied. Don't make me  
waste time!  
Player: yes  
Lara: Well, we'll start then! I will ask for 3 questions  
about my character. The first question is: When was lara  
croft born?  
Player:  
Lara: Please give me an answer right or wrong.Try!
```

L’uso combinato di espressioni regolari e key word verificate parsificando la frase rende il sistema abbastanza robusto rispetto a risposte semanticamente sbagliate. Si consideri l’esempio seguente:

```
Lara: Well, we'll start then! I will ask for 3 questions  
about my character. The first question is: When was lara  
croft born?  
Player:  
Lara: Please give me an answer right or wrong.Try!  
Player: Peter was born in 1968  
Lara: Bad response! Sorry, no points gained. Go on with the  
second question. What are two settings of the movie Tomb  
Raider: The Cradle of life?
```

Sebbene la risposta corretta sia effettivamente “1968”, la presenza di una parola fuori contesto consente al sistema di rilevare che la risposta, nel suo complesso, è errata. D’altra parte in questo risiede anche un’altra limitazione del sistema: sinonimi delle key word non sono riconosciuti come corretti.


```
Lara: Bad response! Sorry, no points gained. Go on with the
second question. What are two settings of the movie Tomb
Raider: The Cradle of life?
Player: Two locations are England and Tibet
Lara: Bad response! Sorry, no points gained. Go on with the
third question. What is Lara Croft's father's name?
```

Per esempio, in quest'ultimo caso le ambientazioni indicate sono effettivamente corrette ma la risposta non viene riconosciuta come tale perché *"locations"* pur essendo un sinonimo di *"places"* o *"settings"* non figura nelle key word predefinite, falsando così l'esito della valutazione. Per superare questo limite, una possibile soluzione sarebbe quella di arricchire le keyword attraverso l'uso di risorse lessicali come WordNet, in modo da includere un insieme più ampio e rappresentativo di sinonimi e varianti linguistiche, migliorando la capacità del sistema di riconoscere risposte semanticamente corrette.

7. Trindi Ticklist

Di seguito sono riportate le domande della Trindi Ticklist e le relative risposte con riferimento al Dialog System presentato in questo lavoro.

- **Q1: L'interpretazione degli enunciati è sensibile al contesto?**

I risultati vengono interpretati nel dominio di utilizzo, per cui il sistema lavora solo nel contesto dell'interrogazione.

- **Q2: Il sistema è in grado di gestire le risposte alle domande che forniscono più informazioni di quelle richieste?**

Sì, è in grado di verificare la correttezza della risposta data anche se questa è corredata dall'utente da ulteriori informazioni a prescindere dalla loro pertinenza.

- **Q3: Il sistema è in grado di gestire le risposte alle domande che danno informazioni diverse da quelle effettivamente richieste?**

Sì, è in grado di gestire risposte fuori contesto o sbagliate, rispondendo con frasi che indicano all'utente di aver sbagliato.

- **Q4: Il sistema è in grado di gestire le risposte alle domande che forniscono meno informazioni di quelle effettivamente richieste?**

No, non è previsto che l'utente possa scrivere parzialmente una risposta, pertanto non è attualmente presente un sistema di riconoscimento parziale dell'input. Le risposte parziali sono considerate non corrette.

- **Q5: Il sistema è in grado di gestire informazioni specificate negativamente?**

No, essendo una interrogazione, è stato considerato inconsueto che l'utente risponda con una frase negata. L'analisi delle risposte è stata impostata solo per frasi affermative. Per esempio, la risposta ad una domanda di tipo vero o falso non viene riconosciuta correttamente se espressa come negazione del tipo "Non è vero" o "Non è falso".

- **Q6: Il sistema è in grado di gestire una domanda senza risposta?**

Sì, il sistema gestisce una risposta "vuota" invitando l'utente a esprimersi in qualche modo sulla domanda che gli è stata posta.

- **Q7: Il sistema è in grado di gestire informazioni incoerenti?**

Sì, rispondendo come indicato nella domanda Q3.

- **Q8: E' possibile ottenere un tutorial sul sistema per quanto riguarda il tipo di informazioni che il sistema può fornire e quali sono i vincoli di input?**

Durante l'esecuzione del sistema no, ma nella relazione sono indicati tutti i dettagli ed eventuali vincoli di risposta che l'utente dovrebbe rispettare.

- **Q9: Il sistema pone solo domande di follow-up appropriate?**

Si, il sistema fa delle domande in base alle risposte dell'utente, per cui segue il flusso di input dall'utente.

- **Q10: Il sistema è in grado di gestire input rumorosi?**

No, il sistema comprende soltanto frasi di senso compiuto.

8. Conclusioni

In conclusione, il progetto ha evidenziato la complessità dell'analisi e comprensione del linguaggio naturale. È emersa in particolare la difficoltà nel gestire le numerose flessioni linguistiche e le molteplici combinazioni sintattiche che una frase può assumere. Ogni metodo, implementato per tenere conto di una specifica regolarità, si rivelava spesso insufficiente a gestire altre varianti linguistiche. La necessità di coprire casi sempre più vari ha portato alla costruzione di metodi complessi, basati su numerose regole esplicite, rendendo evidente quanto sia difficile gestire in modo esaustivo la varietà del linguaggio con approcci deterministici.