

Università Politecnica delle Marche  
Dipartimento di Ingegneria dell'Informazione  
Facoltà di Ingegneria Informatica e dell'Automazione



**Implementazione di un chatbot di assistenza virtuale per la  
libreria Fàgola**

**Docenti:**  
Domenico Ursino  
Michele Marchetti

**Studenti:**  
Silvia Ciuffreda  
Luca Liberatore  
Beatrice Moliterno

**ANNO ACCADEMICO 2022/2023**

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Chatbot ed NLP . . . . .	3
1.2	L'idea del progetto . . . . .	3
<b>2</b>	<b>RASA</b>	<b>4</b>
2.1	RASA NLU . . . . .	5
2.2	RASA CORE . . . . .	8
<b>3</b>	<b>Chatbot Fàgola</b>	<b>11</b>
3.1	Dataset . . . . .	12
<b>4</b>	<b>Tipologie di conversazione</b>	<b>13</b>
4.1	Inizio conversazione . . . . .	13
4.2	Informazioni generali riguardo la libreria Fàgola . . . . .	13
4.3	Ricerca sui libri . . . . .	14
4.3.1	Informazioni su uno specifico libro . . . . .	15
4.3.2	Ricerca per autore . . . . .	16
4.3.3	Ricerca per genere . . . . .	17
4.3.4	Ricerca per età . . . . .	17
4.4	Prenotazione libro . . . . .	18
<b>5</b>	<b>Connessione a Telegram</b>	<b>21</b>
5.1	Testing . . . . .	21

## Elenco delle figure

1	RASA . . . . .	4
2	RASA - Installazione . . . . .	4
3	RASA - Struttura dei file . . . . .	5
4	Architettura RASA . . . . .	6
5	Pipeline RASA . . . . .	6
6	Pipeline RASA - Tokenizer . . . . .	6
7	Pipeline RASA - Featurizer . . . . .	7
8	Pipeline RASA - DIET Classifier . . . . .	7
9	Pipeline RASA - Policy . . . . .	8
10	RASA - Form . . . . .	9
11	RASA - Action . . . . .	9
12	Libreria . . . . .	11
13	Tipologia inizio conversazione . . . . .	13
14	Richiesta orario di apertura . . . . .	13
15	Curiosità riguardo la libreria Fàgola . . . . .	14
16	Posizione fisica . . . . .	14
17	Servizi offerti da Fàgola . . . . .	14
18	calendario settimanale eventi . . . . .	14
19	Eventi in programma . . . . .	14
20	Intent ed esempi per la richiesta di informazioni su un libro data la conoscenza del titolo . . . . .	15
21	Output nel caso in cui l'utente richiede informazioni su un libro per cui vengono trovate corrispondenze multiple . . . . .	15
22	Esempio di conversazione continuativa per la richiesta di informazioni su un libro specifico . . . . .	16
23	Ricerca di un libro tramite un autore definito . . . . .	16
24	Ricerca di un libro tramite un autore definito . . . . .	16
25	Intent ed esempi per la ricerca basandosi sul criterio del genere letterario . . . . .	17
26	Ricerca di un libro tramite il genere letterario desiderato . . . . .	17
27	Ricerca di un libro personalizzata per la fascia d'età 6 anni . . . . .	18
28	Intent ed esempi per la richiesta di prenotazione di un libro . . . . .	18
29	Definizione form per la prenotazione di un libro . . . . .	18
30	Slot per form prenotazione di un libro . . . . .	19
31	Regole per form sulla prenotazione di un libro . . . . .	20
32	Configurazione bot telegram . . . . .	21
33	Richiesta informazioni su un libro specifico ed esempio di prenotazione di un libro pt.1 . . . . .	21
34	Richiesta informazioni su un libro specifico ed esempio di prenotazione di un libro pt.2 . . . . .	22
35	Richiesta informazioni su un libro specifico ed esempio di prenotazione di un libro pt.2 . . . . .	22
36	Esempio di interruzione di una form pt.1 . . . . .	23
37	Esempio di interruzione di una form pt.2 . . . . .	23
38	Esempio di interruzione di una form pt.3 . . . . .	24
39	libro non disponibile per la prenotazione e conseguente disattivazione della form pt.1 . . . . .	24
40	libro non disponibile per la prenotazione e conseguente disattivazione della form pt.2 . . . . .	25

# 1 Introduzione

## 1.1 Chatbot ed NLP

Nell'era digitale in cui viviamo, l'interazione tra esseri umani e macchine sta diventando sempre più comune e integrata nella nostra quotidianità. Una delle manifestazioni più interessanti di questa interazione è rappresentata dai chatbot, i quali sono presenti in diversi campi della nostra società; tra le applicazioni più comunemente note si citano: Servizio clienti, E-commerce, Settore bancario e finanziario, Settore sanitario, Educazione, ecc. Le motivazioni per scegliere un chatbot sono numerose, tra le più importanti si sottolinea la possibilità di garantire un servizio attiva 24 ore su 24, la riduzione dei costi: si pensi ad esempio il costo di mantenere un intero centro di assistenza clienti composto da persone, rispetto al costo di mantenimento di un singolo chatbot che può gestire tutte le richieste contemporaneamente.

I chatbot possono essere inquadrati come un sottoinsieme dell'Intelligenza Artificiale (IA), più precisamente si vanno a collocare in quella parte di IA che riguarda il Natural Language Processing (NLP). L'NLP è una branchia dell'informatica che si occupa di permettere ai computer di capire ed elaborare il linguaggio umano. L'NLP, contrariamente a quanto si possa pensare, non è una disciplina nuova, infatti, ha i suoi albori alla fine degli anni '40, dove si fecero i primi esperimenti di traduzione automatica di vari linguaggi, in quella che è nota come la prima delle quattro fasi della sua evoluzione. Lo sviluppo dell'NLP ha riscontrato non poche problematiche a causa dell'ambiguità del linguaggio umano, definita come la capacità di essere compresi in più di un modo. L'NLP si sviluppa in diverse fasi: elaborazione morfologica, in cui si fa una *tokenizzazione* delle parole; analisi sintattica, si cercano le relazioni sintattiche tra le diverse parole; analisi semantica, dove si cerca di estrapolare il significato esatto di ciò che viene detto; analisi pragmatica, in cui si collegano determinati oggetti di una frase con riferimento ad altri oggetti riscontrati nelle frasi precedenti. La proposta che è alla base dei chatbot è costituita da un sistema informatico che capisca il linguaggio naturale ed abbia la capacità di tradurre le frasi scritte da un essere umano in una rappresentazione interna, con la finalità di generare delle risposte associate a tali domande. Le sfide che sono dietro questo procedimento includono un'analisi sintattica e semantica delle domande, parallelamente al gap lessicale, l'ambiguità ed il multilinguismo. I chatbot possono essere sia di tipo semplice, cioè che rispondono ad una singola query, oppure più sofisticati, cioè in grado di sostenere conversazioni più complesse e di apprendere.

## 1.2 L'idea del progetto

L'idea che sta dietro il progetto è quella di implementare un chatbot di assistenza per una libreria, in particolare si fa riferimento alla libreria Fàgola nel centro di Ancona. I servizi messi a disposizione sono, innanzitutto, le classiche informazioni generali sulla libreria, quali eventi associati, storia della libreria, posizione ed orari di apertura. In aggiunta, sono implementate le classiche funzionalità base di conversazione con un chatbot. Le specifiche avanzate prevedono la ricerca dei libri a disposizione della libreria utilizzando diversi criteri di selezione: per autore, per genere, per età e per titolo del libro. Inoltre, è possibile anche effettuare delle prenotazioni di un libro, verificando la sua disponibilità in base ad una variabile booleana, aggiunta come colonna al file csv utilizzato come database della libreria. Si possono richiedere informazioni su uno specifico libro ed avviare conversazioni con il chatbot che, ricordandosi del titolo del libro di cui si sta parlando, è in grado di rispondere ad ulteriori domande come l'età a cui è consigliato leggerlo, il costo, lo scaffale ed il reparto in cui è stipato. La volontà è quella di realizzare un chatbot più vicino possibile ad un comportamento umano di un addetto alla libreria, che è quindi in grado di supportare conversazioni anche articolate.

## 2 RASA

La scelta del software da utilizzare è ricaduta sul framework RASA, dato che è una risorsa gratis ed open source; inoltre, è molto flessibile, personalizzabile e garantisce la privacy dei dati, perché permette il deploy dell'applicativo sul proprio server in modo tale che tutti i dati restino in locale. Rasa utilizza il NLP ed il machine learning (ML) per capire cosa è stato detto e quindi generare una risposta. La versione utilizzata per svolgere il progetto è la 3.X. Le principali caratteristiche di RASA includono:

- Elaborazione del linguaggio naturale (NLU)
- Gestione conversazioni con gli utenti e tenere traccia delle risposte fornite
- Capacità di apprendimento e addestramento che migliorano nel tempo
- Grande capacità di personalizzazione del framework
- Integrazione con piattaforme di messaggistica (es: Telegram)
- Community attiva in grande espansione



Figura 1: RASA

Per essere più specifici, l'architettura RASA si articola in due componenti principali: RASA NLU e RASA CORE. Il primo si occupa di capire e cosa ha detto l'utente tramite input, il secondo, che dipende dal primo, risponde all'input dell'utente presente, tenendo conto di quello che è stato detto nel passato, eseguendo delle azioni.

Per installare RASA si utilizza Python tramite il comando riportato in Fig2.

```
python -m pip install rasa
```

Figura 2: RASA - Installazione

Prima di addentrarsi nella descrizione dettagliata delle singole componenti di RASA, viene riportata una panoramica dei vari file di default che sono a disposizione a seguito di una fresca installazione.

Di seguito si fa un breve elenco dei file (Fig3):

- "domain.yml" contiene le informazioni sul dominio del chabot, come gli intenti, le entità, le azioni e le risposte possibili.
- "config.yml" possiede la configurazione per i modelli di machine learning utilizzati.

## 2.1 RASA NLU

- "data" è una cartella che può includere file di addestramento per l'intento e l'estrazione delle entità, nonché file di conversazioni per addestrare il modello di dialogo.
- "nlu.yml" contiene esempi per gli intenti e le entità.
- "stories.yml" include esempi di conversazioni multiple con il chatbot.
- "rules.yml" ha le regole predefinite per le politiche di dialogo.

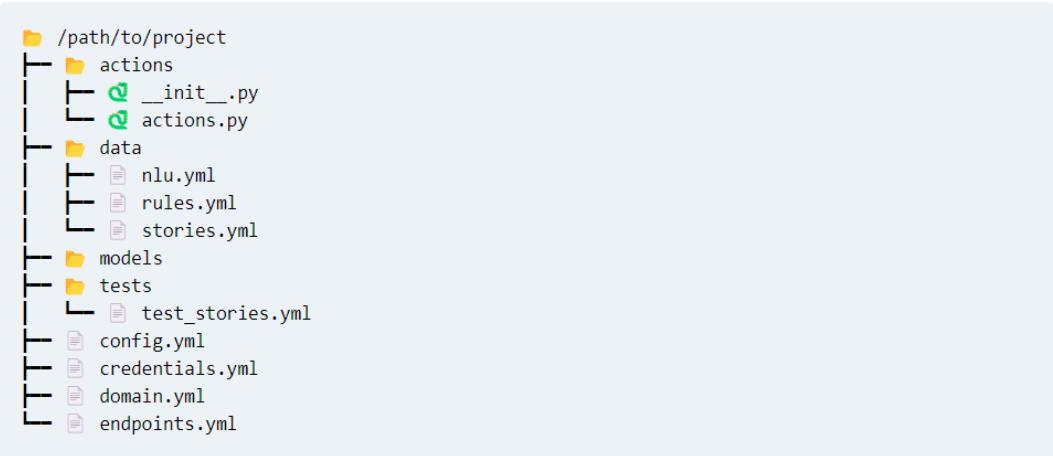


Figura 3: RASA - Struttura dei file

### 2.1 RASA NLU

La sezione di RASA NLU, come già detto, non predice le risposte ma cerca solamente di capire cosa ha detto l'utente che sta interagendo con il chatbot. Nel gergo del chatbot, le richieste dell'utente possono essere riassunte in singoli gruppi di intenti, *intent* per l'appunto: ad ogni *intent* sono associate più formulazione di frasi che indicano lo stesso concetto. Come già anticipato all'inizio della tesi le richieste da parte dell'utente spaziano da quelle più semplici (orario biblioteca, informazioni sulla biblioteca) a quelle più complesse (ricerca e prenotazione di libri); un esempio di *intent* è riportato nei paragrafi successivi in Figura 20. Tutti questi intenti appartengono al file "**nlu.yml**" all'interno dell'architettura di RASA. Essendo RASA un'architettura ibrida, composta da una componente dichiarativa (intenti) ed una di ML (apprendimento supervisionato), per essere addestrato correttamente, il modello, fa riferimento ad una serie di frasi esempio: data una richiesta in input il classificatore assocerà tale frase all'*intent* con punteggio più alto. E' possibile osservare il punteggio che il classificatore assegna ad ogni *intent* tramite il comando "rasa shell nlu"; per allenare il modello invece si usa il comando "rasa train".

Al fianco degli intenti, vengono definite anche le entità (*entity*). Le entità sono degli "oggetti" che permettono di conoscere meglio l'espressione associata alla frase; in altri termini, le entità rappresentano un oggetto all'interno di una classe di oggetti, in modo tale che l'NLU capisca che si sta facendo riferimento ad una particolare istanza di quella classe. Ad esempio, se l'intento è quello di voler ordinare una pizza, l'entità sarà il nome della pizza stessa che, per ovvie ragioni, può variare su una vasta gamma di tipologie. Strettamente correlate con il significato di entità sono le *Lookup table*. Queste ultime rappresentano essenzialmente i valori che possono essere assunti dalle entità: sono degli elenchi di oggetti appartenenti alla stessa classe. Per ritornare all'esempio precedente, una *Lookup table* che faccia riferimento ad una pizza potrà elencare i vari tipi di pizza, cioè i singoli valori di entità (margherita, funghi, ecc.). Nel nostro caso le tabelle di lookup utilizzate fanno riferimento all'autore del libro, al titolo del libro, alla fascia di età, al genere e così via. Queste tre caratteristiche appena elencate, intenti entità e *Lookup table*, risiedono tutte e tre all'interno del file "nlu.yml".

## 2.1 RASA NLU

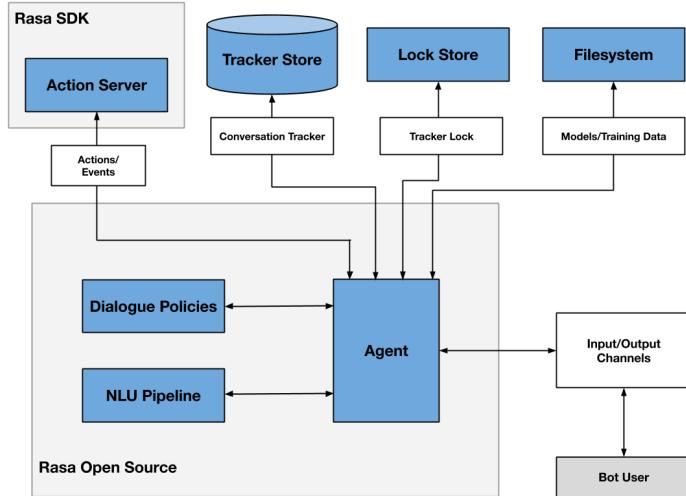


Figura 4: Architettura RASA

Rasa, al fine di classificare gli *intent*, segue una sequenza di passaggi predefinita ben precisa: la pipeline. La pipeline viene definita all'interno del file "**config.yml**", insieme ad altre componenti, quali le policy e il linguaggio da utilizzare per il chatbot: italiano, nel nostro caso. In altri termini, la pipeline si occupa della previsione dell'intento e dell'estrazione delle entità; le policy si occupano di selezionare l'azione successiva. Nella pipeline, dunque, si vanno a definire i passaggi tramite i quali si processerà ogni frase di input. E' bene precisare che l'ordine di esecuzione è di fondamentale importanza, perché ogni frase in input passa sequenzialmente dentro ogni componente specificata nella pipeline. Ci sono diversi tipi di componenti presenti in una pipeline (Fig 5) di Rasa, tra i principali si possono elencare i seguenti:

- Tokenizers
- Featurizers
- Intent Classifiers
- Entity Extractors

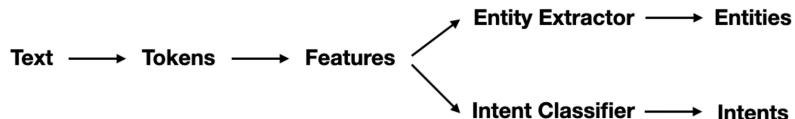


Figura 5: Pipeline RASA

Il primo passo in una pipeline di Rasa è suddividere un'unità di discorso in parti più piccole di testo, chiamate token. Questo procedimento è il primo effettuato nella pipeline. Un esempio è raffigurato in Fig6.

**"Hi, my name is Vincent."** → **[ "Hi", "my", "name", "is", "Vincent" ]**

Figura 6: Pipeline RASA - Tokenizer

## 2.1 RASA NLU

Una volta ottenuti i token, si possono aggiungere caratteristiche numeriche per l'apprendimento automatico. Si passa dunque al secondo componente: il Featurizer. In Fig7 vi è un esempio rispetto al vocabolo inglese "hi". In RASA vi sono due tipi di Featurizer: *Sparse Features* e *Dense Features*. Nel nostro caso ci si è basati sui secondi; questi ultimi sono dei modelli pre-addestrati, che sono rappresentati o dallo SpacyFeaturizers o dallo LanguageModelFeaturizers (da Hugging Face). Nel nostro chatbot della libreria si è scelto di utilizzare lo Spacy; tuttavia affinché tutto funzioni correttamente è necessario anche associarci un tokenizzatore appropriato: lo SpacyTokenizer. Ovviamente, lo Spacy deve prima essere installando nell'ambiente virtuale di RASA ed associato ad un modello pre-addestrato di parole: si è scelto il pacchetto "it\_core\_news\_md".



Figura 7: Pipeline RASA - Featurizer

Il passo successivo, come da schema, riguarda la classificazione degli intenti. Quindi, successivamente alla generazione delle features per ogni token e per le frasi intere, si da tutto in pasto al DIETClassifier, una rete neurale, che è in grado sia di fare classificazione degli *intent* che estrazione delle entità. Il procedimento è illustrato in Fig8.

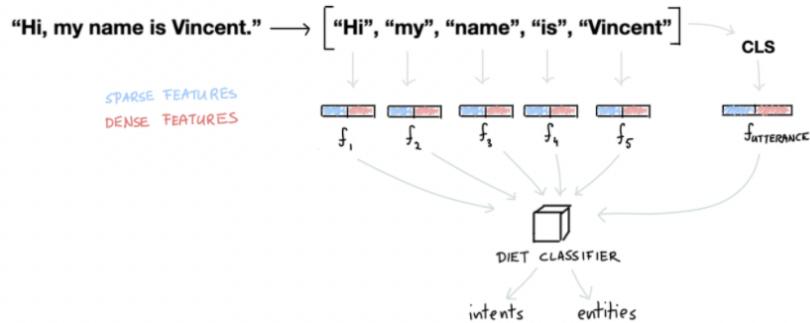


Figura 8: Pipeline RASA - DIET Classifier

L'ultima componente da analizzare all'interno del file "config.yml", come già anticipato, è costituita dalle policy. RASA si basa su tale concetto fondamentale: quando sta per predire un'azione, non guarda solo alle attuali entità e intenti, ma ha un quadro generale dell'intera conversazione fino a quel punto. Il file di configurazione predefinito di Rasa contiene tre policy principali:

- RulePolicy: Gestisce le conversazioni che corrispondono a pattern di regole predefinite. Effettua previsioni basate sulle regole definite nel file rules.yml.
- MemoizationPolicy: Verifica se la conversazione corrente corrisponde a una delle storie presenti nei dati di addestramento. In tal caso, prevede la prossima azione basandosi sulle storie corrispondenti.
- TEDPolicy: Utilizza l'apprendimento automatico per prevedere la prossima migliore azione.

Queste policy effettuano previsioni sulla prossima azione in parallelo e la policy con la maggiore confidenza determina la prossima azione da intraprendere.

## 2.2 RASA CORE

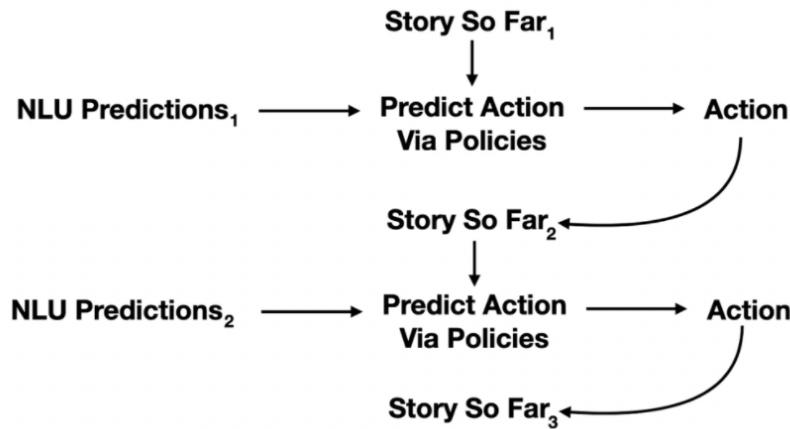


Figura 9: Pipeline RASA - Policy

## 2.2 RASA CORE

Rasa Core è un componente chiave del framework Rasa per lo sviluppo di chatbot. Si concentra sulla gestione delle conversazioni e sulla logica decisionale degli agenti di chatbot. In altre parole, RASA CORE è quella componente che si occupa di predire cosa deve fare il chatbot dopo avere ricevuto un input dall'utente. Esso, basandosi sui pattern conversazionali forniti in fase di addestramento, crea un modello di Machine Learning da questi in modo da imparare e poter rispondere in modo adeguato alle richieste in input. Inoltre, è possibile addestrare un modello di apprendimento automatico utilizzando tecniche come il reinforcement learning per far imparare al bot a prendere decisioni più complesse basate sul contesto della conversazione.

Nel file "**domain.yml**" sono contenute tutte le informazioni che il chatbot ha, più precisamente si elencano le seguenti:

- Responses
- Intents
- Slots
- Entities
- Forms and actions

Di questo elenco, fin'ora, siamo a conoscenza solamente del funzionamento di due di esse: gli intenti e le entità. Ora andiamo ad effettuare una breve descrizione delle altre componenti.

Le "Responses" sono delle risposte semplici e "statiche" che il bot deve dare a seguito di un particolare *intent* in input. Ad esempio, ad un intento che rientra nella categoria dei saluti, il bot dovrà a sua volta rispondere con una frase di saluto, una response per l'appuntamento. Questa tipologia di risposta non è vincolata ad essere unicamente del semplice testo, ma può anche essere un'immagine oppure dei pulsanti digitali da premere.

Gli "Slots" rappresentano la memoria a lungo termine in una conversazione, cioè quelle informazioni che il bot deve memorizzare in modo tale da poterle riutilizzare in futuro. Solitamente vengono passate le entità come slot. Nel nostro caso gli slot verranno utilizzati per passare informazioni importanti sui libri alle "Actions", come Titolo, Autore, Genere, ecc.

Le ultime due componenti che rimangono da analizzare sono le "Forms" e le "Actions". Partiamo dalle prime. Le for servono per raccogliere più informazioni correlate in un unico contenitore; si pensi al solito

## 2.2 RASA CORE

esempio della pizza, per ordinare una pizza, è necessario conoscere il tipo e la dimensione della pizza: le form si occupano di memorizzare tali informazioni. Dal punto di vista concettuale, una form può essere vista come un ciclo che continua a chiedere all'utente informazioni fino a quando non sono stati riempiti tutti gli slot necessari. Un esempio di flusso del genere è mostrato nella Fig10. Nel nostro caso le form verranno utilizzate per la prenotazione dei libri della biblioteca.

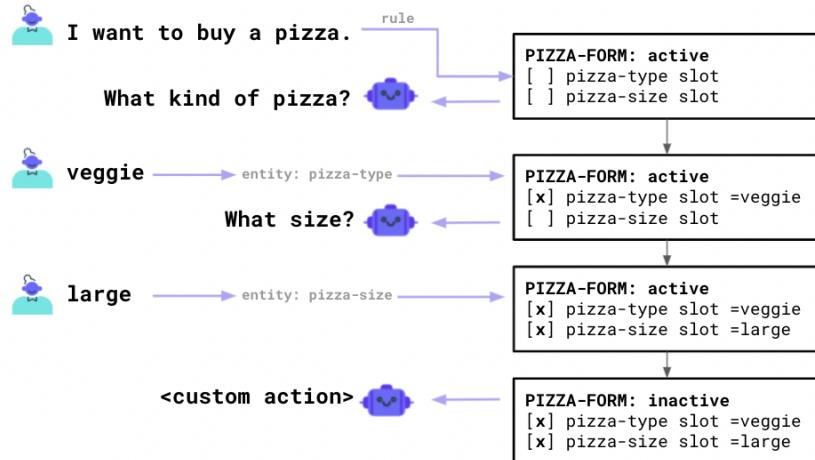


Figura 10: RASA - Form

Come si evince dalla rappresentazione, strettamente legato al concetto di form c'è quello delle azioni. Le action permettono di utilizzare del codice python per effettuare delle risposte più articolate rispetto a quelle semplici viste in precedenza, ad esempio si possono fare chiamate API o ricerche su DB o, come nel nostro caso, ricerche di informazioni sui libri su un file csv per mezzo della libreria Pandas. Le action sono contenute nel file *actions.py*. Ogni action è composta da due funzioni: la funzione "name" che ritorna il nome dell'azione e la funzione "run" che contiene il codice che deve eseguire l'azione. Le action vanno eseguite su un nuovo terminale per mezzo del comando: "rasa run actions". Uno schema del funzionamento delle action è in Fig11.

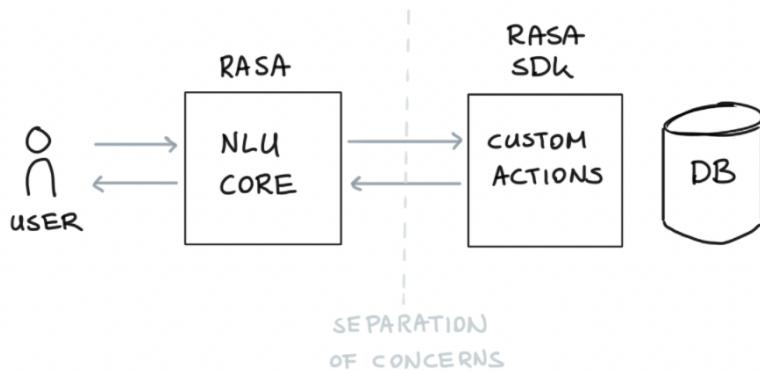


Figura 11: RASA - Action

Le ultime due componenti da analizzare sono le "rules" e le "stories".

Le rules sono delle regole predefinite e appartengono al file **rules.yml**. Queste regole associano particolari intent in input a delle azioni di risposta che deve dare il chatbot, in altre parole sono l'anello di collegamento tra l'input e l'output del chatbot. Sono utilizzate soprattutto per gestire comportamenti predefiniti dei chatbot.

Le stories, appartenenti al file **stories.yml**, descrivono invece flussi di conversazioni più articolati, ovvero delle frasi di botta e risposta tra utente e chatbot. Le stories servono per far allenare il chatbot in modo tale da saper rispondere alle varie evoluzioni differenti di conversazioni con l'utente. Le stories consentono

## 2.2 RASA CORE

di definire in modo più flessibile il flusso di conversazione del chatbot rispetto alle rules. Più le storie sono variegate e numerose e meglio si comporterà il chatbot.

### 3 Chatbot Fàgola

La libreria Fàgola è un ambiente storico per il centro di Ancona. La sua nascita risale al 1898, dove si affermò come punto di incontro per intellettuali e per gli appassionati del sapere del tempo. Dopo la morte del suo fondatore, a fine della Seconda Guerra Mondiale, la libreria fu ereditata dai discendenti, finché non venne rilevata da diversi proprietari e tutt'ora rappresenta un centro nevralgico per la cultura di ogni genere. L'integrazione di un chatbot con una libreria offre numerosi vantaggi che possono migliorare l'esperienza degli utenti e ottimizzare le operazioni all'interno di una libreria. Un chatbot permette un'assistenza immediata, una gestione semplificata delle operazioni e può promuovere eventi ed iniziative in maniera più efficace. L'idea che sta dietro il progetto è quella di poter contribuire al fiorente prospero che ha avuto ed ha la libreria, in modo tale da potersi continuare a confermare un punto di incontro per giovani e non, seguendo ormai le inevitabili onde della tecnologia che sbaraglia tutto ciò che non riesce a starle dietro.

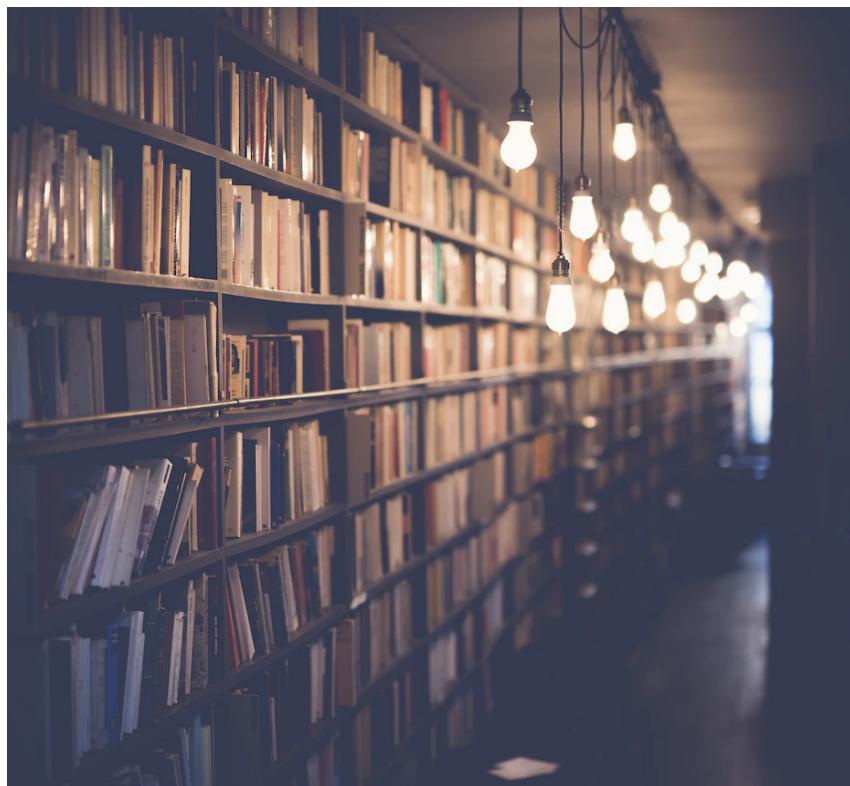


Figura 12: Libreria

### 3.1 Dataset

#### 3.1.1 Dataset

Il dataset è rappresentato da un file csv, il quale viene interrogato tramite le actions, con la finalità di rispondere alle richieste dei clienti. Nel file sono presenti diversi libri esempio, disposti per righe, mentre sulle colonne sono indicate le features corrispondenti. L'elenco degli attributi per ogni libro può essere espresso dal seguente elenco:

- Titolo
- Autore
- Genere
- Età
- Prezzo
- Descrizione
- Reparto
- Scaffale

## 4 Tipologie di conversazione

Nel contesto di questo progetto è essenziale comprendere le diverse tipologie di conversazione che il chatbot è in grado di sostenere al fine di fornire un'interazione più efficace e soddisfacente con gli utenti.

Nel corso di tale sezione, si esploderanno nel dettaglio le varie modalità in cui il chatbot può interagire con gli utenti, consentendo loro di ottenere risposte alle domande desiderate. Di fatto, analizzare le tipologie di conversazione sono fondamentali per sviluppare un chatbot che sia in grado di comprendere e rispondere alle esigenze e alle richieste degli utenti in modo efficace ed efficiente.

### 4.1 Inizio conversazione

Nell'ambito delle tipologie di conversazione, un elemento fondamentale è rappresentato dall'inizio della conversazione. Questo momento, infatti, offre l'opportunità di stabilire l'avvio dell'interazione tra l'utente e il chatbot nonché di presentare le funzionalità del chatbot stesso.

In particolare, l'inizio della conversazione può essere avviato dall'utente, la cui risposta del chatbot si traduce in un saluto che può assumere forme diverse, come "Ciao!", "Buongiorno!", o "Benvenuto nella chat della libreria!", seguito da una breve presentazione in cui illustra il motivo per il quale è stato programmato e in che cosa può essere d'aiuto per l'utente.

In Figura 13 è mostrata una tipologia di inizio conversazione.

```
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input -> ciao

Benvenuto nella chat della libreria Fàgola. Posso darti informazioni generali sulla libreria e chiarire quali sono i servizi che offriamo; puoi chiedermi info su un libro specifico oppure effettuare una ricerca per genere, autore, età . Inoltre, sono in grado effettuare una prenotazione di un libro da te desiderato. In cosa posso esserti d'aiuto?
```

Figura 13: Tipologia inizio conversazione

### 4.2 Informazioni generali riguardo la libreria Fàgola

Un aspetto essenziale nell'implementazione del chatbot per la libreria Fàgola riguarda la disponibilità di informazioni generali sulla libreria stessa. Tali informazioni comprendono:

- l'orario di apertura di Fàgola
- la storia di Fàgola
- posizione fisica di Fàgola
- servizi che Fàgola offre ai suoi clienti
- eventi che organizza Fàgola
- eventi in programma

Attraverso le risposte alle domande presenti nell'elenco, il chatbot può fornire agli utenti le informazioni di base necessarie per comprendere meglio la libreria e sfruttare i servizi offerti. Ciò contribuisce a creare un'interazione informativa e soddisfacente, consentendo agli utenti di accedere rapidamente alle informazioni di loro interesse.

Nelle Figure che seguono, per ciascuna richiesta di informazione riguardo la libreria Fàgola contenuta nell'elenco, sono mostrate le corrispondenti risposte da parte del bot.

```
Your input -> qual è l'orario di apertura?
La libreria Fàgola è aperta dal Lunedì al Venerdì dalle ore 9:00 alle 22:00.
```

Figura 14: Richiesta orario di apertura

#### 4.3 Ricerca sui libri

```
Your input -> dimmi qualcosa sulla libreria
Era il 1898 quando ad Ancona arrivò una famiglia di librai lunigiani e Giuseppe Fogola aprì inizialmente la libreria in corso Garibaldi.
Una libreria che diventò un luogo di incontro per i lettori e un centro culturale frequentato da intellettuali, insegnanti e professori.
Con la morte di Giuseppe nel 1945, le sue due nipotini Fiorella ed Ernestina restarono alla guida dal 1973 al 1990, quando la gestione venne rilevata dal libraio anconetano Paolo Fagnani.
Nel 2011 subentrò Giampaolo Fornasiero che però chiuse il 15 Giugno 2017, finché, a distanza di qualche mese, i dipendenti storici, Andrea Paternesi e Simona Rossi, decisamente di riaprire la libreria in una parte dei locali del vecchio punto vendita.
Da quell'anno Fogola sposta la sua entrata in corso Mazzini 170. Nel 2021 la libreria si è fatta ancora più grande, ampliando i suoi locali; i libri sono rimasti la sua ragione d'essere e il suo nome è mutato solo in parte, per accogliere Altro.
Da Fogola a Fàgola. Oggi Fàgola è un locale accogliente, curato nei minimi dettagli, dall'intenso profumo di cultura, dove la storia si mescola al moderno, le travi di legno a vista, i mattoni sulle colonne, le volte.
Uno spazio concepito per creare qualcosa di insolito per Ancona, con iniziative e incontri.
```

Figura 15: Curiosità riguardo la libreria Fàgola

```
Your input -> dove si trova la libreria?
La libreria Fàgola si trova nel centro città di Ancona esattamente in Piazza Camillo Benso di Cavour, 2.
```

Figura 16: Posizione fisica

```
Your input -> che servizi offrite?
La libreria Fàgola offre una vasta gamma di servizi per soddisfare le esigenze dei suoi visitatori e promuovere l'interesse per la lettura:
1. Spazi di lettura e studio: offre aree dedicate per la lettura e lo studio con tavoli, sedie, postazioni di lavoro e accesso a Internet.
Tali spazi consentono agli utenti di concentrarsi sulla lettura o lo studio in un ambiente tranquillo.
2. Eventi culturali: Fàgola spesso organizza eventi culturali come presentazioni di autori, dibattiti letterari, letture ad alta voce, laboratori di scrittura creativa, club del libro e workshop tematici.
Questi eventi promuovono l'interesse per la lettura e la partecipazione della comunità.
3. Prestito di libri: Il servizio principale offerto da una libreria è il prestito di libri. Gli utenti possono registrarsi come membri della libreria e prendere in prestito libri per un determinato periodo di tempo.
```

Figura 17: Servizi offerti da Fàgola

```
Your input -> quali sono gli eventi che posso aspettarmi di trovare da Fàgola?
Ecco una lista degli eventi che tendenzialmente organizza la libreria Fàgola settimanalmente:
Lunedì (ogni primo del mese) - Bookclub, un gruppo di persone che si riunisce regolarmente per discutere di un libro scelto di volta in volta;
Martedì - Presentazione di nuove uscite; un evento in cui vengono presentati gli ultimi libri pubblicati
Mercoledì - Serata poetica: una serata dedicata alla poesia, in cui poeti emergenti o affermati possono condividere le proprie poesie
Giovedì - Laboratorio di pittura: un laboratorio che combina diverse tecniche artistiche, come acquerello, pastelli, collage o inchiostri, per creare opere d'arte uniche e sperimentali.
I partecipanti potrebbero essere incoraggiati a esplorare la combinazione di diversi materiali e tecniche.
Venerdì - Workshop di illustrazione: un evento in cui gli illustratori possono imparare nuove tecniche e condividere le proprie esperienze.
```

Figura 18: calendario settimanale eventi

```
Your input -> quali sono gli eventi in programma?
I prossimi eventi in programma sono:
1. IO GELO E ARDO: La declinazione della passione in Carmen, Violetta e Lucia - l'opera tra il teatro e il romanzo
GIOVEDÌ 25 MAGGIO 2023 ORE 17,30 INGRESSO LIBERO
2. LUSAKA IN TOUR: Diego Cassinelli presenta il libro "LUSAKA".
GIOVEDÌ 25 MAGGIO ORE 21,00
Per maggiori informazioni visita la nostra pagina Facebook nella sezione dedicata agli eventi: https://www.facebook.com/FOGOLAFAGOLA/?locale=it\_IT
```

Figura 19: Eventi in programma

Per ognuno di questi casi sono state previste sia una storia breve, in cui l'utente avvia la conversazione chiedendo direttamente ciò di cui ha bisogno, e una storia completa che integra anche la sezione saluti.

Le Figure mostrate, inoltre, evidenziano come le risposte delle richieste di informazioni di base riguardo la libreria Fàgola, siano state implementate semplicemente tramite *utter\_response* con all'interno il testo di risposta di interesse e situate, quindi, nel file *domain.yml*.

#### 4.3 Ricerca sui libri

Come evidenziato nell'idea di progetto nella sezione precedente, l'utente può interagire con il chatbot per ottenere informazioni su un libro di cui conosce il titolo o può chiedere aiuto all'assistente virtuale affinché lo guidi alla ricerca del prodotto basandosi su criteri come il genere, l'autore o l'età consigliata.

Fornita una descrizione generale di ciò che il chatbot può effettuare a questo livello, ora si vede nel dettaglio come tutto ciò può essere realizzato. In particolare, sarà illustrata ciascuna funzionalità poc'anzi annunciata e come essa è stata implementata.

## 4.3 Ricerca sui libri

### 4.3.1 Informazioni su uno specifico libro

Se l'utente è già a conoscenza del titolo del libro, potrà semplicemente digitare lo stesso ed, ottenere, dunque, informazioni a riguardo. In questo caso, infatti, il chatbot restituisce una breve descrizione della trama del libro, l'autore e il genere a cui appartiene. Qualora l'utente non fosse soddisfatto della risposta fornita dal bot o semplicemente fosse interessato ad ulteriori informazioni come il prezzo, l'età consigliata e la posizione del libro, ovvero reparto e scaffale all'interno della libreria può richiederle e il bot sarà pronto per aiutarlo.

In Figura 20 è riportato un estratto di codice in cui sono definiti alcuni esempi di *intent* per la richiesta di informazioni del libro essendo già a conoscenza del titolo del libro stesso.

```
- intent: info_libro
examples: |
    - vorrei informazioni sul libro [1984](nome_libro)
    - sarei interessato a più informazioni sul libro [la divina commedia](nome_libro)
    - vorrei più info riguardo il libro [cime tempestose](nome_libro)
    - puoi darmi informazioni sul libro [il nome della rosa](nome_libro)
    - vorrei informazioni sul libro [il gattopardo](nome_libro)
    - vorrei più info riguardo il libro [se questo è un uomo](nome_libro)
    - vorrei sapere di più su [moby dick](nome_libro)
```

Figura 20: Intent ed esempi per la richiesta di informazioni su un libro data la conoscenza del titolo

Si osservano che gli esempi per l'intent *info\_libro* necessitano dei titoli dei libri di cui si vogliono ottenere informazioni. Tali titoli sono contenuti all'interno della lookup table *nome\_libro*, ovvero un elenco che permette di definire un insieme di valori per un'entità senza dover fornire troppi esempi.

Il passo successivo è stato quello di definire un'azione che fosse in grado di gestire la richiesta di cui si sta discutendo. E' stata creata, dunque, dentro il file *actions.py* la classe *ActionFindInfoLibro* la cui logica del metodo *run*, responsabile delle azioni da svolgere, è brevemente illustrata di seguito:

- Estrazione dallo slot *nome\_libro* il valore corrispondente inserito dall'utente
- Lettura ed estrazione dal file *libri.csv* delle informazioni corrispondenti al titolo del libro inserito. Nel caso in cui non ci fosse nulla, viene gestito tale caso limite con un output opportuno che segnali ciò
- Restituzione in uscita dell'output corrispondente.

A questo punto, però, si è posto un ulteriore problema, ossia quello delle molteplici corrispondenze. Tale limite si presenta ad esempio nel caso di una saga che contiene più libri il cui titolo principale rimane lo stesso seguito da un secondo titolo separato da un trattino dal primo, a specificare appunto alla continuità della storia. Di fatto, se l'utente, volesse avere informazioni su "Il signore degli anelli", lo script python seleziona più righe della tabella: "Il signore degli anelli - la compagnia dell'anello", "Il signore degli anelli - le due torri", "Il signore degli anelli - il ritorno del re". In questi casi, dunque, il chatbot restituisce l'elenco dei libri cui l'utente potrebbe riferirsi, chiedendogli successivamente di riformulare la richiesta con uno di quei libri trovati, eliminando definitivamente l'incertezza. L'azione è stata eseguita attraverso l'utilizzo di una lista e di un ciclo for, il quale scansiona ciascuna riga della tabella per trovare la corrispondenza esatta o in parte del titolo fornito in input dall'utente, aggiunti nella lista e fornita in output all'utente. Tale eventualità è mostrata nella Figura 21, in cui la risposta del chatbot è stata testata tramite la shell di Rasa.

```
Your input -> vorrei informazioni sul libro il signore degli anelli
Ho trovato più libri che corrispondono al titolo che stai cercando. Per favore riformula con uno dei seguenti nomi dei libri:
- il signore degli anelli - la compagnia dell'anello
- il signore degli anelli - le due torri
- il signore degli anelli - il ritorno del re
```

Figura 21: Output nel caso in cui l'utente richiede informazioni su un libro per cui vengono trovate corrispondenze multiple

In aggiunta, al fine di avere una certa continuità di botta e risposta tra l'utente e il chatbot, è stato preferito mandare in output soltanto in parte le informazioni contenute nel file csv relative al libro

### 4.3 Ricerca sui libri

inserito dall'utente ma fornirle in seguito in caso di richiesta; si sta parlando delle richieste sul prezzo, età consigliata e posizione fisica del libro all'interno della libreria, ovvero reparto e scaffale di riferimento. Affinchè ciò avvenga, il chatbot deve tenere memoria dello slot del titolo del libro anche nelle successive risposte; a tal proposito si è fatto uso della classe *SlotSet*, un componente di Rasa che consente di gestire e tenere traccia delle informazioni estratte durante una conversazione all'interno degli slot. Ciò fornisce al chatbot la capacità di personalizzare le risposte, prendere decisioni basate sul contesto e fornire un'esperienza interattiva e mirata all'utente.

Per rendere meglio l'idea di ciò che è stato appena illustrato, è mostrato nella Figura 22 un esempio di conversazione.

```
Your input -> vorrei informazioni sul libro la divina commedia
Il libro 'la divina commedia' è stato scritto da Dante Alighieri e appartiene al genere 'poesia'.
'la divina commedia' è considerata uno dei capolavori più importanti della letteratura italiana e mondiale. La storia è un viaggio allegorico che il protagonista, Dante stesso, intraprende attraverso l'Inferno, il Purgatorio e il Paradiso. Nel "Inferno", Dante descrive il suo percorso attraverso i nove cerchi dell'Inferno, dove incontra anime dannate e personaggi famosi del passato. Nel "Purgatorio", Dante ascende il monte del Purgatorio, dove le anime vengono purificate dai loro peccati prima di raggiungere il Paradiso. Infine, nel "Paradiso", Dante entra nel regno celeste e viene guidato attraverso i nove cieli, fino ad arrivare alla visione di Dio
Puoi chiedermi altro, se vuoi
Your input -> a quale età è consigliato leggere questo libro?
La lettura del libro 'la divina commedia' è consigliata per età a partire da 14+.
Your input -> quanto costa il libro?
Presso la nostra libreria il libro 'la divina commedia' è disponibile al prezzo di € 9.99.
Your input -> in quale reparto posso trovare questo libro?
Il libro 'la divina commedia' è collocato nella seguente posizione:
Reparto:letteratura italiana Scaffale: B2.
Per ulteriori informazioni ti consiglio di rivolgerti al personale della libreria.
```

Figura 22: Esempio di conversazione continuativa per la richiesta di informazioni su un libro specifico

#### 4.3.2 Ricerca per autore

La possibilità di effettuare ricerche per autore rappresenta un'importante funzionalità del chatbot implementato per la libreria. L'utente, infatti, potrebbe essere interessato a una ricerca mirata a partire dal proprio autore preferito. Sulla base della richiesta dell'utente ed, utilizzando le informazioni presenti nel file *libri.csv* della libreria, dunque, il chatbot è in grado di estrarre i titoli dei libri scritti dall'autore richiesto.

Il chatbot così implementato offre agli utenti un modo conveniente per esplorare e scoprire libri scritti dai loro autori preferiti. Infine, i suggerimenti personalizzati facilitano la selezione del libro desiderato, contribuendo a migliorare l'esperienza di ricerca e di lettura degli utenti all'interno della libreria.

In Figura 23 è riportato un estratto di codice in cui sono definiti alcuni esempi di intent per la richiesta di ricerca dei titoli dei libri a partire dall'autore prescelto.

```
- intent: info_autore
examples: |
    - quali libri avete di [George Orwell](autore_libro)?
    - quali libri hai di [Umberto Eco](autore_libro)?
    - mi interessa leggere qualcosa di [Dante Alighieri](autore_libro).
    - mi piacerebbe leggere un libro di [Miguel de Cervantes](autore_libro)
    - quali libri avete a disposizione di [Lev Tolstoj](autore_libro)?
    - quali libri sono disponibili di [Tomasi di Lampedusa](autore_libro)?
    - mi puoi consigliare un libro di [Herman Melville](autore_libro) che avete in libreria?
```

Figura 23: Ricerca di un libro tramite un autore definito

In Figura 24 sono mostrati due esempi di risposta per la ricerca sulla base di un autore definito.

```
Your input -> avete a disposizione qualche libro di Eco
I libri disponibili dell'autore che stai cercando sono i seguenti:
- il nome della rosa
Your input -> quali libri avete di george orwell
I libri disponibili dell'autore che stai cercando sono i seguenti:
- 1984
Your input -> quali libri avete a disposizione di J.R.R. Tolkien
I libri disponibili dell'autore che stai cercando sono i seguenti:
- il signore degli anelli-la compagnia dell'anello
- il signore degli anelli-le due torri
- il signore degli anelli-il ritorno del re
```

Figura 24: Ricerca di un libro tramite un autore definito

A tal proposito, la gestione delle risposte è stata realizzata per mezzo dell'inizializzazione di una lista, inizialmente vuota, per memorizzare i titoli dei libri corrispondenti all'autore richiesto. In seguito alla

### 4.3 Ricerca sui libri

creazione dell'oggetto dataframe per la lettura del file *libri.csv* contenente le informazioni sui libri della libreria, viene eseguito un ciclo for iterando su ciascun riga del dataframe. Se l'autore del libro corrisponde esattamente al nome dell'autore richiesto, il titolo del libro viene aggiunto alla lista; allo stesso modo, se l'autore del libro richiesto è incluso (anche in minuscolo) nell'autore del libro, il titolo viene anch'esso aggiunto alla lista.

Infine, se la lista contiene dei titoli dei libri (ovvero se sono stati trovati dei libri corrispondenti all'autore richiesto), viene generato un messaggio di risposta contenente l'elenco dei titoli dei libri trovati. Tale messaggio viene inviato al mittente utilizzando il dispatcher e successivamente vengono reimpostati tutti gli slot con il metodo *AllSlotReset()*.

In definitiva, l'utente può visionare i prodotti e decidere di digitare uno per ottenere maggiori informazioni; in questo modo si torna alla sezione 2.3.1 Informazioni su uno specifico libro

#### 4.3.3 Ricerca per genere

Qualora l'utente non conosca il titolo specifico del libro, il chatbot assume il ruolo di assistente virtuale per fornire assistenza nella ricerca di un libro basandosi sul criterio del genere letterario.

In Figura 25 è mostrato un estratto di frasi di esempio dell'intent *info\_genere* per effettuare una ricerca digitando il genere letterario desiderato dall'utente.

```
- intent: info_genere
examples: |
    - quali libri del genere [distopico](genere_libro) avete?
    - che tipo di libri di genere [poesia](genere_libro) avete?
    - avete a disposizione qualche libro del genere [storico](genere_libro)?
    - che libri di genere [romantico](genere_libro) avete disponibili?
    - che libri di genere [fantasy](genere_libro) avete in catalogo?
    - mi potete indicare qualche libro del genere [saggio](genere_libro)?
```

Figura 25: Intent ed esempi per la ricerca basandosi sul criterio del genere letterario

Ad ogni modo, la logica di funzionamento dell'azione che si occupa di svolgere la richiesta di riferimento risulta analoga all'azione descritta nella sezione precedente, pertanto la sua descrizione è stata omessa.

In Figura 26 è mostrata una tipologia di conversazione in cui la ricerca di un libro viene effettuata a partire dalla definizione di un genere letterario.

```
Your input -> quali libri del genere fantasy avete?
I libri disponibili per il genere che stai cercando sono i seguenti:
- la storia infinita
- il signore degli anelli-la compagnia dell'anello
- il signore degli anelli-le due torri
- il signore degli anelli-il ritorno del re
```

Figura 26: Ricerca di un libro tramite il genere letterario desiderato

Una volta aver preso visione dei libri per il genere letterario indicato dall'utente, quest'ultimo può decidere di digitare uno per ottenere maggiori informazioni, in questo modo si torna alla sezione 2.3.1 Informazioni su uno specifico libro.

#### 4.3.4 Ricerca per età

Per ultima istanza, l'azione concerne la ricerca per età si concentra sulla capacità del chatbot di aiutare gli utenti a trovare libri adatti a diverse fasce di età. Questa funzionalità potrebbe essere particolarmente utile per genitori, insegnanti o chiunque desideri trovare, ad esempio, letture adeguate per i giovani lettori.

Si è immaginato di avere un utente che desidera trovare libri per bambini di una specifica età, ad esempio 6 anni. In tal senso, l'utente può chiedere al chatbot di consigliare dei libri adatti ai bambini di quell'età. Il chatbot utilizzerà le informazioni presenti nel suo sistema per suggerire titoli appropriati che potrebbero interessare i giovani lettori di 6 anni.

In Figura 27 è mostrata una tipologia di conversazione in cui la ricerca di un libro viene effettuata a partire dalla fascia d'età desiderata dall'utente.

#### 4.4 Prenotazione libro

```
Your input -> ciao
Buongiorno, sono il bot della libreria Fàgola. Posso darti informazioni generali sulla libreria e chiarire quali sono i servizi che offriamo; puoi chiedermi info su un libro specifico oppure effettuare una ricerca per genere, autore, età. Inoltre, sono in grado effettuare una prenotazione di un libro da te desiderato. In cosa posso aiutarti?
Your input -> quali libri ci sono a disposizione per l'età 6+
I libri disponibili per l'età che stai cercando sono i seguenti:
- il piccolo principe
- le avventure di pinocchio
- il mago di oz
```

Figura 27: Ricerca di un libro personalizzata per la fascia d'età 6 anni

Le indicazioni personalizzate facilitano, dunque, la scelta dei libri più adatti ai giovani lettori, migliorando l'esperienza di lettura e aiutando a promuovere il piacere della lettura fin dalla tenera età all'interno della libreria.

#### 4.4 Prenotazione libro

Come affermato in precedenza, il chatbot è in grado anche di effettuare una prenotazione di un libro, verificando innanzitutto la sua disponibilità in libreria all'interno del sistema, e successivamente richiedere il nome dell'utente che intende effettuare l'operazione.

La realizzazione di tale funzionalità consente una maggiore personalizzazione dell'esperienza dell'utente; quest'ultimo, infatti, si sente maggiormente coinvolto e considerato individualmente nel processo di prenotazione, il che aumenta la sua soddisfazione e fiducia nella libreria. In aggiunta, aumenta l'efficienza nella gestione delle prenotazioni, in quanto utilizzando il titolo del libro e il nome dell'utente come punti di riferimento, il chatbot semplifica il processo di gestione delle richieste di prenotazione.

Per implementare la funzionalità della prenotazione, è stata sfruttata una componente che il framework Rasa mette a disposizione, la cosiddetta form, la quale consente di gestire in modo strutturato l'interazione con l'utente durante una conversazione. Esse offrono un modo organizzato per raccogliere e validare le informazioni richieste dall'utente attraverso una serie di domande e risposte. Tale meccanismo si adatta molto bene alla situazione in questione, dato che l'idea è quella di leggere prima il titolo del libro, e soltanto se esso è disponibile per la prenotazione presso la libreria si procede con la seconda domanda relativa al nome dell'utente che intende completare l'operazione.

Innanzitutto, in Figura 29 viene mostrato un estratto di codice che riporta alcuni esempi per l'intent in questione.

```
- intent: prenotazione_libro
examples:
  - vorrei fare la prenotazione di un libro
  - voglio prendere un libro in prestito
  - vorrei qualcosa da leggere
  - Si potrebbe prendere a prestito un libro dalla libreria?
  - è possibile prenotare un libro dalla libreria?
  - vorrei richiedere in prestito un libro
```

Figura 28: Intent ed esempi per la richiesta di prenotazione di un libro

La form vera e propria è stata definita nel file domain.yml, come mostrato in Figura 29.

```
forms:
  prenotazione_libro_form:
    required_slots:
      - nome_libro2
      - disponibilità
      - nome_utente
```

Figura 29: Definizione form per la prenotazione di un libro

#### 4.4 Prenotazione libro

Come si può osservare, la sezione *required\_slots* specifica gli slot che devono essere riempiti dall’utente durante l’interazione con la form.

Nel caso specifico, la form *prenotazione\_libro\_form* richiede il riempimento dei seguenti slot:

- *nome\_libro2*: rappresenta il titolo del libro per il quale l’utente desidera effettuare la prenotazione. Durante l’interazione con la form, Rasa chiederà all’utente di fornire il titolo del libro.
- *disponibilità*: una variabile che indica la disponibilità o meno del libro richiesto. Durante l’interazione con la form, Rasa verificherà la disponibilità del libro nel sistema della libreria.
- *nome\_utente*: rappresenta il nome dell’utente che desidera effettuare la prenotazione. Durante l’interazione con la form, Rasa chiederà all’utente di fornire il proprio nome per poter completare l’operazione.

L’obiettivo della form, dunque, è raccogliere le informazioni necessarie per effettuare una prenotazione di un libro, ovvero il titolo del libro, la disponibilità e il nome dell’utente. Una volta che tutti gli slot richiesti sono stati compilati correttamente, Rasa può utilizzare i dati raccolti per eseguire azioni come la registrazione della prenotazione nel sistema della libreria.

Il meccanismo di memorizzazione, quindi, è reso possibile tramite gli slot, il cui legame con i campi *required\_slot* viene reso esplicito sempre all’interno del file *domain.yml*, come si può osservare in Figura 30.

```

nome_libro2:
  type: text
  influence_conversation: true
  mappings:
    - type: from_entity
      entity: nome_libro2
      conditions:
        - active_loop: prenotazione_libro_form
          requested_slot: nome_libro2
nome_utente:
  type: text
  influence_conversation: true
  mappings:
    - type: from_text
      conditions:
        - active_loop: prenotazione_libro_form
          requested_slot: nome_utente
disponibilità:
  type: bool
  influence_conversation: true
  mappings:
    - type: from_entity
      entity: disponibilità
      conditions:
        - active_loop: prenotazione_libro_form
          requested_slot: disponibilità

```

Figura 30: Slot per form prenotazione di un libro

Un aspetto degno di nota riguarda l’opzione di mappatura *"type: from\_text"* contenuta nello slot richiesto *nome\_utente*, la quale differisce dagli altri due slots. Essa indica, di fatto, che il valore fornito dall’utente dovrebbe essere direttamente assegnato allo slot *nome\_utente* senza alcuna estrazione di entità specifica. Questo approccio è utile se si desidera semplicemente memorizzare qualsiasi cosa l’utente digiti come slot, senza dover identificare una specifica entità correlata.

Il passo successivo ha visto la creazione di un’azione ad hoc che, su specifica richiesta dell’utente, fosse in grado di effettuare la prenotazione di un libro qualora il libro fosse disponibile presso la libreria e gli slots richiesti per la form fossero completati correttamente.

L’azione in questione segue brevemente la seguente logica di funzionamento:

- Sulla base del campo inserito dall’utente estrae il valore dallo slot *nome\_libro2* accedendo alla variabile *slot\_value*, e lo confronta con i titoli dei libri presenti nel file CSV *libri.csv*.

#### 4.4 Prenotazione libro

- Se il titolo del libro inserito corrisponde a un libro disponibile (la colonna "Disponibilità" nel CSV è impostata su "true"), viene impostato lo slot "disponibilità" su True e viene inviato un messaggio di conferma all'utente. Viene inoltre restituito un dizionario contenente gli slot *nome\_libro2* e *disponibilità* con i relativi valori per proseguire la form così come è stata implementata.
- Se il titolo del libro inserito corrisponde a un libro ma non è disponibile (la colonna "Disponibilità" nel CSV è impostata su "false"), viene inviato un messaggio all'utente che indica che il libro non è disponibile per la prenotazione. Viene impostato lo slot "disponibilità" su False e viene resettato lo stato del form, il quale viene disattivato.
- Soltanto se il titolo del libro fosse presente nel database della libreria e, soprattutto, disponibile per la prenotazione, il chatbot sarebbe pronto per chiedere il nome dell'utente, estrarre il valore dal campo inserito dall'utente stesso e inviare, tramite il dispatcher, un messaggio di conferma per notificare dell'operazione conclusa con successo.
- E' stato realizzato anche il caso limite in cui il titolo del libro inserito non corrisponde a nessun libro nel CSV e viene inviato un messaggio all'utente per verificare se il titolo del libro fosse scritto correttamente.

In ultima istanza, sono state introdotte tre regole che hanno permesso di gestire l'intero meccanismo della prenotazione, includendo l'attivazione del form, l'azione di submit e l'interruzione della form stessa. Questo approccio ha consentito di configurare la conversazione in modo da attivare il form corretto e gestire la raccolta delle informazioni necessarie per la prenotazione del libro.

L'estratto di codice relativo a queste regole è riportato nella Figura 31.

```

- rule: Attiva prenotazione form
  steps:
    - intent: prenotazione_libro
    - action: prenotazione_libro_form
    - active_loop: prenotazione_libro_form

- rule: Submit prenotazione form
  condition:
    - active_loop: prenotazione_libro_form
  steps:
    - action: prenotazione_libro_form
    - active_loop: null
    - slot_was_set:
      - requested_slot: null
    - action: utter_submit
    - action: utter_libro_slots
    - action: utter_help
    - action: action_all_slots_reset

- rule: Interruzione in prenotazione form
  condition:
    # Condition that form is active.
    - active_loop: prenotazione_libro_form
  steps:
    # This unhappy path handles the case of an intent `bot_challenge`
    - intent: bot_challenge
    - action: utter_iamabot
    # Return to form after handling the `bot_challenge` intent
    - action: prenotazione_libro_form
    - active_loop: prenotazione_libro_form

```

Figura 31: Regole per form sulla prenotazione di un libro

## 5 Connessione a Telegram

Nella fase finale del progetto, è stato possibile collegare il chatbot a una piattaforma di messaggistica istantanea al fine di renderlo accessibile agli utenti attraverso un'interfaccia moderna, soprattutto tramite smartphone. La piattaforma scelta per questo scopo è stata Telegram, selezionata per la sua semplicità di caricamento del chatbot tramite una procedura guidata interna all'applicazione stessa e per la possibilità di connessione con il servizio ngrok, che consente l'esecuzione del chatbot su un server locale. Nella Figura 32 è mostrata la configurazione necessaria nel file credentials.yml per effettuare il deploy del chatbot su Telegram. Riguardo alla figura, i seguenti elementi sono stati impostati:

```
telegram:  
  access_token: 6297661587:AAETT50AL7bjIjPQYJ0XZV_eJy1DwgSVogk  
  verify: FagolaBot_bot  
  webhook_url: "https://922c-87-18-72-12.eu.ngrok.io/webhooks/telegram/webhook"
```

Figura 32: Configurazione bot telegram

- *access\_token*: il token ottenuto durante la procedura guidata di deploy su Telegram tramite il bot FagolaBot\_bot.
- *verify*: il nome assegnato al bot durante la fase di deploy su Telegram.
- *webhook\_url*: l'endpoint che permette la comunicazione tra la macchina locale e il chatbot. Questo endpoint è stato creato utilizzando ngrok.

### 5.1 Testing

Successivamente, vengono presentati una serie di screenshot che mostrano il chatbot in azione su Telegram, fornendo una dimostrazione pratica dei risultati discussi nelle sezioni precedenti.

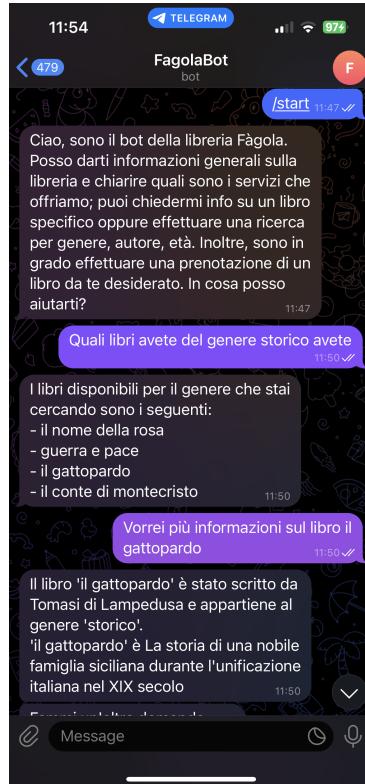


Figura 33: Richiesta informazioni su un libro specifico ed esempio di prenotazione di un libro pt.1

## 5.1 Testing

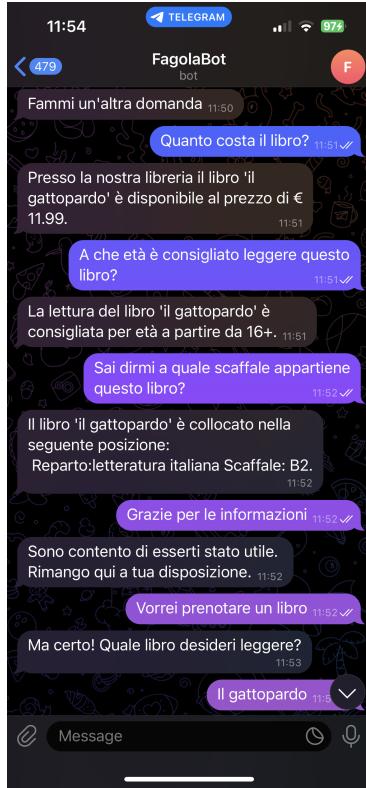


Figura 34: Richiesta informazioni su un libro specifico ed esempio di prenotazione di un libro pt.2

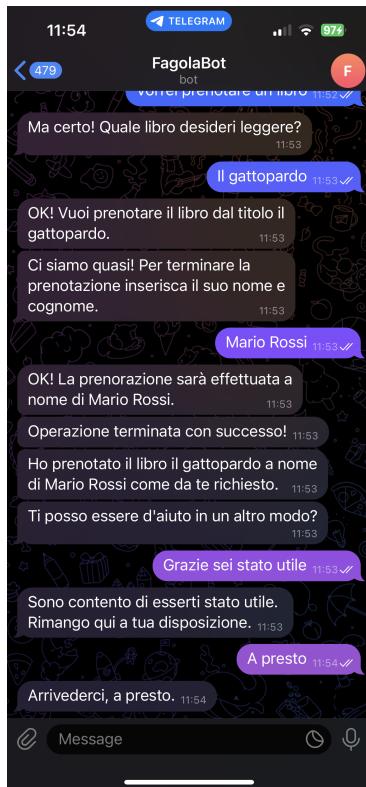


Figura 35: Richiesta informazioni su un libro specifico ed esempio di prenotazione di un libro pt.2

## 5.1 Testing

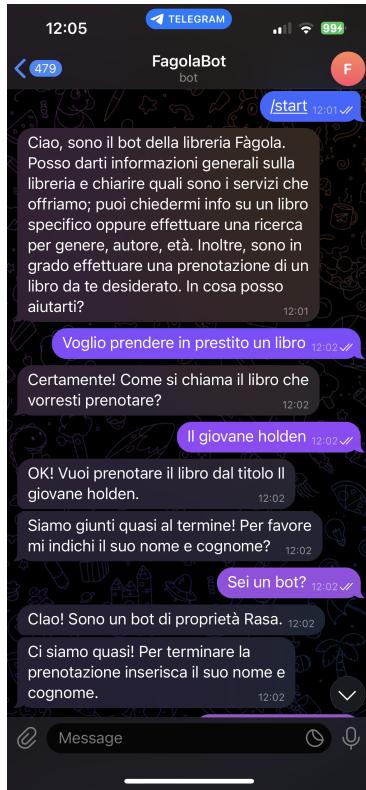


Figura 36: Esempio di interruzione di una form pt.1

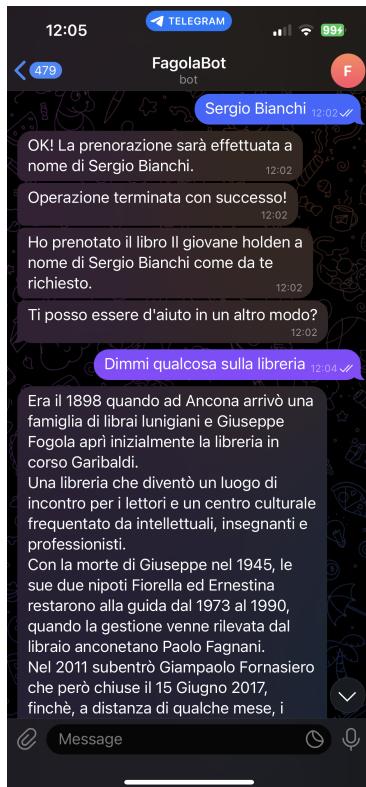


Figura 37: Esempio di interruzione di una form pt.2

## 5.1 Testing

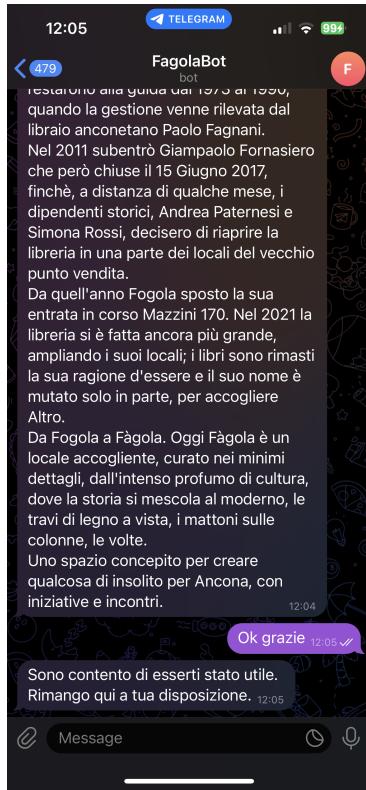


Figura 38: Esempio di interruzione di una form pt.3

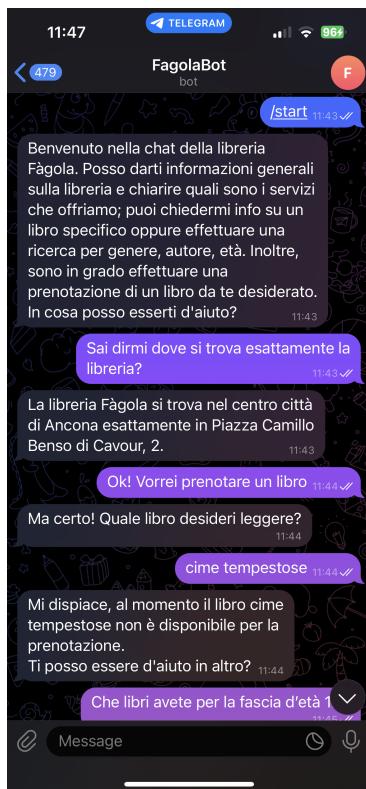


Figura 39: libro non disponibile per la prenotazione e conseguente disattivazione della form pt.1

## 5.1 Testing

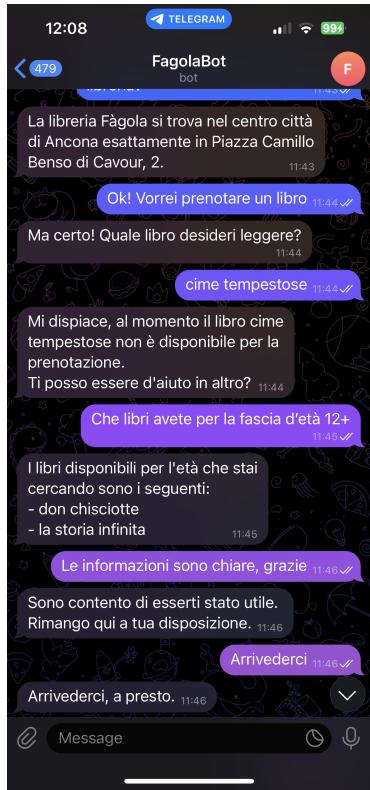


Figura 40: libro non disponibile per la prenotazione e conseguente disattivazione della form pt.2