Università Politecnica delle Marche

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica e dell'Automazione

Chatbot per fornire informazioni sportive tramite framework Rasa

Docenti
Prof. Ursino Domenico
Prof. Marchetti Michele

Studenti
Mori Nicola
Sospetti Mattia
Zitoli Francesca

Anno accademico 2021-2022

Indice

1 Introduzione

Un chatbot è un software in grado di replicare la conversazione con un umano, quindi permette di comunicare con un dispositivo digitale come se si stesse comunicando con una persona reale. La seguente relazione tratterà l'implementazione di un chatbot per fornire informazioni sportive di diversi sport, approfondendo il campionato italiano di calcio. Il chatbot sarà capace di rendere note le informazioni in base allo sport che l'utente intende approfondire, inoltre nel caso del calcio potrà specificare la squadra di Serie A di cui vuole avere le notizie, oltre la possibilità di poter visualizzare sia la classifica del campionato che la classifica dei giocatori che hanno segnato più goal nella competizione.

1.1 Rasa

Rasa è un framework open source in Python per supportare la creazione di chatbot utilizzando una metodologia di machine learning supervisionato. Rasa è composto da due parti **Rasa Core** e **Rasa NLU**; dove la prima parte è il cuore del chatbot ovvero la parte che si occupa dell'apprendimento e sceglie le risposte secondo i modelli che crea durante l'allenamento, la seconda, invece, è il motore di comprensione del linguaggio, ovvero riesce ad analizzare la grammatica e la logica del linguaggio, ovvero in base a cosa ha detto l'utente nel passato e nel presente riesce ad elaborare una risposta.

Un progetto Rasa presenta innanzitutto un file di **domain.yml**, ovvero un file che funge da collante per i vari elementi del chatbot, in quanto in esso sono presenti gli elementi per comprendere il linguaggio, rispondere ed eseguire azioni. Gli elementi che costituiscono il file di domain sono:

- *Intents*: ovvero lo scopo che l'utente ha in fase di conversazione, ovvero è lo strumento che il chatbot utilizza per capire cosa gli sta comunicando l'utente che interagisce.

- *Entities*: sono un oggetto specifico che l'utente sta ricercando all'interno dell'intent, servono per conoscere meglio l'intent della frase.

- *Slots*: sono come delle variabili di linguaggio in cui il chatbot riesce a memorizzare determinate informazioni presenti nell'intent, sono fondamentali per mantenere la traccia di elementi discriminanti per la conversazione.

- *Templates*: sono delle semplici frasi di risposta che il chatbot può utilizzare.

- *Actions*: sono le azioni che il bot può compiere, ovvero funzioni che il chatbot utilizza al verificarsi di determinati intent.

L'apprendimento è supervisionato con un classificatore di frasi. Per i diversi intent presenti nel codice devono essere presenti esempi per fare in modo che il chatbot venga addestrato correttamente. Una volta addestrato l'utente quando invierà un messaggio al chatbot questo lo classificherà in base all'addestramento eseguito assegnando un punteggio ad ogni intent, quello con valore più alto sarà come il chatbot lo interpreterà.

Di fondamentale importanza è il file **nlu.md**, ovvero il file dove sono contenuti gli intent con i loro relativi esempi, in maniera tale che il chatbot riesca a riconoscere, in base agli esempi, a quale intent ricondurre il messaggio inviato dall'utente. Gran parte dell'allenamento del chatbot viene eseguito prendendo in esame questo file.

Un altro file presente in un progetto Rasa è il file **stories.md**, dove vengono definite le 'storie', ovvero delle sequenze di dialogo con cui Rasa riesce a comprendere come rispondere a determinati messaggi.

Il file **rules.yml** invece contiene delle regole specifiche che il chatbot deve rispettare, come ad esempio nel momento in cui un utente invia un messaggio e questo viene classificato con uno specifico intent allora il chatbot dovrà per forza rispondere con una determinata action.

Il file **actions.py** è il file dove si scrivono le custom actions che il chatbot potrà eseguire per soddisfare determinate richieste. Le custom actions è sostanzialmente un'azione che il chatbot che può eseguire seguendo un codice personalizzato dall'utente, come ad esempio contattare un API oppure accedere ad un database.

2 Progettazione

La prima fase del progetto è stata quella di decidere cosa il chatbot dovesse fare, la prima funzione è quella di fornire notizie in merito allo sport che l'utente comunica, quindi durante la conversazione l'utente si ritroverà ad una fase in cui sceglierà lo sport di cui vuole avere le informazioni e quindi il bot dovrà essere in grado di fornirle. L'idea è quella di fornire le ultime 10 notizie di una testata giornalistica con il titolo dell'articolo e il link per raggiungere l'articolo nel caso si vuole approfondire il grado di informazione. Successivamente l'idea è stata quella di approfondire e scegliere lo sport più seguito in Italia, il calcio, e di conseguenza aggiungere un ulteriore grado di profondità d'informazione.

Abbiamo deciso di estrarre news sportive dal sito di "Sport Mediaset" al link, `https://www.sportmediaset.mediaset.it/`, mentre per alcune informazioni peculiari abbiamo deciso di consultare il sito della Lega Serie A al seguente link : `https://www.legaseriea.it/it`.

2.1 Sport

Le notizie che il chatbot deve fornire all'utente, quindi, sono di carattere sportivo, la scelta è stata ponderata in merito agli sport più seguiti in Italia, analizzando nelle varie testate giornalstiche quali fossero quelli maggiormente trattati. Di seguito la lista degli sport dai quali si possono trarre notizie:

- **Football**
- **Basket**
- **Volley**
- **Formula 1**
- **Tennis**
- **Formale E**
- **Sci**
- **Moto GP**

2.2 Calcio

In quanto il calcio è lo sport più seguito in Italia ed in particolare il campionato di Serie A il chatbot sarà in grado di fornire notizie sportive per ogni singola squadra delle 20 che compongono il campionato; oltre a questo sarà possibile anche visualizzare la classifica del campionato e la classifica dei marcatori relativo ad esso. Di seguito le squadre che compongono il campionato:

- Atalanta
- Bologna
- Cagliari
- Empoli
- Fiorentina
- Genoa
- Inter
- Juventus
- Lazio
- Milan
- Napoli
- Roma
- Salernitana
- Sampdoria
- Sassuolo
- Spezia
- Torino
- Udinese
- Venezia
- Verona

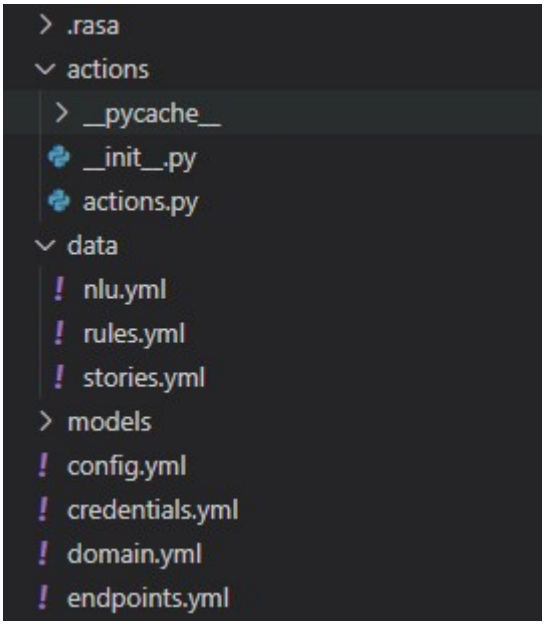

Figura 1: Struttura della cartella di progetto

3 Implementazione

L'implementazione del chatbot è stata eseguita seguendo la creazione del progetto tramite Rasa. All'atto della creazione del progetto Rasa crea la cartella con i file che poi dovranno essere modificati a seconda del chatbot che si vuole implementare.

Come possiamo notare dall'immagine 1 sono già presenti i file che determinano il funzionamento del chatbot, come il file *domain.yml*, inoltre nella cartella data abbiamo i file su cui si basa principalmente il training, quindi *nlu.yml*, *rules.yml*, *stories.yml*.

3.1 Notizie sportive

Il primo problema affrontato è stato quello di permettere all'utente di scegliere lo sport fornendo le notizie relativo ad esso; abbiamo deciso, quindi, che nel momento in cui la persona accede per la prima volta al chatbot di mostrare un messaggio di presentazione e di seguito la richiesta dello sport di cui l'utente vuole avere notizie. Quindi il lavoro si è diviso in due parti, la prima gestire il messaggio di avvio del chatbot al momento della connessione ad esso, il secondo creare un'*intent* che ogni qual volta l'utente dichiara di voler essere aggiornato sulle notizie di un determinato sport allora è in grado di interpretare correttamente il messaggio. Per la prima parte abbiamo de-

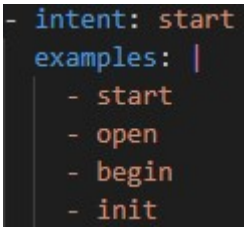

Figura 2: Intent start

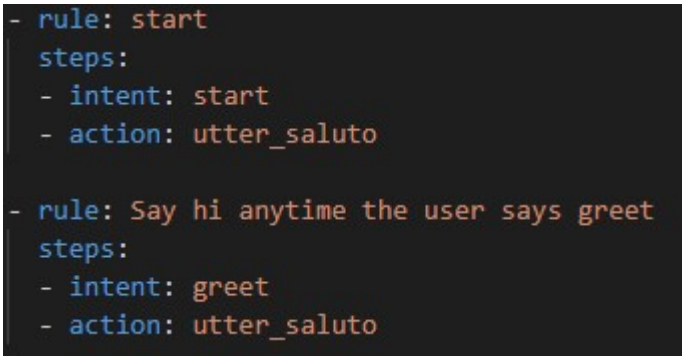

Figura 3: Regole per l'inizio della conversazione

ciso di utilizzare la combinazione di un *intent* nominato **start**, Figura 2 per gestire l'inizio della comunicazione del chatbot.

Da questo abbiamo deciso anche di inserire nel file *rules.yml* due regole: la prima per gestire l'inizio di comunicazione con il chatbot, la seconda invece per gestire l'eventuale saluto che l'utente può rivolgere nei suoi confronti, ad entrambe si risponderà con un messaggio di saluto unito alla richiesta di quale sia lo sport interessato, Figura 3.

Il saluto iniziale invece viene mostrato in Figura 4.

Successivamente quindi una volta l'utente potrà richiedere quale sia lo sport al quale lui sia interessato. Quindi l'obiettivo è quello di creare un *intent* che sia in grado di interpretare le possibili richieste da parte dell'utente; in merito a ciò abbiamo deciso di separare l'*intent* relativo al calcio da quello di tutti gli altri sport, poiché come già detto in fase di progettazione abbiamo ritenuto opportuno approfondire alcuni aspetti del calcio dato che sia lo sport più seguito in Italia. Quindi abbiamo deciso di creare un **intent**

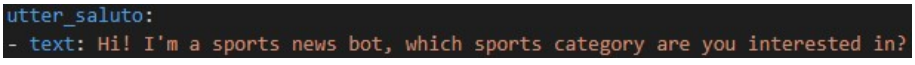

Figura 4: Saluto iniziale

che sia in grado di raccogliere i casi di possibili richieste di sport escluso il calcio. Per fare ciò abbiamo creato un'*entity* nominata **sport**, le entities sono informazioni strutturate all'interno di un messaggio utente. A questo punto abbiamo bisogno di uno *slot*, ovvero di una struttura chiave valore che ci permette di memorizzare delle informazioni che vengono inviate dall'utente durante la conversazione, per questo motivo ne abbiamo creato uno omonimo all'entity, quindi di nome **sport**, dove al suo interno sarà memorizzato lo sport scelto dall'utente, Figura 5.

```yaml
slots:
  sport:
    type: text
    influence_conversation: true
    mappings:
    - type: from_entity
      entity: sport
```

Figura 5: Slot sport

Quindi spostandosi nel file *nlu.yml* abbiamo creato un intent **fs_sport**, nel quale ci sono tutti gli esempi con cui un utente può richiedere di vedere le news di un particolare sport; dalla figura 6 è possibile notare come tra parentesi quadre ci sia il valore che l'entity può assumere, mentre tra parentesi tonde il nome dell'entity al quale ci si riferisce. A questo punto l'intent si attiverà ogni qual volta l'utente chiede di voler vedere le ultime news di uno specifico sport.

```yaml
- intent: fs_sport
  examples: |
    - [tennis](sport)
    - i'd like [basket](sport)
    - i would like to watch [formula1](sport)
    - i would like news about [tennis](sport)
    - i want [motogp](sport) info
    - [f1](sport)
    - I'm interested in [volleyball](sport) news
    - [ski](sport)
```

Figura 6: Intent fs_sport

Il passo successivo è quello di creare un action nel file **action.py** che sia in grado di restituire all'utente le informazioni inerenti allo sport scelto dall'u-

tente. L'action grazie al **Tracker** e la funzione *get_ slot*, alla quale passiamo il nome dello slot, sarà in grado di estrapolare il nome dello sport inserito dall'utente. Una volta acquisito il nome dello sport possiamo contattare il sito dal quale andremo a ricavare le news sportive andando a concentrarci sulle ultime dieci in ordine cronologico, filtriamo il risultato memorizzando solamente il titolo dell'articolo e il link per aprirlo completamente, infine restituiremo dieci elementi composti da titolo e link associato.

L'intero iter è specificato in un'apposita story nel file *stories.yml* dove viene gestito il percorso che porta a questo tipo di conversazione, Figura 7. Nella Figura 8 invece è possibile mostrare un possibile esempio di conversazione con il relativo output.

```
- story: topnewsgenericsport path
  steps:
  - intent: greet
  - action: utter_saluto
  - intent: fs_sport
    entities:
    - sport: sport
  - action: action_particular_topnews
```

Figura 7: Story per la scelta di uno sport generico

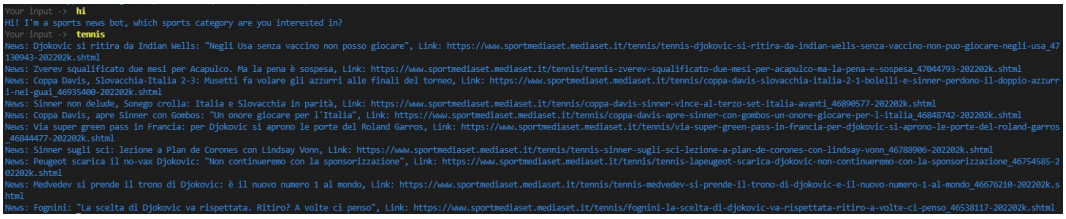

Figura 8: Output notizie di uno sport generico

3.2 Notizie calcistiche

Per le notizie calcistiche abbiamo deciso di costruire delle stories separate in maniera tale che fosse possibile approfondire maggiormente questo ambito. Quindi, per prima cosa abbiamo deciso di creare un *intent* esclusivo, ovvero specifico per quando l'utente decide che lo sport che vuole seguire è il calcio, Figura 9.

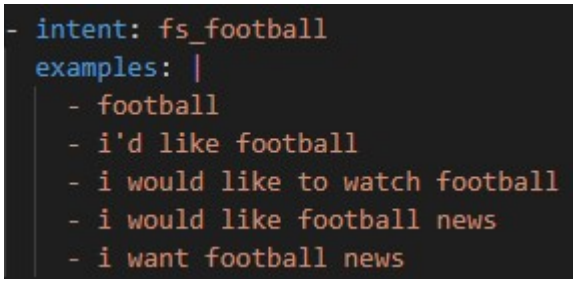

Figura 9: Intent specifico per il calcio

A questo punto il chatbot risponderà chiedendo all'utente se vuole essere informato con delle notizie generiche riguardo il calcio, se invece vuole le notizie di una specifica squadra del campionato di Serie A, se vuole la classifica del campionato o la classifica dei giocatori del campionato che hanno segnato più goal. Per questo motivo abbiamo creato un *intent* per ogni casistica:

- **fs_topnews**: serve per la richiesta dell'utente per le top news relative al calcio, Figura 10.

Figura 10: Intent per le ultime notizie di calcio

- **rank**: sarà utilizzato per classificare il messaggio dell'utente che richiede la classifica del campionato, Figura 11.
- **scorer**: utilizzato se l'utente vorrà visualizzare la classifica dei marcatori, 12.
- **fs_particular**: viene attivato se l'utente richiede di visualizzare le notizie ad un particolare team, Figura 13.

Per quanto riguarda i primi tre *intent* nel file *actions.py* abbiamo implementato delle action apposite per generare la risposta a queste richieste. Per

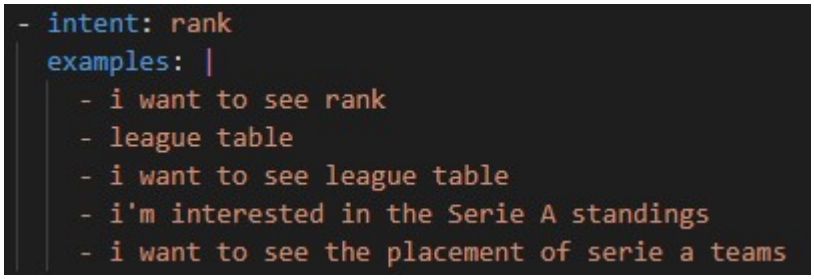

Figura 11: Intent per la classifica

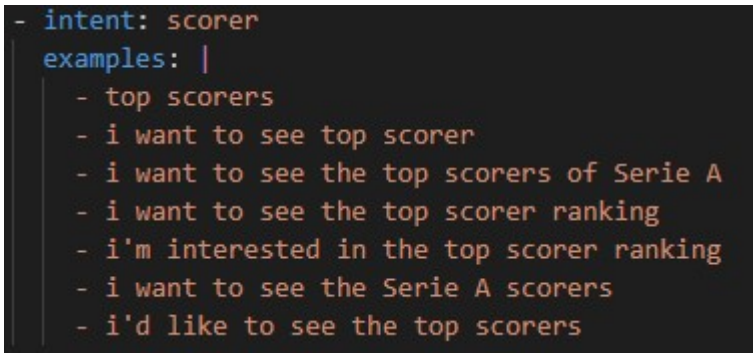

Figura 12: Intent per la classifica marcatori

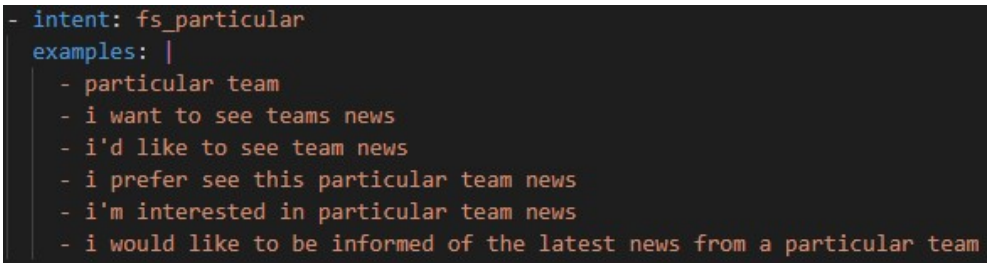

Figura 13: Intent per le news su una particolare squadra

l'intent *fs_ topnews* la funzione è stata costruita sulla falsa riga di quella utilizzata per lo sport generico dove il link per estrarre le notizie verrà costruito inserendo la parola **calcio** che permetterà di entrare nel sito giornalistico nella sezione dello sport in maniera tale che siano recuperabili notizie relativo ad esso. Per il l'intent *rank* invece avremo un'action che dal sito ufficiale della lega della Serie A sarà in grado di estrarre la classifica dalle squadre, quindi poi andremo a restituire in output un messaggio dove sarà possibile visualizzare l'intera classifica del campionato, dove sarà specificata la posizione e i punti acquisiti. Infine per l'intent *scorer* il meccanismo è il medesimo

ovvero dal link della lega della Serie A dove è presente la classifica marcatori sarà possibile ottenere la classifica dei marcatori, dove sarà specificata la posizione, il nome del calciatore e i goal segnati.

Per l'ultimo intent, invece, abbiamo deciso di aggiungere un'ulteriore step, ovvero; nel momento in cui l'utente esprime la necessità di voler ottenere notizie di una specifica squadra il chatbot risponderà con un'ulteriore domanda espressa dall'*utter_football_team*, figura 14, ovvero di quale squadra l'utente è interessato, a questo punto l'utente andrà ad inserire il nome della squadra, per questo motivo abbiamo un nuovo *intent* nominato **squadra** che accoglierà il nome, Figura 15. Per memorizzare il nome abbiamo creato uno *slot* ed un'*entity* dello stesso nome dell'intent, Figura 16. Una volta memorizzato il nome avremo l'attivazione dell'*action_particular*, la quale grazie al nome della squadra memorizzato nello slot, riuscirà ad estrapolare le notizie di una squadra del campionato dal sito utilizzato per le notizie generiche di uno sport.

```
utter_football_team:
- text: Which Serie A team do you want to see the news about?
```

Figura 14: Risposta per la particolare squadra

Per concludere la fase di implementazione delle funzionalità abbiamo creato delle stories nel file *stories.yml* che rispettassero queste sequenze di conversazione come il modello sviluppato nella figura 7 adattata però ai percorsi descritti in precedenza.

Ultimo step della fase di implementazione è stato l'utilizzo dei sinonimi, infatti per generare il link da cui attingere le notizie degli sport abbiamo notato che in diversi sport esistono dei sinonimi, per questo motivo abbiamo fatto in modo che scrivere un determinato sport con un suo sinonimo riconducesse al nome necessario affinché il link da cui si estraggono notizie funzioni, Figura 17

3.3 Connessione a Telegram

Dopo aver implementato le funzionalità del chatbot abbiamo ritenuto che fosse utile collegare una tecnologia di questo tipo con un'applicazione di largo consumo al giorno d'oggi come Telegram, cosicché fosse maggiormente accessibile alle persone. Per fare ciò Rasa permette di modificare il file *credentials.yml* aggiungendo una sezione telegram, come mostrato in Figura 18.

Per il collegamento a Telegram è necessario specificare tre campi:

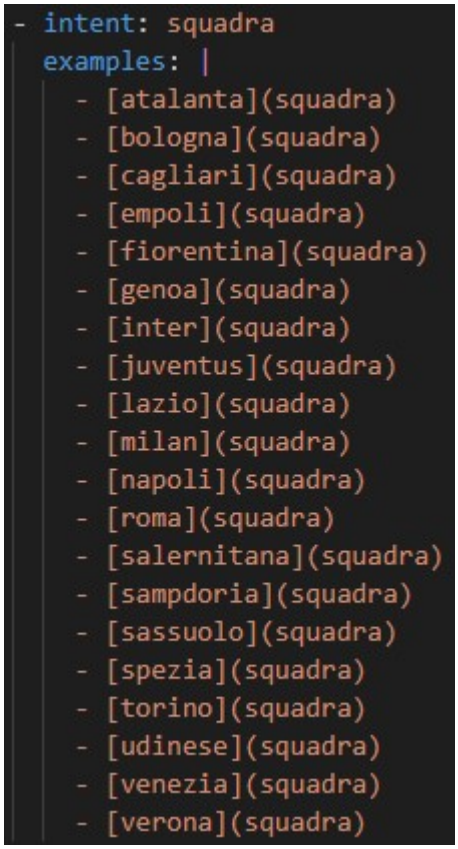

Figura 15: Intent per le squadre

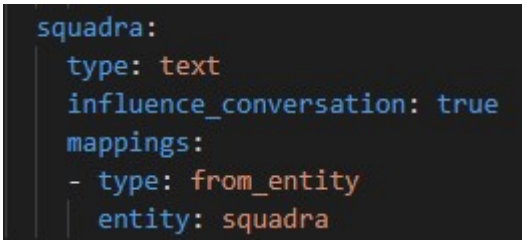

Figura 16: Slot per la squadra

- **access_token**: token che si ottiene da un'applicazione denominata BotFather presente su Telegram che permette la creazione del bot.
- **verify**: nel quale va inserito il nome del bot che abbiamo inserito in fase di creazione del chatbot in BotFather.
- **webhook_url**: si ottiene utilizzando un'applicazione chiamata *ngrok* che restituisce l'indirizzo per la connessione con un server in locale sulla

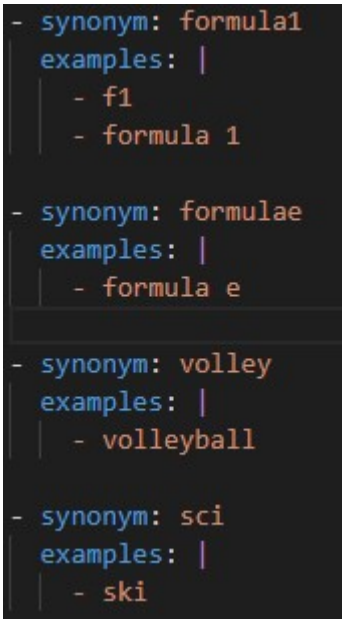

Figura 17: Sinonimi

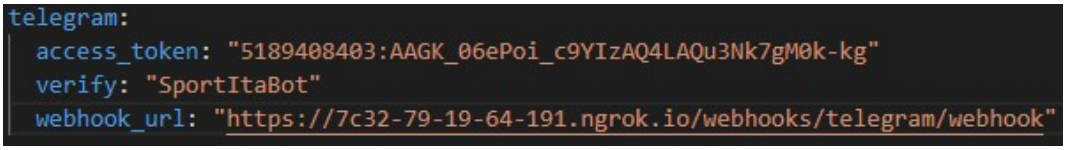

Figura 18: Connessione a telegram

porta 5005. Serve per la comunicazione tra il bot e la macchina.

4 Testing

Di seguito mostriamo gli screen di Telegram che mostrano come le funzionalità implementate vengano eseguite correttamente.

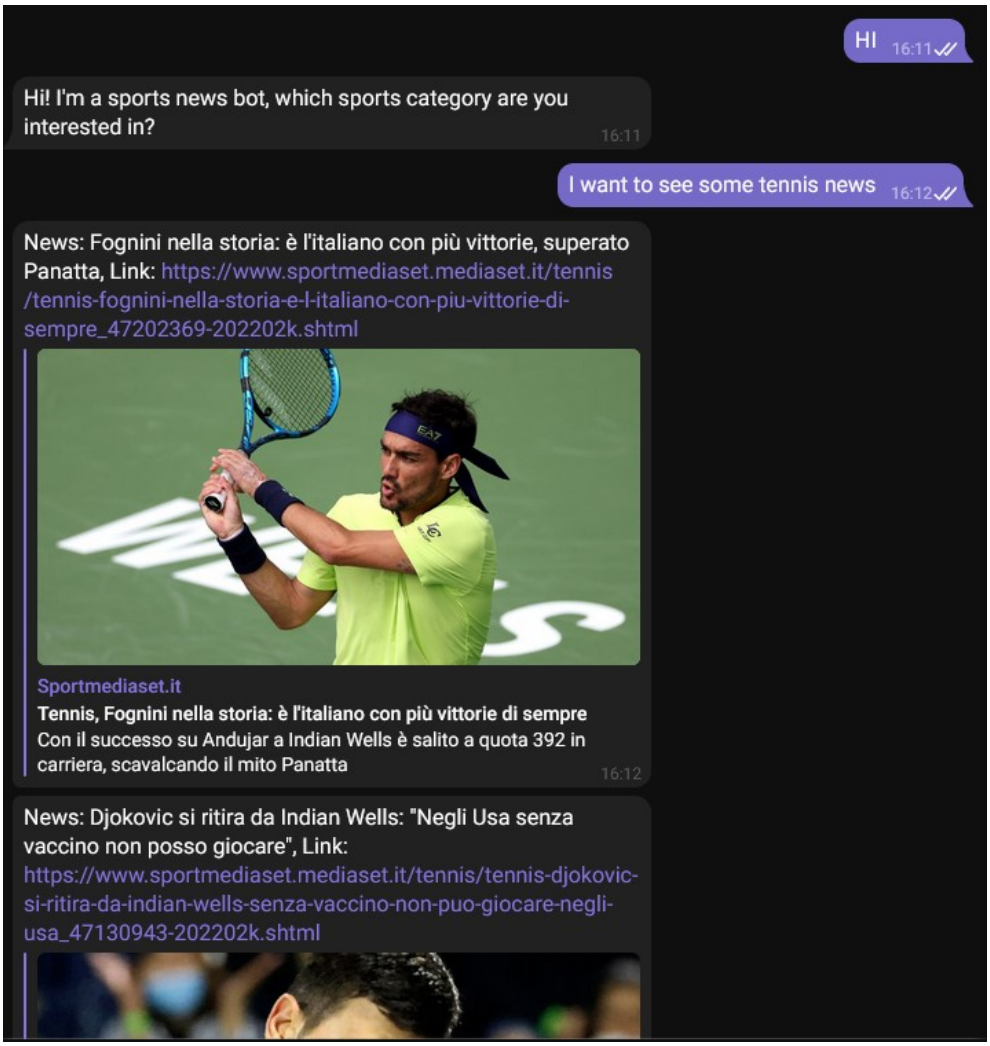

Figura 19: Esempio: Scelta di un generico sport

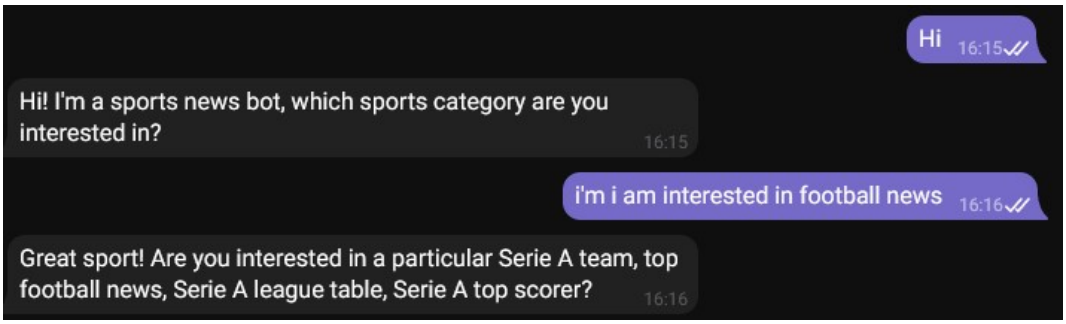

Figura 20: Esempio: Scelta del calcio

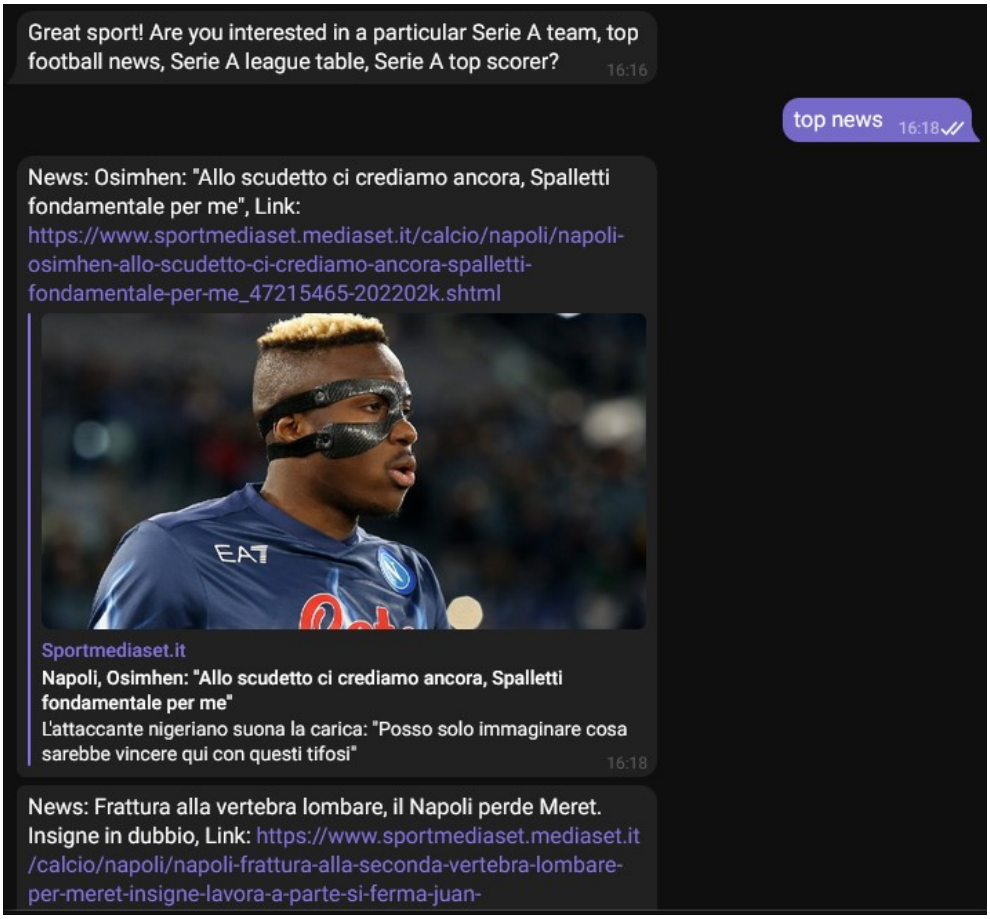

Figura 21: Esempio: Top News relative al calcio

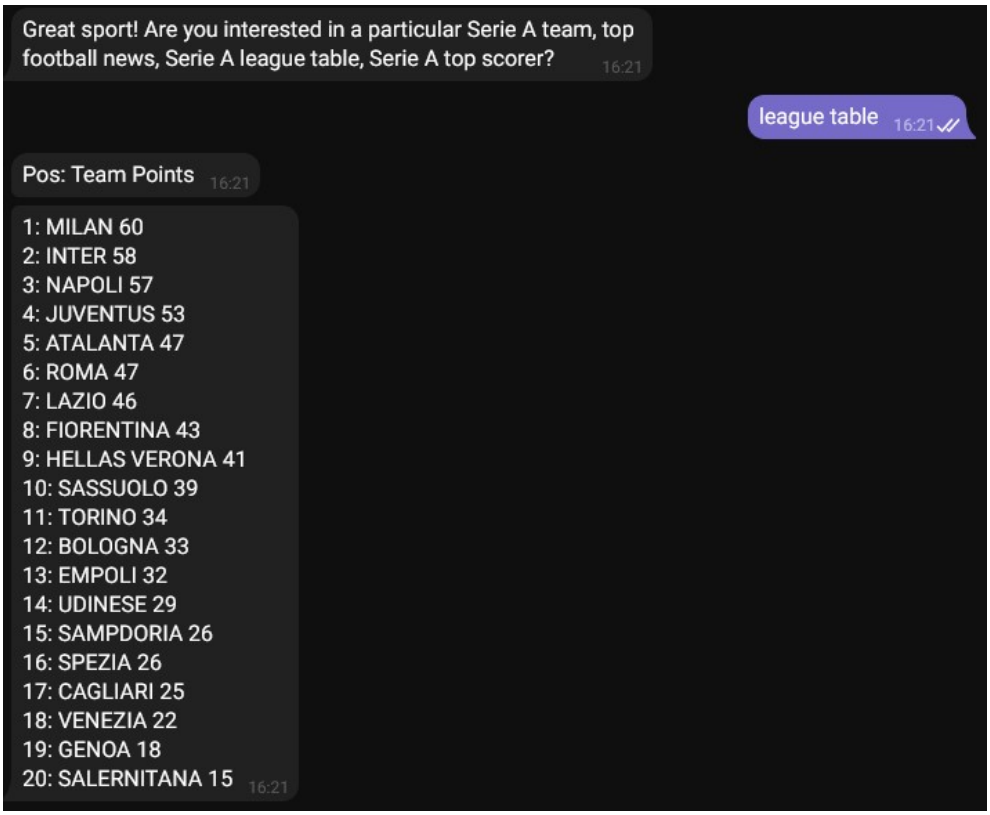

Figura 22: Esempio: Classifica

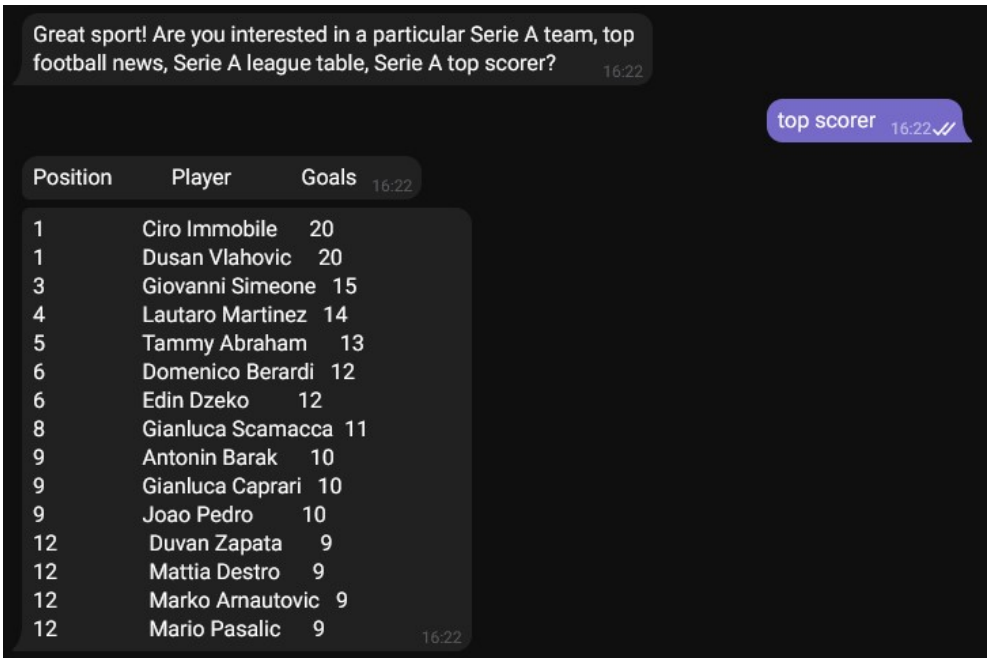

Figura 23: Esempio: Classifica marcatori

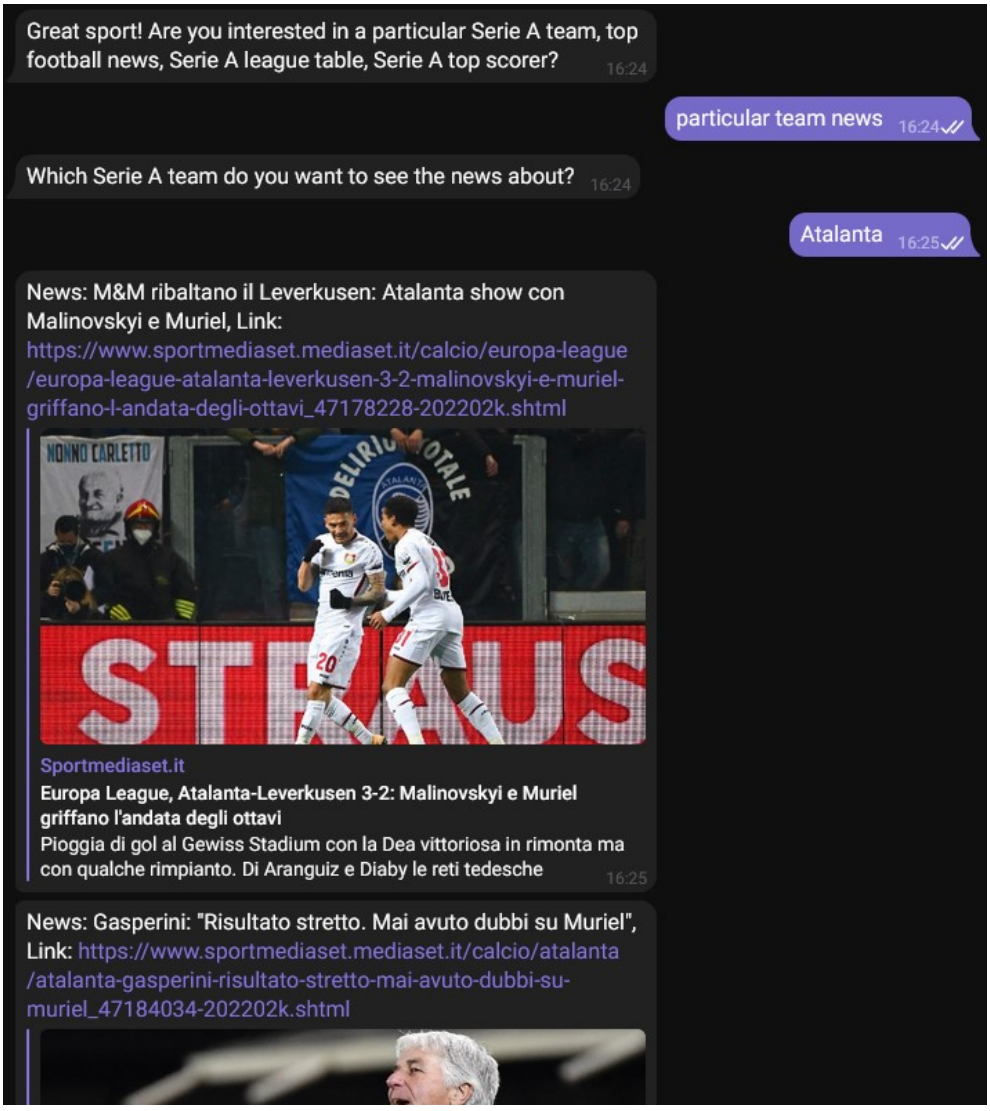

Figura 24: Esempio: Ultime notizie di una particolare squadra