TalkAID

Anna Benedetta Salerno Michele D'Arienzo Raffaele Monti Luigi Petrillo

Samuele Sparno

23 Gennaio 2024

1 Sistema Attuale

Il progetto TALKAID rappresenta davvero un passo avanti significativo nel campo della riabilitazione e del supporto alle persone con disabilità del linguaggio. La possibilità di offrire trattamenti completamente a distanza e in maniera asincrona è un'innovazione che potrebbe aprire nuove opportunità per un numero ancora maggiore di individui affetti da queste patologie.

I metodi tradizionali potrebbero non essere altamente personalizzati alle esigenze specifiche dei pazienti, poiché potrebbe essere difficile adattarli in modo rapido ed efficiente.

La registrazione e il monitoraggio dei progressi dei pazienti potrebbero essere complessa e limitata a causa della mancanza di strumenti tecnologici dedicati.

La componente di Intelligenza Artificiale è ritenuta fondamentale ai nostri obiettivi siccome aggiunge un livello di personalizzazione e adattabilità, permettendo ai pazienti di ricevere esercizi mirati in base al loro grado di severità della patologia. Questo non solo rende il trattamento più efficace, ma anche più agevole per i logopedisti, in quanto saranno aiutati dall'IA nella scelta degli esercizi migliori. I pazienti verranno incoraggiati a perseguire con impegno il percorso di miglioramento attraverso le loro statistiche.

2 Sistema Proposto

Lo scopo del nostro progetto è quello di realizzare un agente intelligente che possa:

- Consigliare un insieme di esercizi mirato per il paziente, basandosi sull'esperienza del paziente o sul lasso di tempo trascorso dall'ultima interazione con quella tipologia di esercizio;
- Migliorare il sistema di consigli nel tempo, facendolo evolvere sulla base dei feedback del logopedista, il quale deciderà se un esercizio è appropriato o meno.

3 Specifica PEAS

Di seguito abbiamo elencato la specifca PEAS dell'agente intelligente.

3.1 Performance:

Le prestazioni dell'agente sono valutate attraverso le seguenti misure:

- La sua capacità di consigliare esercizi più mirati possibile per il paziente in base alla patologia.
- Il punteggio dell'esercizio che il paziente ha effettuato su quell'esercizio.
- Quanto tempo è passato dall'ultima volta che il paziente ha svolto l'esercizio.

3.2 Environment:

L'ambiente è:

- Completamente osservabile, in quanto si ha accesso a tutte le informazioni relative ai pazienti, in particolare le patologie e gli esercizi svolti, e alla lista degli esercizi svolti per ogni paziente.
- Non deterministico, in quanto lo stato dell'ambiente cambia indipendentemente dalle azioni dell'agente.
- Sequenziale, in quanto le esercitazioni effettuate dai pazienti e le scelte del logopedista influenzano le decisioni future dell'agente.
- Dinamico, in quanto nel corso delle elaborazioni dell'agente, un paziente potrebbe svolgere un esercizio, cambiando in tal modo le sue esigenze.
- Discreto, il numero di percezioni dell'agente è limitato in quanto ha un numero discreto di patologie, esercizi, pazienti, azioni e percezioni possibili.
- A singolo agente, in quanto l'unico agente che opera in questo ambiente è quello in oggetto.

3.3 Actuators:

- Pagina web dove viene creata una lista di esercizi in maniera tabellare consigliata per ogni paziente.

3.4 Sensors:

- Gli esercizi svolti dal paziente, gli esercizi assegnati al paziente da parte del logopedista, il punteggio per esercizio del paziente, e il tempo passato dall'ultima volta che il paziente ha effettuato l'esercizio.
- Il dataset è un riadattamento del database utilizzato dal sito web principale, eliminando gli attributi non necessari e aggiungendo gli attributi gravità di lettura e scrittura, necessari al fine di poter avere una più completa visualizzazione delle necessità reali del paziente.

4 Soluzione proposta

Date le nostre necessità, abbiamo potuto constatare che ciò di cui abbiamo bisogno è un algoritmo di ottimizzazione.

Nonostante un'attenta valutazione di vari algoritmi, tra i quali spicca l'utilizzo di tecniche come la segmentazione degli utenti o il collaborative filtering, in particolare il clustering potrebbe aiutare a limitare quali esercizi consigliare in base alle patologie del paziente. Purtroppo, al momento non esistono dataset pertinenti per la nostra valutazione, quindi disponiamo di un insieme di dati limitato che non permette a un ipotetico algoritmo di apprendimento di effettuare un training ottimale.

Abbiamo quindi optato per un algoritmo di ricerca locale genetico, poiché è in grado di individuare un punto ottimo tra le diverse alternative, producendo soluzioni sempre migliori rispetto a una funzione obiettivo, anche in assenza di un dataset molto esteso. È importante notare che ciò non garantisce l'ottimalità, dato che solitamente produce soluzioni sub-ottimali. Proprio per questo motivo, affidiamo il lavoro di supervisione al logopedista.

Il nostro obiettivo è ottenere una lista di esercizi per ciascun paziente che possa soddisfare le sue specifiche esigenze. Questa lista sarà poi presa in considerazione dal logopedista. Di conseguenza, potremmo dire che ogni individuo sarà associato a un insieme di esercizi raccomandati specifici per quel determinato paziente.

5 Raccolta e sviluppo del dataset

Avendo la necessità di utilizzare un dataset sul quale il nostro modello avrebbe dovuto estrapolare le informazioni riguardanti gli esercizi e i pazienti, la sfida consisteva in due opzioni:

- Cercare un dataset pre-esistente al fine di avere molte informazioni, e al massimo effettuare data cleaning e adeguarlo a quello che ci serve.
- Creare un dataset, formulando gli esercizi, aggiungendo i pazienti e creando le varie valutazioni per ogni esercizio.

Purtroppo cercare un dataset pre-esistente non è stato proficuo data l'unicità delle nostre richieste; infatti, non abbiamo trovato dataset che riguardassero nel dettaglio la valutazione di esercizi logopedici.

Di conseguenza, abbiamo deciso di creare un dataset nostro, utilizzando un database in MySQL e generando le varie tipologie di esercizi di nostra iniziativa, ottenendo 84 esercizi. La generazione dei pazienti e la generazione delle valutazioni degli esercizi sono state prodotte sinteticamente mediante l'utilizzo di valori randomici.

- Di seguito vengono mostrati i vari codici in SQL realizzati per poter popolare il nostro database.

5.1 Popolazione del DataBase

```
    INSERT INTO exercise (ID_user, ID_exercise, InsertionDate, CompletionDate, Evaluation, Feedback)
    VALUES
    (
        FLOOR(904 + RAND() * (1003 - 904 + 1)), -- Random user ID
        FLOOR(1 + RAND() * (84 - 1 + 1)), -- Random exercise ID
        DATE_SUB(NOW(), INTERVAL FLOOR(RAND() * 365) DAY), -- Random date within the past year
        DATE_SUB(NOW(), INTERVAL FLOOR(RAND() * 365) DAY), -- Random completion date within the past year
        FLOOR(RAND() * (100 - 0 + 1)), -- Random evaluation between 0 and 100
        CASE
            WHEN RAND() <= 0.33 THEN -1
            WHEN RAND() <= 0.66 THEN 0
            ELSE 1
        END -- Random feedback (-1, 0, or 1)
    );

```

Listing 1: Generazione Casuale esecuzione esercizio

```
    DELIMITER //

    CREATE PROCEDURE GenerateUsers()
    BEGIN
        DECLARE i INT DEFAULT 1;
        DECLARE j INT;

        WHILE i <= 10 DO
            SET j = 1;
            WHILE j <= 10 DO
                INSERT INTO user (ID_Therapist) VALUES (i);
                SET j = j + 1;
            END WHILE;
            SET i = i + 1;
        END WHILE;
    END //

    DELIMITER ;

    CALL GenerateUsers();

```

Listing 2: Generazione degli utenti

```
    INSERT INTO patientcondition (ID_condition, ID_patient, Severity,
WritingSeverity, ReadingSeverity)
    SELECT
        FLOOR(1 + RAND() * 12) as ID_condition, -- Random condition ID (1 to 12)
        u.ID as ID_patient,
        1 as Severity,
        FLOOR(1 + RAND() * 10) as WritingSeverity,
        FLOOR(1 + RAND() * 10) as ReadingSeverity
    FROM
        user u
    WHERE
        u.ID_Therapist != 0;

```

Listing 3: Generazione delle severity (tutti gli user)

```
    UPDATE patientcondition pc
    SET
        pc.Severity = LEAST(GREATEST(pc.WritingSeverity, pc.ReadingSeverity) + IF(
RAND() < 0.5, -1, 1), 10)
    WHERE
        pc.Severity = 1;

```

Listing 4: Generazione delle severity (tutti gli user) pt.2

```
    INSERT INTO patientcondition (ID_condition, ID_patient, Severity,
WritingSeverity, ReadingSeverity)
    SELECT
        FLOOR(1 + RAND() * 12) as ID_condition, -- Random condition ID (1 to 12)
        u.ID as ID_patient,
        1 as Severity,
        FLOOR(1 + RAND() * 10) as WritingSeverity,
        FLOOR(1 + RAND() * 10) as ReadingSeverity
    FROM
        user u
    WHERE
        u.ID_Therapist != 0
    ORDER BY
        RAND() -- Randomly order the users and pick the first one
    LIMIT 30;

```

Listing 5: Generazione delle severity (solo x users)

5.2 Analisi del DataBase

In questa sezione, illustriamo il processo di ideazione delle tabelle, riportando il loro nome, descrivendone la funzione e indicando gli attributi contenuti.

- condition - Contiene tutte le informazioni sulle patologie degli utenti, importante per le relazioni tra esercizi e patologie.

Tabella 1: condition Table

Attributo	**Descrizione**
ID_condition	id della patologia
Description	serve per capire cosa provoca la patologia, non è un dato rilevante per il nostro agente
Name	nome della patologia

- exercise - Contiene tutti gli esercizi svolti, utile per poter tenere traccia dell'andamento dei vari pazienti.

Tabella 2: exercise Table

Attributo	**Descrizione**
ID_user	id del paziente associato all'esercizio
ID_exercise	id dell'esercizio che ha svolto il paziente
InsertionDate	data di assegnazione dell'esercizio da parte del logopedista
CompletionDate	data di completamento dell'esercizio
Evaluation	che punteggio il paziente ha raggiunto
Feedback	feedback per mostrare se l'esercizio è piaciuto o meno

- exercise_glossary - Contiene le varie informazioni riguardanti gli esercizi, interessante poiché ci permette di poter trovare il target specifico di un determinato esercizio.

Tabella 3: execise_glossary Table

Attributo	**Descrizione**
ID_exercise	id dell'esercizio
ExerciseName	nome dell'esercizio
ExerciseDescription	breve descrizione dell'esercizio
Type	tipo dell'esercizio, può essere un esercizio di lettura testo, associare immagini, trovare la frase corretta etc...
Difficulty	grado di difficoltà dell'esercizio
Target	il target è la patologia alla quale questo esercizio è mirato

- patient_condition - Contiene le informazioni riguardo la patologia che affligge il paziente, in base a questa il nostro agente potrà selezionare esercizi specifici per il paziente.

Tabella 4: patient_condition Table

Attributo	**Descrizione**
ID_condition	id della patologia affetta dal paziente
ID_patient	id del paziente riferito
Severity	gravità della patologia del paziente
WritingSeverity	gravità della condizione del paziente nello scrivere
ReadingSeverity	gravità della condizione del paziente nel leggere

5.3 Interazioni con il DataBase

Le interazioni effettuate dal nostro agente sul database sono esclusivamente operazioni di estrapolazione dati, elencando le funzioni create sono:

Per quanto riguarda le estrazioni degli esercizi:

- select_exercises_not_done
- select_done_exercises
- select_random_exercise

Per quanto riguarda le estrazioni dei pazienti:

- informationUser

5.3.1 Estrazione degli esercizi

Successivamente vi è un estratto del codice dove viene mostrata ogni funzione inerente alle interazioni con il database.

select_exercises_not_done: Questa funzione permette di selezionare gli esercizi non ancora svolti dall'paziente.

```
        def select_exercises_not_done(ID: int) -> list:
    connessione = Connector()
    lista = []
    cursor = None
    try:
        if connessione.get_connection() is not None:
            cursor = connessione.get_connection().cursor(dictionary=True)
            query = """
                        SELECT *
                        FROM exercise_glossary eg
                        WHERE NOT EXISTS (
                            SELECT 1
                            FROM exercise e
                            WHERE e.ID_exercise = eg.ID_exercise AND e.ID_user = %s);
                    """

            parametro = (ID,)
            cursor.execute(query, parametro)

            records = cursor.fetchall()

            for record in records:
                esercizio = Exercise(record["ID_Exercise"],
                                     record["Difficulty"],
                                     record["Target"],
                                     record["Type"],
                                     None,
                                     None,
                                     None)
                lista.append(esercizio)
            return lista

    except mysql.connector.Error as e:
        print("Error while connecting to MySQL ", e)
        return list()
    finally:
        if connessione.get_connection() is not None:
            if cursor is not None:
                cursor.close()
            connessione.get_connection().close()
```

select_done_exercises: Questa funzione permette il recupero egli esercizi svolti dal paziente.

```
def select_done_exercises(ID: int) -> dict:
    connessione = Connector()
    cursor = None
    esercizi = {}
    try:
        if connessione.get_connection() is not None:
            cursor = connessione.get_connection().cursor(dictionary=True)
            query = """
                        SELECT
                            e.ID_exercise AS ExerciseID,
                            eg.Difficulty AS ExerciseDifficulty,
                            eg.Type AS ExerciseType,
                            eg.Target AS ExerciseTarget,
                            e.Evaluation AS ExerciseEvaluation,
                            e.CompletionDate AS ExerciseCompletionDate,
                            e.Feedback AS ExerciseFeedback,
                            DATE_FORMAT(e.CompletionDate, '%Y-%m-%d') AS Exercis
    eCompletionDate,
                            e.Evaluation AS ExerciseEvaluation,
                            e.Feedback AS ExerciseFeedback
                        FROM
                            exercise e
                        JOIN
                            exercise_glossary eg ON e.ID_exercise = eg.ID_exercise
                        WHERE
                            e.ID_user = %s
                        ORDER BY
                            e.CompletionDate DESC
                        LIMIT 50;
                    """

            parametro = (ID,)
            cursor.execute(query, parametro)

            records = cursor.fetchall()

            for record in records:
                esercizio = Exercise(record["ExerciseID"],
                                     record["ExerciseDifficulty"],
                                     record["ExerciseTarget"],
                                     record["ExerciseType"],
                                     record["ExerciseEvaluation"],
                                     record["ExerciseCompletionDate"],
                                     record["ExerciseFeedback"])
                esercizi[record["ExerciseID"]] = esercizio
            return esercizi

    except mysql.connector.Error as e:
        print("Error while connecting to MySQL", e)
        return dict()
    finally:
        if connessione.get_connection() is not None:
            if cursor is not None:
                cursor.close()
            connessione.get_connection().close()
```

select_random_exercise: Questa funzione recupera casualmente alcuni esercizi e, nel caso in cui vi siano esercizi precedentemente completati dal paziente, restituisce anche i relativi risultati.

```
def select_random_exercise(n: int, ID: int) -> list[Exercise]:
    connessione = Connector()
    lst = list()
    cursor = None
    try:
        if connessione.get_connection() is not None:
            cursor = connessione.get_connection().cursor(dictionary=True)
            query = """
                        SELECT
                          eg_random.ID_exercise,
                          eg_random.Difficulty,
                          eg_random.Target,
                          eg_random.Type,
                          DATE_FORMAT(e.CompletionDate, '%Y-%m-%d') AS Exercis
    eCompletionDate,
                          e.Evaluation,
                          e.Feedback
                        FROM (
                          SELECT *
                          FROM exercise_glossary
                          ORDER BY RAND()
                          LIMIT %s
                        ) AS eg_random
                        LEFT JOIN exercise e ON eg_random.ID_exercise = e.ID_exercise
                          AND e.ID_user = %s
                          AND e.InsertionDate = (
                            SELECT MAX(InsertionDate)
                            FROM exercise
                            WHERE ID_exercise = eg_random.ID_exercise
                              AND ID_user = %s
                          )
                        ORDER BY e.InsertionDate DESC;
                    """
            parametro = (n, ID, ID,)
            cursor.execute(query, parametro)
            records = cursor.fetchall()

            if records is not None:
                for record in records:
                    esercizio = Exercise(record["ID_exercise"],
                                         record["Difficulty"],
                                         record["Target"],
                                         record["Type"],
                                         record["Evaluation"],
                                         record["ExerciseCompletionDate"],
                                         record["Feedback"])
                    lst.append(esercizio)

            return lst

    except mysql.connector.Error as e:
        print("Error while connecting to MySQL ", e)
        return list()
    finally:
        if connessione.get_connection() is not None:
            if cursor is not None:
                cursor.close()
            connessione.get_connection().close()
```

5.3.2 Estrazione dei pazienti

informationUser: Questa funzione permette il recupero di informazioni inerenti alle varie patologie che il paziente soffre con il livello di gravità per ogniuna di essa.

```
def informationUser(ID: int) -> dict:
    connessione = Connector()
    cursor = None
    patologie = {}
    try:
        if connessione.get_connection() is not None:
            cursor = connessione.get_connection().cursor(dictionary=True)
            query = """
                    SELECT
                        c.Name,
                        pc.Severity,
                        pc.WritingSeverity,
                        pc.ReadingSeverity
                    FROM
                        patientcondition pc
                    JOIN
                        `condition` c ON pc.ID_condition = c.ID_condition
                    WHERE
                        pc.ID_patient = %s;
                            """
            parametro = (ID,)
            cursor.execute(query, parametro)

            records = cursor.fetchall()

            if len(records) == 0:
                return dict()
            else:
                for record in records:
                    tupla = (record["Severity"],
                             record["WritingSeverity"],
                             record["ReadingSeverity"])
                    patologie[record["Name"]] = tupla
                return patologie

    except mysql.connector.Error as e:
        print("Error while connecting to MySQL ", e)
        return dict()
    finally:
        if connessione.get_connection() is not None:
            if cursor is not None:
                cursor.close()
            connessione.get_connection().close()
```

6 Studio della funzione di fitness

abbiamo realizzato 7 costanti per identificare l'importanza di quel parametro, in base a quelle definiamo quanto quel parametro è importante

dato che volevamo valutare piu' opzioni, e nella speranza che un giorno ci saranno piu' dati, abbiamo valutato piu' opzioni

7 Sviluppo del nostro algoritmo genetico