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.

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.

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.

Actuators:

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

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 SQL 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.

```
    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)

6 Studio della funzione di fitness

7 Sviluppo del nostro algoritmo genetico

Convergenza Per convergenza si intende la capacità (obiettivo) di un GA di migliorare iterativamente le soluzioni candidate verso il punto di ottimo. Quando gli individui tendono a somigliarsi troppo, bloccando troppo presto il progresso globale dell'intera popolazione, parliamo di convergenza prematura.

Diversità Per diversità si intende la capacità di un GA di definire individui che possano navigare il panorama degli stati in maniera efficace, evitando la stagnazione verso punti di ottimo locale e soprattutto bisogna evitare che gli individui siano troppo simili il che porterebbe alla terminazione del GA.

Funzione di fitness Rappresenta un elemento chiave per la definizione di un algoritmo genetico. Formalmente parlando, la funzione di fitness è una funzione in grado di associare un valore a ogni soluzione. Misura il livello di adeguatezza degli individui rispetto al problema considerato e, inevitabilmente, guida il processo di selezione. Chiaramente, la funzione dipenderà dal problema in esame e dovrà ben rappresentare il grado di adeguatezza di una soluzione.

Nulla vieta di combinare più elementi ad esempio, una combinazione lineare di più fattori. In tutti i casi, indipendentemente da come impostiamo la funzione di fitness, stiamo ancora parlando di funzioni mono-obiettivo, in quanto l'obiettivo è formalizzato usando un'unica funzione obiettivo.

1. Size (Popolazione) - Numero individui di ciascuna generazione. - Se la size è fissa, bisogna decidere il numero; - Se la size è variabile, è opportuno decidere una dimensione massima. La ricerca potrebbe diventare eccessivamente lenta.

2. Size Mating Pool - Numero di individui selezionati e che partecipano alla riproduzione. Più grande è la size, più lento sarà l'algoritmo. Più piccola è la size, più limitata sarà la diversità che l'algoritmo garantirà.

3. Probabilità crossover - Con quale probabilità avviene il crossover tra due genitori. - Possiamo stabilire una probabilità che l'algoritmo userà per valutare se effettuare l'operazione di crossover o meno. Più alta è la probabilità, più spesso i genitori produrranno nuovi individui e, quindi, più diversità ci sarà in termini di soluzioni. - Anche in questo caso, però, bisogna stare attenti. Più alta sarà la probabilità, più è probabile che l'algoritmo navighi lo spazio di ricerca senza andare a convergenza. Più bassa sarà la probabilità, più è probabile una convergenza prematura.

4. Probabilità mutazione - Con quale probabilità avviene una mutazione. Considerazioni simili all precedenti.

5. Inizializzazione - Con quale criterio si crea la prima generazione. - Possiamo creare una popolazione iniziale in maniera totalmente casuale. - Oppure, possiamo basarci su euristiche. Se riusciamo a rendere i GA meno ciechi (dando informazioni problem-specific) possiamo evitare di iniziare con individui quasi sicuramente pessimi (se non addirittura inammissibili). - Se possibile, basarsi su euristiche è desiderabile: la ricerca può essere immediatamente guidata verso soluzioni ammissibili/migliori, riducendo il carico computazionale.

6. Rappresentazione individui - Tipo di codifica degli individui. - Abbiamo visto, nell'esempio, l'uso di una rappresentazione basata su stringa binaria. - Ma possiamo rappresentare gli individui come meglio crediamo per il problema in esame. Altre rappresentazioni molto usate sono le stringhe di interi/reali o addirittura gli alberi.

7. Algoritmo di selezione - Come avviene la selezione degli individui. - Casuale: sconsigliata in molti casi, poiché aumenta le chance di non convergenza. - Roulette Wheel, visto nell'esempio: gli individui ricevono una probabilità di selezione pari al valore della loro fitness relativa all'intera popolazione. Molto fedele alla natura, ma un individuo "molto forte" verrà selezionato troppe volte, rischiando la convergenza prematura. - Rank: si compie un ordinamento totale degli individui rispetto alla fitness e si assegna a ciascuno individuo il rango in base alla posizione (e.g., il primo otterrà rango 1, il secondo rango 2, e così via).Ciascun individuo riceve una probabilità di selezione inversamente proporzionata al rango. - Truncation: si compie un ordinamento totale in maniera analoga ad una Rank Selection, ma la selezione non avviene su base casuale quanto con una selezione rigida dei migliori M (¡ n) individui. Non possono esserci selezioni ripetute. Ha vantaggi e svantaggi simili a Rank, ma toglie una componente di casualità, introducendo l'onere di definire il valore di un ulteriore parametro, i.e., M (quale è un valore ottimale?) - K-way tournament: K (¡ n) individui sono selezionati casualmente (formando il torneo) e il migliore tra questi K passa la selezione definitivamente. Si ripete finché non si arriva ad M tornei (e quindi M vincitori). Si possono avere selezioni ripetute. E' poco complessa ed ha interessanti somiglianze con la natura (legge del più forte, letteralmente), tutti motivi per cui è una scelta molto popolare. Tuttavia impone l'onore di una buona scelta di K ed M. Si può adottare una pressione di selezione: invece di far vincere sempre il migliore (assoluto) di un torneo, ogni individuo ha una probabilità di vittoria proporzionata alla sua fitness. In sostanza, i singoli tornei diventano delle piccole Roulette Wheel. Si possono impedire le ripetizioni (in tal caso, se il mating pool è grande quanto la popolazione, ogni individuo ha la garanzia di partecipare ad esattamente K tornei). Osserviamo che con K=1, si ha una Truncation Selection. Di fatti, si procederebbe selezionando gli individui migliori fino ad M, escludendo gli altri.

8. Algoritmo di crossover Come avviene il crossover tra individui. - Single Point: visto nell'esempio, si seleziona un punto del patrimonio genetico dei genitori e si procede alla generazione di due figli tramite scambio di cromosomi. - Two-Point: simile al single point, va a considerare però due punti di incrocio. Aumenta chiaramente la diversità dei geni degli individui generati. - K-Point: Generalizzazione dei precedenti, è poco utilizzato dal momento che richiede la definizione di una strategia di combinazione dei K cromosomi dei genitori. - Uniform: Ciascun gene i-esimo viene scelto casualmente tra i due geni i-esimi dei genitori. Ha il massimo grado di casualità. E' un particolare caso di K Point con K=Size_Individuo. - Arithmetic: Ciascun gene i-esimo è il valor medio dei geni i-esimi dei genitori. Valido solo per rappresentazioni numeriche (precisamente, per le rappresentazione per cui ha senso calcolare il mid-point). Ne segue che i due figli saranno gemelli. —¿ Per differenziare i due figli, è possibile assegnare

un peso (fisso o casuale) ai due genitori e fare la media pesata invece che quella aritmetica. Quindi, il primo figlio usa peso per il primo genitore e 1- per il secondo, viceversa per il secondo figlio.

9. Algoritmo di mutazione - Come avviene la mutazione degli individui. - Bit Flip: visto nell'esempio, consiste nella modifica di un singolo gene binario. - Random Resetting: cambio casuale di un gene a un altro valore ammissibile. La distribuzione da cui campionare il valore è a scelta: uniforme/normale/ecc. - Swap: scambio di due geni, scelti casualmente. - Scramble: si sceglie un subset di geni in modo casuale e lo si permuta casualmente. - Inversion: si sceglie un subset di geni in modo casuale e lo si ribalta. In altri termini, in termini di mutazione abbiamo molta flessibilità. Qualsiasi variazione va bene in linea di principio, ma è bene che la mutazione sia ammissibile (i.e., da un individuo ammissibile si passi a un altro ammissibile, ovvero che la mutazione non vada a violare i vincoli del problema). Alcuni algoritmi di mutazione lavorano localmente, ovvero su singoli geni dell'individuo, altri a livello globale. Chiaramente, gli algoritmi globali creano maggiore diversità, ma aumentano il rischio di mancata convergenza. Tipicamente, gli algoritmi locali sono preferibili proprio perché limitano l'effetto di casualità che la mutazione può avere sulla convergenza dell'algoritmo.

10. Stopping Condition - Con quale criterio decidiamo di terminare l'evoluzione oppure proseguire. - Tempo di esecuzione: l'evoluzione termina se si è superato un tempo di esecuzione T. Allo stop o si decide di restituire l'ultima generazione ottenuta o la migliore ultima (preferibile). - Costo: se è presente una funzione di costo (i.e., una funzione—diversa dalla fitness—che associa ad individuo un costo secondo un certo criterio), allora al raggiungimento o superamento di quel costo l'evoluzione termina. - Numero di iterazioni: si itera per un massimo di X generazioni. - Assenza di miglioramenti: visto nell'esempio, se per Y generazioni consecutive non ci sono miglioramenti (significativi), allora l'evoluzione termina. - Ibride: ovvero, basate su una combinazione delle precedenti. - Problem specific: conoscendo i dettagli del problema, possiamo definire condizioni ad-hoc. Spesso, invece di parlare esplicitamente di stopping condition, si preferisce la locuzione budget di ricerca (search budget), per indicare meglio il concetto di risorse a disposizione (tempo/costo/iterazioni/ecc.). In molti contesti, una stopping condition composta da tempo di esecuzione/costo e assenza di miglioramenti tende ad essere sufficientemente buona.