**MODULO DI INTELLIGENZA ARTIFICIALE**

PROGETTO KEY-KNOWNLEDGE

Sommario

1. **INTRODUZIONE1**
2. **RUOLO DELL’IA1**
3. **DATASET2**
4. **PSEUDOCODICE DELL’ALGORITMO3**
5. **MOTIVAZIONI4**
6. **PEAS4**
7. **LINK A GITHUB5**
8. **INTRODUZIONE**

**Il progetto consiste nello sviluppo di un’applicazione Mobile-Android per giocare a un quiz. Il vincitore sarà il giocatore che risponde al maggior numero di domande, generate da un agente intelligente che ha la particolarità di evolversi e adattarsi all’utente. Le domande sono suddivise in categorie (STORIA, GEOGRAFIA, SCIENZE, ARTE, CULTURA GENERALE). Una singola partita è sempre 1v1.**

1. **RUOLO DELL’IA**

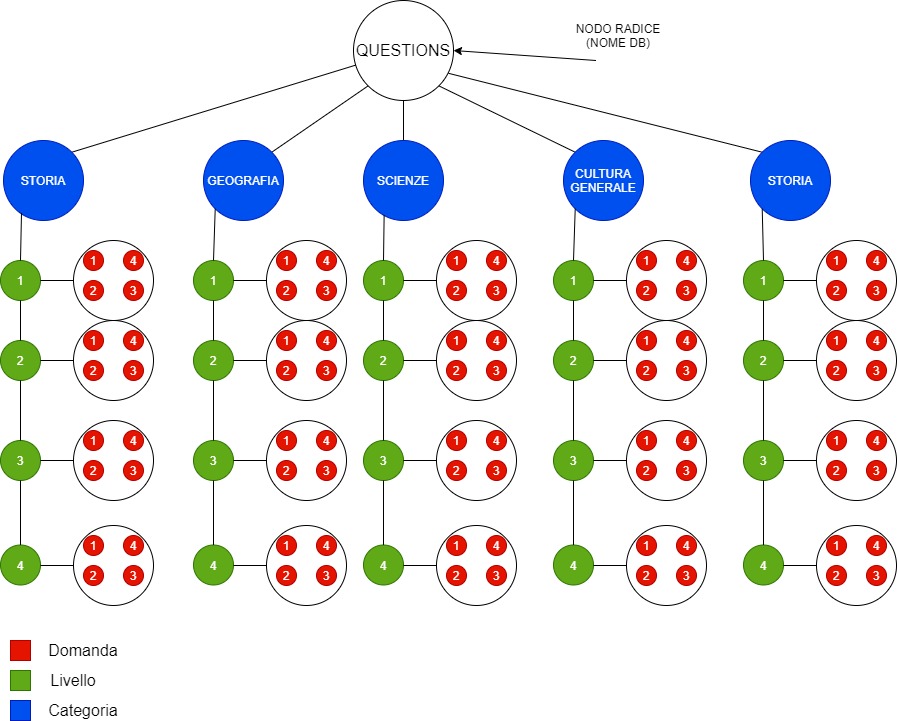
**L’obiettivo dell’intelligenza sarà quello di mettere in difficoltà l’utente proponendogli domande sempre più difficili. Ogni volta che l’utente risponde a una domanda, il suo risultato viene usato dall’intelligenza per capire su quali argomenti è più o meno bravo l’utente.**

**ES: Se un utente risponde in maniera corretta a diverse domande di storia sarà meno probabile che l’intelligenza gli proporrà ancora domande di storia.**

**Algoritmo utilizzato:**

**Algoritmo di ricerca online, in profondità, locale con backtracking e manipolazione di probabilità.**

1. **DATASET**



1. **PSEUDOCODICE DELL’ALGORITMO:**

**Immagine che contiene testo

Descrizione generata automaticamente**

**Immagine che contiene testo

Descrizione generata automaticamente**

1. **MOTIVAZIONI:**

Abbiamo utilizzato questo tipo di algoritmo: “**Ricerca online, in profondità, locale con backtracking”.**

**RICERCA ONLINE: In modo tale che sia** in grado di analizzare la risposta data dall’utente in tempo reale.

**IN PROFONDITA’**: Perché l’obiettivo è mettere in difficoltà l’utente e quindi aumentare la difficoltà delle domande. Per come è strutturato il nostro dataset e per come accediamo ai livelli, il livello massimo di difficoltà si trova nel livello più profondo dell’albero.

**LOCALE**: Perché non ha una soluzione globalmente ottima ma fornisce una soluzione localmente ottima al problema.

**BACKTRACKING: Perché abbiamo la necessità di tornare indietro agli stati precedentemente visitati: al livello di profondità d, ci sono i nodi che costituiscono il livello i di una determinata categoria. Quindi in alcune situazioni, dobbiamo tornare al livello i-1, ad esempio se l’utente sbaglia 2 domande di fila.**

1. **PEAS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tipo di agente | Misura di prestazione | ambiente | Attuatori | Sensori |
|  |  |  |  |  |
| Selezionatore di domande | Domande di una difficoltà che va in base alle conoscenze dell’utente. | Quiz | Schermo del dispositivo Android | Schermo del dispositivo Android |
|  | Domande di categorie diverse, scelte in base a una probabilità. La probabilità cambia in base alle conoscenze dell’utente. |  |
|  |  |  |  |  |

**INDICATORI DI PERFORMANCE**

**Completezza:** L’algoritmo è completo perché non esistono cicli, dato che restituirà sempre una domanda diversa dalle precedenti. Nel nostro caso l’albero di ricerca ha profondità finita, ma anche nel caso in cui avesse una profondità infinita, l’algoritmo terminerebbe. Solo nel caso in cui il quiz contiene un numero di domande maggiore a quelle contenute nel dataset, l’algoritmo non troverebbe una soluzione e non terminerebbe.

**Ottimalità: Non esiste una soluzione ottima globale dell’algoritmo, ma cerca sempre la soluzione migliore corrente, in base agli stati precedenti. L’algoritmo non garantisce di trovare una soluzione migliore rispetto a quella corrente.**

**Complessità temporale:**

**Il nostro algoritmo ha complessità lineare: O(n) con n = numero dei nodi contenuti nell’insieme N dei nodi esplorati. Perché il costo dell’algoritmo dipende dal numero di elementi contenuti in n, che deve esplorare ogni volta per generare un nuovo nodo. Solo nel caso in cui seleziona la prima domanda la complessità è costante: O(1) perché dipende dalla funzione random per selezionare la domanda.**

**Complessità spaziale:**

**In fase di inizializzazione, viene creato:**

* **Un insieme N di nodi esplorati, con N = Ø.**
* **Un insieme C di categorie con k elementi iniziali.**

**Ogni volta che viene chiamato, aggiunge un nodo all’insieme n e aggiunge, nel caso pessimo, k-1 categorie a C. Supponendo che l’algoritmo viene invocato j volte, avremo alla fine che: N contiene j nodi e C contiene k + (j x (k-1)). Quindi la complessità spaziale totale è O(j + k+(j x (k-1)))**

**ROPRIETA’ DELL’AMBIENTE**

* **Completamente osservabile**
* **Agente Singolo**
* **Deterministico**
* **Sequenziale**
* **Statico**
* **Discreto**
* **Noto**

1. **LINK A GITHUB**

<https://github.com/Crex99/UNI-PROJECT/tree/master/FIA>