POKEBRAIN

Ballacchino Giuseppe
Carlino Antonio
Trotta Francesco Giuseppe

Cos'è Pokémon?

Pokémon è un gioco JRPG. Il nome deriva dall'abbreviazione di Pocket Monsters (mostri tascabili). L'obiettivo principale nei giochi è catturare, allenare e far combattere creature chiamate Pokémon.

Cos'è Pokémon?

Ogni Pokémon possiede:

- **Tipo** (uno o due, che influenzano resistenze e debolezze).
- **Statistiche** (PS, Attacco, Difesa, Attacco Speciale, Difesa Speciale, Velocità).
- **Abilità** (effetto passivo che influisce sul combattimento).
- **Mosse** (fino a quattro, di tipo offensivo, difensivo o di supporto).
- **Natura** (modifica leggermente le statistiche).
- **Strumento tenuto** (opzionale, fornisce vantaggi in battaglia).
- **IV ed EV** (valori nascosti che influenzano la crescita delle statistiche).

Qual è lo scopo di PokeBrain

PokeBrain è un agente intelligente progettato per affrontare avversari in Pokémon Showdown utilizzando il modello Minimax. Ogni giocatore ha a disposizione 6 Pokémon; quando uno dei due rimane con tutti i Pokémon sconfitti, ovvero con gli hp pari a 0, perde la lotta.

Cosa abbiamo utilizzato?

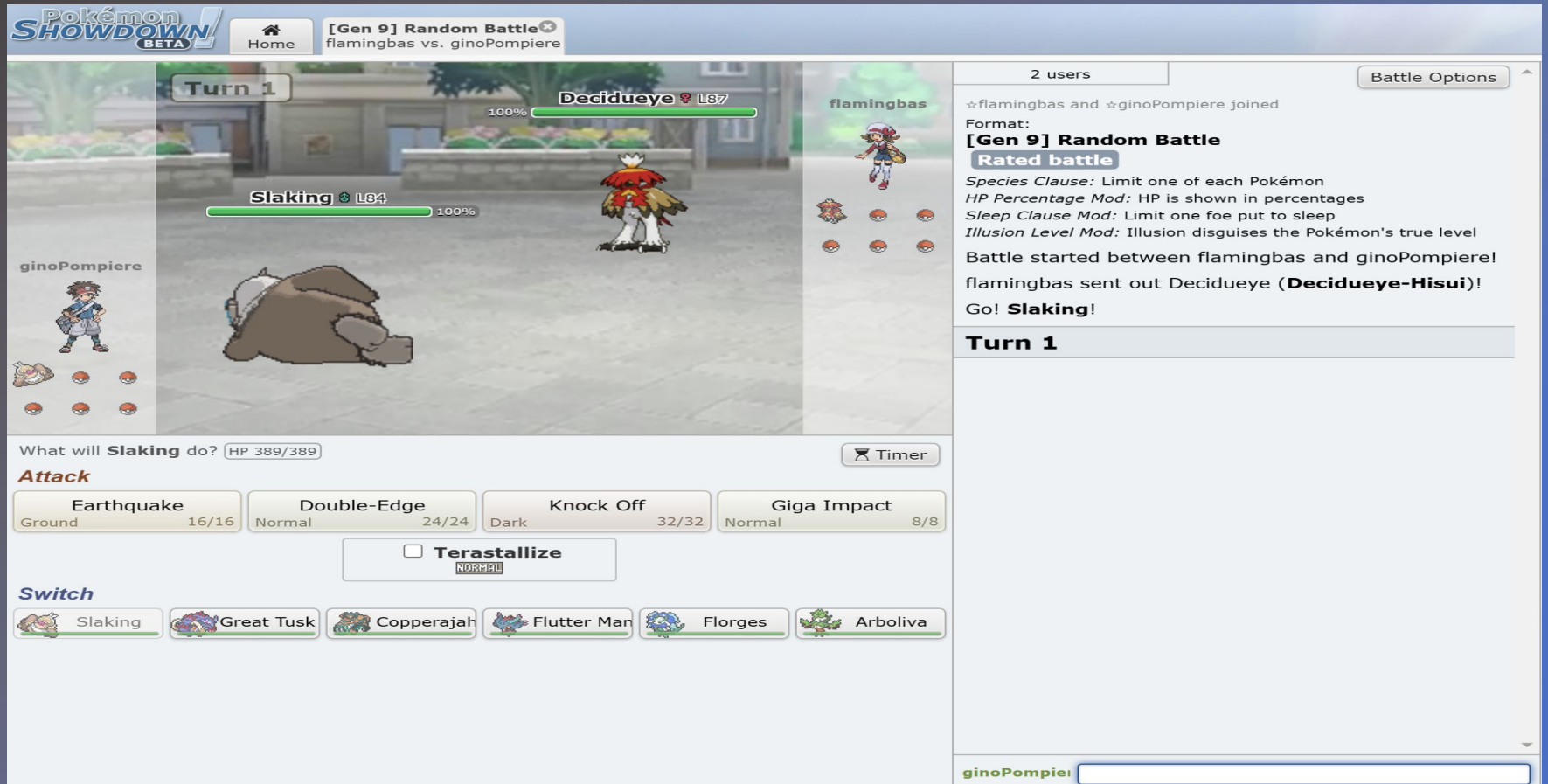
Per lo sviluppo, abbiamo utilizzato Pokémon Showdown, già citato precedentemente, preso dal repository ufficiale di GitHub e integrato con la libreria poke-env. Quest'ultima fornisce un'interfaccia in Python per interagire con il simulatore, consentendo la gestione delle battaglie e l'implementazione di agenti intelligenti.

Cos’è Pokemon Showdown?

Pokémon Showdown è un simulatore di battaglie Pokémon online che permette ai giocatori di sfidarsi senza dover allenare o catturare Pokémon nei giochi ufficiali. È gratuito e accessibile via browser o tramite un'app desktop.

È il punto di riferimento per chi vuole testare strategie e migliorare nel competitivo Pokémon.

Cos'è Pokemon Showdown?



Cos'è Poke-env?

GitHub: https://github.com/hsahovic/poke-env.git

Poke-env è una libreria Python che ha facilitato la gestione e la creazione dell'agente intelligente, nonché l'interazione con Pokémon Showdown. Un ringraziamento a **hsahovic**.

Le classi più utilizzate in poke-env

Le classi utilizzate in poke-env includono:

- **Battle:** Rappresenta una singola battaglia Pokémon, gestendo le dinamiche del combattimento, come l'uso delle mosse, il cambio dei Pokémon e lo stato complessivo della battaglia.
- **Pokemon:** Gestisce un singolo Pokémon in battaglia, contenente informazioni su statistiche, mosse, tipi e altre caratteristiche.
- **Move:** Rappresenta una mossa che un Pokémon può utilizzare, con dettagli come il tipo, la potenza e gli effetti speciali.

Le classi più utilizzate in poke-env

La classe **Player** rappresenta un agente che partecipa ad una battaglia. Essa gestisce la squadra di Pokémon del giocatore, le azioni che può compiere (come selezionare mosse o cambiare Pokémon) e l'interazione con l'ambiente di gioco durante la battaglia.

Le classi più utilizzate in poke-env

La classe **Random Player** è una **sotto-classe** di **Player**.

Un RandomPlayer è un agente che prende decisioni in modo completamente casuale durante una battaglia Pokémon. In pratica, un random player seleziona azioni (come mosse o cambi di Pokémon) senza alcuna strategia predefinita, ma in modo del tutto aleatorio, scegliendo tra le opzioni disponibili.

Il Random Player è stato utilizzato soprattutto per testare l'ambiente di gioco e come punto di riferimento per testare PokeBrain.

Le classi più utilizzate in poke-env

Il metodo **choose_move** della classe **Player** è fondamentale. Viene utilizzato per selezionare quale mossa eseguire durante la battaglia. Per gli agenti, come un Random Player o una IA basata su strategie più complesse viene sovrascritto per personalizzare il comportamento decisionale.

Funzione di **choose_move**:

- **Scopo**: Il metodo **choose_move** serve per determinare quale mossa un Pokémon deve eseguire durante il combattimento, in base alla situazione attuale della battaglia.
- **Restituzione**: Il metodo restituisce una mossa selezionata, che è un oggetto della classe Move, che rappresenta una delle mosse disponibili per il Pokémon in campo.

Come funziona PokeBrain?

Pokebrain, come detto in precedenza, ha una struttura minimax, quindi ad albero. Ogni nodo rappresenta lo stato di gioco.

Ogni nodo (classe NodoMinimax) memorizza:

- L'oggetto battle
- Il riferimento al nodo padre
- Il team avversario
- Il Pokémon attivo avversario
- Un dizionario degli HP dei Pokémon avversari (per mantenere coerenza con la previsione degli stati successivi di gioco)
- Il nostro Pokémon attivo
- Un dizionario degli HP dei nostri Pokémon
- Un valore
- Un'azione
- L'azione prevista
- La lista dei figli

Team Avversario

Potendo gestire entrambi i giocatori, si è riusciti a salvare il team avversario in un array e a passarlo a PokeBrain. Il team viene preso tramite l'oggetto battle.
Successivamente, viene salvato in un array di oggetti pokemon, poi memorizzato nel nodo. Questo ci permetterà di avere un ambiente deterministico, poiché sapremo effettivamente le mosse e i Pokémon dell'avversario. L'unico problema è che opponentTeam viene istanziato completamente a partire dal secondo turno. Di conseguenza, la prima mossa che verrà effettuata sarà di default.

I dizionari degli HP

I dizionari degli HP, sia per i nostri Pokémon che per quelli avversari, vengono utilizzati per tenere traccia degli HP durante le mosse successive, che non riguardano lo stato corrente del gioco. In questo modo, a seconda della mossa collegata al nodo, possiamo determinare i danni subiti dal nostro Pokémon o inflitti all'avversario. Questi dizionari associano a ciascun Pokémon (chiave) i suoi HP (valore).

NodoMinimax.py

All'interno della classe NodoMinimax, oltre alla struttura del nodo, avremo metodi per costruire l'albero. In particolare, saranno presenti metodi per creare i nodi delle mosse di un Pokémon, con 4 nodi per le mosse disponibili e 5 per il cambio del Pokémon. Lo stesso vale per i nodi dell'avversario, con la differenza che in questi ultimi calcoleremo anche i danni inflitti e subiti. Inoltre, aggiorneremo i dizionari degli HP modificando i valori associati a ciascun Pokémon.

Perche' questa differenza tra i metodi?

La differenza tra i metodi dell'avversario e quelli del nostro agente nasce dalla necessità di calcolare la velocità del Pokémon, determinando così chi attacca per primo. Una volta istanziati i nodi del nostro giocatore, istanzieremo quelli dell'avversario, permettendo così di calcolare il danno inflitto e di memorizzare il risultato.
Successivamente, se il Pokémon avversario sopravvive, calcoleremo il danno che subirebbe. Se invece è più veloce, viceversa. In questo modo, possiamo determinare i danni che subiremo noi e quelli che subirà l'avversario

Utils.py

I metodi per calcolare i danni inflitti e subiti, chiamati dai metodi di creazione dei nodi degli avversari, si trovano in Utils. In Utils sono presenti tutti i metodi per il calcolo dei danni, sia fisici che speciali, oltre a un metodo per calcolare gli HP totali del Pokémon, uno per la velocità e uno per determinare l'efficacia della mossa in base alla tipologia dell'avversario.

Pokeminimax.py

PokeMinimax è un gestore che si occupa di creare e gestire l'albero. Ha un metodo per scegliere la mossa con la valutazione più alta, un metodo per valutare il nodo e un metodo per la creazione ricorsiva dei figli.

Pokeminimax.py

Il metodo ScegliMossa verrà attivato dal metodo choose_move. Istanzierà il nodo radice, lo stato corrente del gioco, valuterà se il nostro Pokémon è esausto. Se lo è, chiama il metodo BestSwitch, che creerà i 5 nodi switch, calcolerà la valutazione ed assegnerà il valore più alto al padre. Altrimenti, attiverà il metodo minimax, generando i nodi figli dell'agente per ogni singola mossa possibile, se è il suo turno. In caso contrario, genererà i nodi avversari. Infine, darà una valutazione per ogni singolo nodo ed assegnerà il valore più alto/basso al padre.

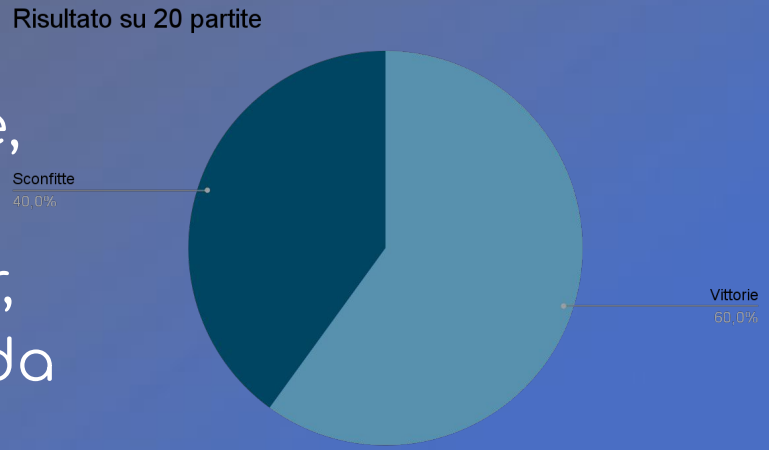
CalcoloValore

La valutazione semplicemente assegnerà un valore. Verrà attribuito un punteggio se abbiamo sconfitto un Pokémon avversario, con un ulteriore incremento se la mossa è super efficace. Allo stesso modo, la valutazione subirà una riduzione se il nostro Pokémon subisce danni, con una penalizzazione ancora maggiore nel caso in cui venga sconfitto.

Conclusioni: Pro

Dopo aver compreso il funzionamento di Pokébrain, è stato testato contro un giocatore Random Battle, modificato in modo da salvare il suo team all'interno di un array.

Il test è stato condotto su 20 partite, delle quali 12 sono state vinte da Pokébrain, ovvero da MinimaxPlayer, mentre le restanti sono state vinte da RandomPlayer.



Conclusioni: Pro

Sappiamo che l'algoritmo Minimax funziona meglio quando l'avversario esegue sempre la mossa ottimale durante il proprio turno, cosa che RandomPlayer non è in grado di fare. Di conseguenza, questo tipo di avversario rappresenta un banco di prova piuttosto svantaggioso per MinimaxPlayer. Abbiamo quindi testato Pokébrain contro un altro MinimaxPlayer, e i risultati hanno dimostrato che è in grado di ragionare meglio. Si è osservato che ha effettuato molte più mosse super efficaci.

Conclusioni: Contro

Problemi di Pokébrain

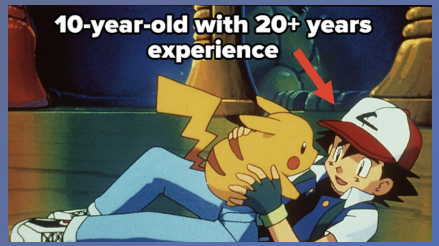
Pokébrain presenta due problemi principali:

1. La **prima mossa** viene effettuata di default perché il team avversario non viene inizializzato correttamente.
2. La **seconda mossa** può anch'essa essere di default, poiché a volte il team avversario non è ancora pronto.

Inoltre, si verifica occasionalmente un loop di mosse, si verifica soprattutto quando ha contro un RandomPlayer

Miglioramenti

Per migliorare l'efficienza, si potrebbe implementare una valutazione più accurata nell'albero delle mosse, integrando il Reinforcement Learning per ottimizzare le decisioni dell'agente. Inoltre, si potrebbe affinare la logica delle mosse, rendendo la strategia più complessa e dinamica, permettendo una valutazione più precisa delle diverse situazioni di gioco. Infine, sarebbe utile sviluppare una potatura dell'albero per ridurre la complessità computazionale.



Fine

GRAZIE PER L'ATTENZIONE!