Dossier – Intelligence Artificielle

& Framework Django

Projet de fin de quadrimestre

BAERT Antoine

HERRENT Antoine

SERVAIS Jordan



IESN, Hénallux Namur Belgique

CHARLIER Christine

SMAL Anne

Vendredi, 18 décembre 2020

Table des matières

[I. Introduction à notre projet 3](#_Toc59182372)

[A. Explication de notre jeu 3](#_Toc59182373)

[II. Notre base de données 4](#_Toc59182374)

[A. Schéma de la base de données 4](#_Toc59182375)

[B. Explications des tables 4](#_Toc59182376)

[1. La table User 4](#_Toc59182377)

[2. La table UserGame 5](#_Toc59182378)

[3. La table Game 5](#_Toc59182379)

[4. La table AI 5](#_Toc59182380)

[5. La table State 6](#_Toc59182381)

[6. La table Move 6](#_Toc59182382)

[7. La table Esperance 6](#_Toc59182383)

[III. La découpe de notre projet 6](#_Toc59182384)

[A. Framework Django 6](#_Toc59182385)

[B. Schéma explicatif de l’application Game 7](#_Toc59182386)

[IV. Notre Intelligence Artificielle 8](#_Toc59182387)

[A. Explications préalables 8](#_Toc59182388)

[B. Choix de la fonction de Reinforcement Learning 8](#_Toc59182389)

[C. Explication de notre intelligence artificielle 9](#_Toc59182390)

[1. Explications générales 9](#_Toc59182391)

[2. Les fonctions de l’intelligence artificielle 10](#_Toc59182392)

[3. Comment jouer contre une IA ? 11](#_Toc59182393)

[4. L’entrainement de notre IA 11](#_Toc59182394)

[V. Conclusion 12](#_Toc59182395)

# Introduction à notre projet

Comment fonctionne le Framework Django et le Reinforcement Learning ?

Notre projet a pour but de comprendre et de coder un jeu ainsi qu’une Intelligence Artificielle capable de jouer à ce dernier.

Notre projet est composé de plusieurs parties, la première étant l’implémentation du Framework Django. Nous avons utilisé plusieurs langages de programmation pour se faire, la majeure partie du back-end est réalisé en python tandis que le front-end est codé en HTML, CSS et Javascript. La méthode utilisée pour la découpe du projet et sa structure est la découpe MVC. Cette partie est relation directe avec notre base de données PostgreSQL.

La seconde est la création d’une intelligence artificielle à l’aide d’une méthode de Reinforcement Learning, la Q-Function.

## Explication de notre jeu

Notre jeu est relativement simple, le but est de prendre un maximum de case en se déplaçant sur une grille de jeu.

La grille est composée de 64 cellules, ou les deux joueurs commencent aux extrémités.

Pour pouvoir jouer au jeu, il vous suffit de vous connecter à notre plateforme à l’aide de votre Username et votre Password. Si vous n’avez pas de compte, vous pouvez vous enregistrer à l’aide bu bouton Register.

Une fois connecté vous avez la possibilité d’ouvrir le menu se situant dans le coin gauche de la page. Dans ce menu se trouve cinq options :

* **Create** : Vous permet de créer une partie et de choisir votre couleur pour cette partie.
* **Join** : Vous permet de rejoindre une partie tout en choisissant votre couleur.
* **Resume :** Vous permet d’afficher la liste de vos parties en cours et de continuer ces dernières.
* **Create AI :** Vous permet de créer une intelligence artificielle en choisissant les paramètres gama, learning\_rate et l’epsilon\_greedy. Celle-ci s’entrainera sur le serveur après sa création.
* **List AI :** Vous permet d’afficher la liste des intelligences artificielle disponible et contre lesquels vous pouvez créer une partie ou les entrainer.

Une fois une partie lancée, vous arrivez directement sur la grille de jeu. Un court descriptif de la partie est affiché (les deux joueurs, leurs couleurs ainsi que le tour).

Sous la grille vous pouvez trouver les quatre boutons permettant d’avancer (UP, DOWN, LEFT et RIGHT).

Pour quitter le jeu, un bouton Logout se trouve dans le coin supérieur droit. Celui est retrouvable dans le menu « resume ».

# Notre base de données

Premièrement, nous avons utilisé une base de données PostgreSQL. Cela nous a permis d’enregistrer le gameState (l’état de la partie, la grille de jeu) directement sous forme de tableau.

## Schéma de la base de données

Diagram

Description automatically generated

## Explications des tables

### La table User

Cette table permet d’enregistrer les informations relatives à un user, on retient son « Username » et un identifiant « Id ». Nous utilisons cette table lors de la connexion à notre jeu.

Dans notre Model de « connection », nous n’avons pas définit le model User car nous utilisons la classe prédéfinie par Django « AbstractUser » disponible directement dans Django via django.contrib.auth.models, cette dernière est prévue pour gérer les utilisateurs.

### La table UserGame

La table UserGame est l’intersection entre un User et une Game. Elle permet d’enregistrer un User x dans une Game y. Pour chaque User dans une Game nous retenons :

* **Color** = IntegerField, permet de retenir la couleur sélectionnée par le joueur.
* **PosX** = IntegerField, la position x du joueur.
* **PosY** = IntegerField, la position y du joueur.
* **UserNumber** = IntegerField, le numéro du joueur dans la Game (1 ou 0).

Cette table possèdera quatre clés étrangères :

* **UserId** = lien vers un User (un joueur).
* **Game** = lien vers une Game (la partie).
* **Ia** = lien vers une intelligence artificielle.
* **MovePrecedent** = lien vers une espérance, cela nous permettra à retrouver l’espérance du mouvement précédent.

### La table Game

Cette table permet d’enregistrer une partie. Chaque partie sera définie par :

* **Id** = IntegerField, l’identifiant d’une partie
* **GameState** = ArrayField(ArrayField()), un tableau de tableau enregistrant l’état de la partie.
* **CurrentUser** = IntegerField, le joueur qui est ou était en train de jouer. (Le tour)
* **Winner** = IntegerField, le numéro du joueur qui a gagné, ou pas si la partie est encore en cours.

Dans le model de cette dernière, nous enregistrons deux champs supplémentaires :

* **Players** = lien vers User en passant par UserGame, cela nous permet d’enregistrer l’historique des parties de l’utilisateur.
* **Ias** = lien vers AI en passant par UserGames, cela nous permet d’enregistrer l’historique des parties de l’intelligence Artificielle.

### La table AI

La table AI nous permet d’enregistrer les caractéristiques notre intelligence artificielle :

* **Epsilon\_greedy** = FloatField compris entre 0 et 1, taux entre les explorations et exploitations
* **Learning\_rate** = FloatField compris entre 0 et 1, la prise en compte des récompenses plus ou moins lointaine dans le futur.
* **Gama** = FloatField compris entre 0 et 1, la prise en compte des résultats des parties plus ou moins lointaine dans le futur.

Cette table possède une clé étrangère, cette dernière reprend tous les états possibles de la partie, ce champ est comparable à la Q-Table de notre intelligence artificielle :

* **States** = lien vers la table State.

### La table State

Cette table State, nous permet d’enregistrer un état, un état est caractérisé par :

* **posXUser1** = IntegerField, position X du joueur 1.
* **posYUser1** = IntegerField, position Y du joueur 1.
* **posXUser2** = IntegerField, position X du joueur 2.
* **posYUser2** = IntegerField, position Y du joueur 2.
* **Turn** = IntegerField (1 ou 2), le numéro du joueur qui doit jouer / IA.
* **GameState** = ArrayField(ArrayField()), un tableau de tableau enregistrant l’état de la partie.

### La table Move

La table Move nous permet d’enregistrer un mouvement. Un mouvement est identifié par :

* **moveY** = IntegerField (0,1 ou -1), le mouvement sur Y
* **moveX** = IntegerField (0,1 ou -1), le mouvement sur X

### La table Esperance

Cette table permet d’enregistrer une espérance. Cette dernière est définie par un mouvement « Move » et un état « State ».

* **Esperance** = FloatField (entre 0 et 1), l’espérance associé au mouvement et à l’état.

La table Esperance possède donc deux clés étrangères :

* **State** = lien vers un état.
* **Move** = lien vers un mouvement.

# La découpe de notre projet

## Framework Django

Notre projet utilise le Framework Django. Ce dernier nous a permis de structurer notre projet avec trois applications différentes.

Chacune de nos applications possède :

* Un Model, lien avec la base de données.
* Un fichier Views, permet de gérer et faire le lien entre le model et les Templates.
* Un dossier Templates, contenant tous nos fichiers codés en HTML, CCS et JS. Ces derniers permettent l’affichage des différentes pages web.
* Dans notre application IA, nous avons ajouté un fichier Business, ce dernier permet de gérer la logique interne à l’intelligence artificielle.
* Notre application Game est plus complexe, voir schéma suivant…

Nos trois applications Django :

* Connection, dans cette dernière nous avons géré la connexion d’un utilisateur à notre jeu.
* Game, ici nous avons rassemblé tout le code directement lié à une partie.
* AI, dans cette application, nous gérons le code de l’intelligence artificielle.

## Schéma explicatif de l’application Game

Ce schéma permet de comprendre le flux des données à travers nos différentes couches pour l’application Django « Game » :

Diagram

Description automatically generated

Nous avons donc ajouté trois fichiers supplémentaires à la structure classique de Django, le mapper permet de faire le lien entre les models et le DTO. Le DTO nous permet de créer les classes orienté objet utilisé dans le business et pour finir, le business utilisons pour faire toute la logique principale du jeu

# Notre Intelligence Artificielle

## Explications préalables

**L’environnement** = le plateau de jeu (constitué de 64 cases)

**L’agent** = l’IA, elle se déplace sur le tableau de jeu

**L’état** = Les positions des deux joueurs, l’état de la partie au moment t et le tour (A qui est-ce de jouer)

**L’action** = le mouvement que l’agent va effectuer

**La Policy** **𝜋** = ce qui est préconisé de faire à chaque état

**Les récompense** = gain potentiel en fonction de l’action

**Le Gama γ** = [0 ;1] Prise en compte des autres parties plus ou moins lointaine dans le futur

**Le Learning Rate** = [0 ;1] La prise en compte des récompenses plus ou moins lointaine dans le futur.

**L’ε-Greedy** = [0 ;1] Taux entre les explorations et les exploitations

**L’exploration** = tester de nouvelles actions et apprendre l’environnement

**L’exploitation** = refaire des actions ou la récompense est certaine

## Choix de la fonction de Reinforcement Learning

* “V Function, measures the goodness of each state **s,** how good or bad it is to be in this particular state following a policy**𝜋**: **V𝜋(𝑠) »**
* “Q Function, the value of taking action **a** in state **s** under a policy **π:** **Q𝜋(𝑠, 𝑎) »**

La V Function ne prend donc en compte seul l’état dans lequel se trouve l’agent. Une fois que ce dernier a fait son action, l’algorithme va regarder la récompense du nouvel état et mettre à jour son backtracking. En créant ce backtracking, l’agent va apprendre les actions à faire et ne pas faire en fonction de l’état s.

La Q Function elle, prend en paramètre l’état actuel s et l’action future a. Elle va donc regarder autour d’elle les mouvements possible et lequel va être le plus bénéfique avant de changer d’état. Au fur et à mesure de l’exploration la fonction va créer une Q table qui retiendra pour chaque état toutes les récompenses possibles en fonction de l’action a.

Pour notre projet, nous allons utiliser la Q Function, celle-ci est plus adapté à notre jeu qui est petit (maximum 4 actions possible par mouvement). Il est donc facile d’appliquer la Q Function à ces 4 actions et déterminer la meilleure.

Si nous prenions la V Function, l’apprentissage serait plus lent.

## Explication de notre intelligence artificielle

### Explications générales

Comme dit précédemment, nous allons utiliser la Q-Function pour notre projet. Notre agent évoluera donc dans un environnement de 64 cases.

Chaque ligne représente un état de notre jeu (Un état = les positions des deux joueurs, l’état de la partie au moment t et le tour), chaque état est composé de 4 reward possible **r**, chacun de ces rewards est associé à un mouvement (UP, DOWN, LEFT ou RIGHT).

Au final, nous obtenons une q-table :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UP | DOWN | LEFT | RIGHT |  |
| esperance | esperance | esperance | esperance | STATE |
| esperance | esperance | esperance | esperance | STATE |
| esperance | esperance | esperance | esperance | STATE |

Concernant le reward **r** à l’instant **t**, il est égal au nombre de cases prises par l’IA – le nombre de case prise par l’adversaire. Ce reward évoluera au fur et mesure de l’état de la partie.

Le reward le plus élevé, par state, est égal à la meilleure action.

Diagram

Description automatically generatedDans notre projet, tout se fait grâce à des appels à la base données. Par exemple, si nous avons besoin d’une esperance ou d’un mouvement, nous allons les rechercher dans la base de données.

Voici un schéma décrivant les différentes étapes du fonctionnement de notre intelligence artificielle :

Pour mieux comprendre le stockage de notre IA dans notre base de données : voir le schéma de BD.

### Les fonctions de l’intelligence artificielle

La logique de notre intelligence artificielle est composée de 3 fonctions :

***def reward(game\_state)***

Retourne une récompense en fonction de l’état global de la partie. Le paramètre **game\_sate,** est un tableau de tableau, ils reprennent la grille complète du jeu. Une case = 0, 1 ou 2 en fonction du joueur qui possède la case ou 0 si aucun.

***def take\_action(epsilon, state, possible\_moves)***

Cette fonction permet de faire prendre à l’IA de manière random une décision, soi explorer soi faire l’action qui lui rapporte le plus de récompense.

* + - * **Epsilon** = l’epsilon greedy, détermine si exploration ou exploitation.
      * **State** = L’état au moment t.
      * **Possible\_moves** = tous les mouvements que l’IA pourrait faire sachant qu’elle se trouve sur le state x.

***def play(PosXUser1, PosYUser1, PosXUser2, PosYUser2, Game\_state, UserGame, Possible\_moves, turn)***

La fonction play, est la fonction principale, elle implémente les deux précédentes. Elle permet de faire prendre une action à l’agent et de modifier / remplir la q-table à l’aide du mouvement précédent et du reward.

* + - * **PosXUser1** = La position X du joueur 1
      * **PosYUser1** = La position Y du joueur 1
      * **PosXUser2** = La position X du joueur 2
      * **PosYUser2** = La position Y du joueur 2
      * **Game\_state** = L’état global de la partie, un tableau de tableau, ou une case est soit 1, 2 ou 0.
      * **UserGame** = un userGame, nous permet de retrouver l’espérance de l’état et du mouvement précédent.
      * **Possible\_moves** = Un tableau reprenant les mouvements possibles, que l’IA pourrait réaliser.
      * **Turn =** Le tour, le numéro du joueur a qui est le tour.

Notre formule de remplissage de la Q-Tbale (la table esperance dans notre base de données) :

***prevEsp.esperance = prevEsp.esperance + userGame.ia.learning\_rate \* (action\_reward + userGame.ia.gamma \* best\_current\_esperance - prevEsp.esperance)***

Nous récupérons, le Learning rate et le gamma grâce à un lien direct avec notre base de données.

Nous avons deux autres fonctions qui permettent la modification des paramètres de l’intelligence artificielle et l’affichage de ces dernières.

***def create\_ia(form)***

Cette fonction permet de récupérer les paramètres de l’intelligence

artificielle via un formulaire et de changer l’IA avec ces derniers.

* + - * **form** = un formulaire, ce formulaire est composé de trois champs, epsilon greedy, learning rate et le gamma.

***def list\_ia() :***

Cette fonction permet de récupérer la liste des différentes intelligences

artificielle. Cela nous permettre par la suite de les afficher.

### Comment jouer contre une IA ?

Pour jouer conte une intelligence artificielle, il vous suffit d’afficher la liste des IA disponible et de cliquer sur « Play ».

Plusieurs IA sont disponibles, vous pouvez choisir celle qui vous convient le mieux en fonction de ses paramètres.

Attention aux différents niveaux de difficulté, plus l’Epsilon Greedy sera élevé plus l’intelligence sera compliquée à battre !

Si vous souhaitez créer votre propre intelligence, avec ses propres paramètres, vous pouvez le faire à l’aide de la section « Create IA » dans le menu. Une fois sur cette page, vous pouvez insérer les paramètres de l’IA (Epsilon Greedy, Learning Rate et Gamma). Lorsqu’elle est créée, elle se trouvera dans la liste des IA, ou vous pourrez jouer contre elle.

### L’entrainement de notre IA

L’entrainement de notre intelligence artificielle est relativement simple. Nous allons faire jouer l’IA contre elle-même.

Pour entrainer une intelligence artificielle, nous avons créé un bouton directement dans le menu « List IA ». Sur cette page, vous avez accès à la liste des IA pouvez entrainer l’IA que vous souhaitez.

Attention, nous vous déconseillons d’entrainer une IA plus de quelques parties, car ce processus peut prendre relativement beaucoup de temps.

# Conclusion

Notre projet nous a permis de découvrir et d’implémenter plusieurs concepts, le premier étant le Framework Django, le second une fonction de Reinforcement Learning (la Q-Function) et pour finir le langage Python. D’autres outils secondaires sont venus s’ajouter comme PostgreSQL, docker et Git.

**Vous pouvez retrouver notre projet sur github :** <https://github.com/TurboMachina/ProjetDjangoIA_2020>