Compte Rendu

TP Gamification adaptative

Khalid CHBAB - M2 IA - p2112712

(khalid.chbab@etu.univ-lyon1.fr)

Première partie : Recommandations à partir de profils

Étape 1.

Le dossier hexad contient 12 fichiers correspondant aux coefficients et valeurs p pour chaque élément du jeu. Du même pour dossier de motivation ou on a les p-values et coefficients pour chaque élément de jeu.

Pour les matrices HEXAD, chaque colonne représente la variation de chaque type de motivation (motivation intrinsèque, motivation extrinsèque, amotivation) par rapport au type d'élève. Pour les matrices Motivation, les colonnes représentent la motivation initiale et les lignes la variation de chaque type de motivation.

Prenant l'exemple des Matrice des joueurs qui ont eu l'élément 'Score' ***(scorePathCoef.csv et scorepVals.csv)***. Chaque case représente la variation de chaque motivation pour ce type de joueur ou une valeur proche de zéro indique une faible variation. Une valeur positive indique une augmentation de la motivation en question, et une valeur négative indique une diminution de cette même motivation. Par exemple pour la motivation intrinsèque on constate que cette motivation à augmenter pour les joueurs de type player, achiever et disruptor en ordre décroissant par contre elle diminue pour les joueurs de type socialiser et pour les autres reste la même. Mais ces valeurs n'ont aucune signification si sont prises en compte indépendamment de la matrice de p-values. Chaque case de cette matrice représente la fiabilité de cette mesure de motivation, les valeurs proches de 0 ont plus de fiabilité est si on dépasse la valeur de **0.05 ou 0.1** la mesure n'a pas d'importance dans notre cas.

Par exemple dans ***scorepVals*** la variation de motivation intrinsèque est de +48% pour les joueurs de type player avec une fiabilité importante (pval = 0.001587734642307). En revanche, la variation de motivation extrinsèque pour les type de joueur philanthropist est de -49% avec fiabilité importante de (pval = 0.0369750483702073)

"On précise que on vas travailler avec 0.1 puisque avec 0.05 on perte majorité d'information"

Etape 2 : Recommandations d'éléments ludiques

La première étape de mon algorithme de recommandation d'éléments ludiques est d'éliminer/mettre à 0 les valeurs ou le p-val est plus p-val = 0.1, je constate que vous pouvez modifier cette valeur en modifiant le variable PVAL dans le code.
Pour calculer les motivations initiales j'ai pris la somme des composantes caractéristiques de chacune des motivations

Motivation Intrinsèque Initiale = micoI + miacI + mistI

J'ai également considéré que les 3 motivations ont d'importance équivalente. Donc dans la fonction calcule() j'ai sommé la motivation intrinsèque et extrinsèque en déduisant l'amotivation.

On répète ces procédures de calcul pour chaque élément de jeu.

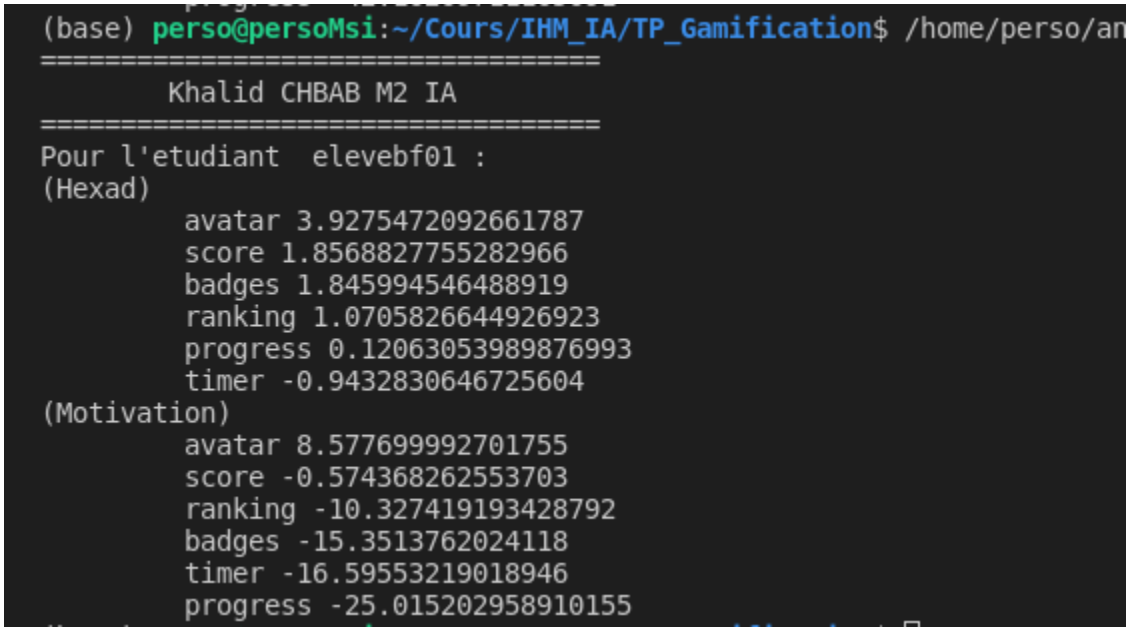

Figure 3 : Affichage des recommandations pour l'élèvebf01 selon son profil HEXAD et Motivation

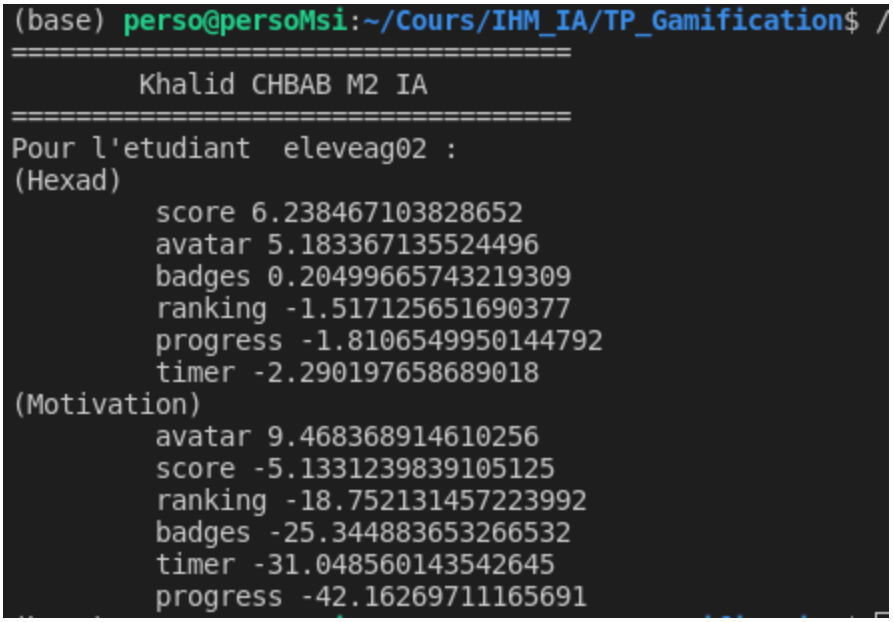

Figure 3 : Affichage des recommandations pour l'élèveag02 selon son profil HEXAD et Motivation

Vous trouverez le code dans fichier calculate_affinity.py

Deuxième partie : Algorithme d'adaptation

Étape 1

Après l'exécution de code plusieur fois, j'ai constater l'existence de 4 cas différents :

- Cas 1 : les vecteurs d'affinity sont les mêmes, donc nous recommander le premier élément.
- Cas 2 : le premier élément des deux vecteur est le même, donc nous recommander le premier élément.
- Cas 3 : toutes les valeurs sont négatives ou 0, dans ce cas on normalise nos données et on calcule le moyen de chaque élément, et on recommande le meilleur élément avec la valeur maximum. Si il y'a d'égalité en fait le choix aléatoirement.
- Cas 4 : les vecteurs sont totalement différents, ou mélangés aléatoirement, pareil au cas 3 on fait la même chose.

J'ai pris la décision de normaliser les valeurs à l'intérieur des vecteurs puisque j'ai constaté que le vecteur de motivation contenait des valeurs positives très élevées en comparaison au valeur du vecteur du profil Hexad. Après cela, on calcule la moyenne pour chaque élément et on recommande l'élément premier.

Étape 2.

```
Etant donnée : l'etudiant etud
Faire
        Vecteur_hexad ← recommandation_hexad(etud)
        Vecteur_mot ← recommandation_mot(etud)
        Si Vecteur_hexad == Vecteur_mot
                Return Vecteur_hexad[0]
        Sinon si Vecteur_hexad[0] == Vecteur_mot[0]
                Return Vecteur_hexad[0]
        Fin si
        Vecteur_hexad ← normalize(Vecteur_hexad)
        Vecteur_mot ← normalize(Vecteur_mot)
        max = -inf
        recommande = None
        Pour chaque v contenu dans Vecteur_hexad
                tmp ← AVG(Vecteur_hexad[v], Vecteur_mot[v])
                Si tmp > max
                        recommande = v
                        Max = tmp
                Fin si
        Fin pour
        Return recommande
Fin
```

Étape 3.

Mon code est disponible dans le fichier algorithm.py. Il s'occupe d'appeler le fichier de la première partie afin de générer les vecteurs d'affinités et de déterminer quel est l'élément de jeu le plus intéressant pour l'étudiant. L'étudiant et la précision sont des variables globales fixées à l'intérieur du fichier.

Étape 4.

Le code d'évaluation est disponible dans le fichier eval.py. Pour chaque élève, on a calculé leurs recommandations adaptées en fonction de leurs profils. Puis on a calculé quelques mesures statistiques utilisant les built-in functions de Pandas.

```
(base) perso@persoMsi:~/Cours/IHM_IA/TP_Gamification$ /home/perso/anaconda3/bin/python /home/perso/Cours/IHM_IA/TP_Gamification/eval.py
==================================
        Khalid CHBAB M2 IA
==================================
Nombre des etudiant :  258
Nombre de recommendations correspondantes 46/258
Nombre de recommendations différente 212/258
avatar    252
score       6
Name: Recomendation, dtype: int64
==================================
          Info
==================================
              Time  CorrectCount  FullyCompletedLessonCount       MiVar       MeVar     amotVar
count    258.000000    258.000000                 258.000000  258.000000  258.000000  258.000000
mean    5491.542636     69.554264                   1.224806   -2.375969   -1.449612    2.833333
std     2130.147131     31.628789                   1.574309    6.980387    6.997455    3.281734
min      907.000000      7.000000                   0.000000  -23.000000  -23.000000   -7.000000
25%     4179.000000     48.000000                   0.000000   -7.000000   -6.000000    1.000000
50%     5073.500000     66.000000                   1.000000   -3.000000   -1.000000    3.000000
75%     6716.750000     85.000000                   2.000000    2.000000    3.000000    4.000000
max    12439.000000    211.000000                   8.000000   31.000000   22.000000   16.000000
(base) perso@persoMsi:~/Cours/IHM_IA/TP_Gamification$
```

On constate que 46 des élèves ont des recommandations similaires à notre algo et 212 élèves qui ont des résultats différents. On constaté sur l'ensemble des élèves, notre algorithme de recommandation sélectionne fréquemment l'élément avatar (252 fois sur 258 étudiants). On constate aussi que le moyen de variation des motivation en général est positif.

Pour conclure, je peux dire que mon algorithme ne semble pas fonctionnel. Et que la recommandation selon un équilibre entre les profil HEXAD et Motivation me semble pas la bonne méthode.