Benjamin Ben Arous
Piraveen Kandiah
Alexis Fauxbâton

2018/2019

ÉTUDE DE L'ÉVOLUTION D'UNE COMMUNAUTÉ DE JOUEURS EN FONCTION DE LEUR OPINION

ARE Dynamic

Sorbonne Université

Étude de l'évolution d'une communauté de joueurs en fonction de leur opinion

Le projet d'ARE que nous avons décidé d'entreprendre consiste à étudier l'évolution de l'opinion ainsi que des rapports d'influences entre les différents joueurs d'une communauté. L'objectif de cette étude est d'analyser le comportement des différents joueurs (Gamers, Joueurs lambda, Influenceur) selon leur opinion (Très content, Content, Moyen content, Pas content) concernant un jeu donné. Il s'agit ici du jeu Fortnite.
Pour ce faire nous avons décidé d'utiliser la bibliothèque python NetworkX, afin de représenter au mieux nos programmes d'évolutions de nos joueurs.
Nous avons obtenu des graphiques qui reflètent l'évolution de l'opinion de notre communauté ainsi que des rapports d'influences en présence d'individus comme des influenceurs au cours du temps et en fonction des différents paramètres.

The ARE Project that we have been working on during this semester is about the evolution of the opinion of some players in a community. We also added some influence links between some players. The objective of this simulation is to see the impact of the player's type (Gamers, Normal Player, Influencer) in relation to their opinion (Satisfied, Very Satisfied, Moderately Satisfied, Not Satisfied) on a given game.
In order to do this, we decided to use the NetworkX python library to model our community of players over time. We have obtained some graphs and we will show you some results that we criticized. These graphs represent pretty accurately the evolution of the opinion of our community

I°) INTRODUCTION :

Au cours de ce projet nous étudierons l'évolution de l'opinion de notre communauté de joueurs. Pour ce faire nous modéliserons de manière chronologique cette évolution de manière codépendante à l'opinion qu'ont les utilisateurs de la société-mère en question. Par exemple le jeu APEX Legends à pour société mère EA qui souffre ces derniers temps d'une baisse nette en terme de réputation. Ceci a poussé, Respawn, le développeur du jeu, à s'éloigner de la dîte société en guise de stratégie marketing.

Notre équipe de développement était constitué de 3 personnes : Piraveen Kandiah, Benjamin Ben Arous, Alexis Fauxbaton.

Afin d'obtenir un résultat pour notre projet plusieurs solutions se présentaient à nous. Au départ nous voulions présenter un plan quadrillé sur lequel seraient présents 3 cercles représentant 3 jeux. Et chaque cercles seraient constitués de points représentants les joueurs. Nous avions pour objectif de réaliser un modèle comme cela. Cependant, nous nous sommes rendu compte que ce à quoi nous voulions aboutir était ambitieux par rapport à notre niveau actuel. C'est pourquoi nous avons finalement opté pour une modélisation par des graphiques en utilisant NetworkX.
Nous verrons dans un premier temps, les paramètres fondamentaux de notre modèle ainsi que le modèle à l'instant initial. Enfin, nous étudierons les résultats de notre modèle lorsque que nous appliquons les différents paramètres étudiées telles que l'opinion.

II°) PRÉSENTATION DE LA THÉMATIQUE :

Le projet que nous avons décidé de mener repose principalement sur des probabilités que nous avons imaginé en nous appuyant sur des données réelles, au cours des séances. En effet, nous disposons d'une communauté de joueurs. Il y a différents types de joueurs que nous pouvons retrouver parmi eux. Il y a tout d'abord les influenceurs, ce sont des joueurs qui ne sont pas forcément des gamers confirmé. Cependant, ils sont présents sur un certain jeu et leur but est d'exercer une influence sur les autres joueurs pour faire en sorte que l'opinion sur le jeu s'améliore ou pas. Ensuite, il y a aussi les joueurs lambda, qui sont représentés commes des individus normaux qui peuvent avoir une opinion bonne ou mauvaise sur le jeu. Enfin, il y a les gamers confirmé, qui sont des joueurs réguliers sur chaque jeux.

Les notions fondamentales que nous avons établi pour l'élaboration de notre projet sont les différentes probabilités de variations d'opinion des joueurs, ou encore les probabilités que des bad buzz ou des good buzz surviennent sur la communauté de joueurs. Nous avons également pensé à des paramètres concernant la probabilité qu'une personne soit de plus en plus content ou de moins en moins content.

III°) CONTRIBUTION EN TERME DE DÉVELOPPEMENT

Au cours de ce cycle d'ARE, nous avons réalisé divers programmes qui avaient pour but de construire une communauté de joueurs différents. Nous avons créée sur le Notebook Jupyter des dictionnaires contenant en clé les différents types de joueurs que nous voulions représenté dans notre communauté. Il s'agit des Gamers confirmé, les joueurs lambdas, ainsi que les influenceurs.

Benjamin : (Opinion utilisateur, Avis, Variations)

Je me suis tout d'abord chargé des programmes initiaux. J'ai programmé la mise en place de l'opinion d'un seul utilisateur en m'appuyant sur des opinions les plus réalistes possibles en affichant un résultat compris entre 0 et 100, avec, en fonction du résultat obtenu en sortie, l'opinion de l'utilisateur : 'Pas content'; 0<x<24, 'Moyen content'; 25<x<49, 'Content'; 50<x<74, 'Très content'; 75<x<100 .
Dans un second temps, je me suis occupé également de l'avis d'un groupe; c'est à dire pour un nombre donné de personne, j'ai généré un avis indépendant pour chacun avec un dictionnaire permettant à la fois de lui donner son chiffre correspondant (avec des clés numérotés de 1 à n) et l'avis de chacun. En clair, ce programme permet de générer l'avis personnel initial d'un nombre donné de personnes. Il s'agit du point d'appui des graphiques et des modélisations, toutes les modifications d'opinion qui auront lieu au cours du temps s'appuient sur ces opinions initiales.
J'ai également réalisé la variation de l'opinion; il s'agit d'un programme 'support' permettant de simplifier la mise en place de l'opinion suivante de la population.
Tout au long de ce cycle d'ARE, nous étions tout d'abord parti sur une autre piste qui n'était pas la bonne, ainsi de nombreux programmes que j'avais réalisé précédemment, n'ont finalement pas été utilisés pour notre projet car trop complexes à adapter sur les graphiques ou tout simplement inadaptés. Ainsi, même les principaux programmes ont dû être adaptés.

Piraveen : (Opinion suivante, paramètres initiaux, types de joueurs NetworkX)

En ce qui me concerne, je me suis occupé de la partie opinion ainsi que d'une partie sur les liens d'influences par rapport aux différents types de joueurs. Afin de réaliser des programmes concernant l'opinion des joueurs j'ai dû tout d'abord créer un dictionnaire contenant les paramètres initiaux. Il s'agit des probabilités qu'une personne soit contente, pas contente,etc.. Par la suite, j'ai établi un dictionnaire contenant des probabilités basé sur la logique. Il contient, par exemple la probabilité qu'un joueur content voit son opinion augmenté ou au contraire baissé. C'est ce, sur quoi est basé le programme "opinion suivante" dont je me suis occupé qui renvoie un dictionnaire mettant en clé le numéro représentant un joueur de la communauté et en valeur la satisfaction du joueur d'une échelle de 0 à 100. Ce programme prend comme paramètre un dictionnaire décrivant ce phénomène à la génération 1 et renvoie le dictionnaire mis à jour avec les nouvelles opinions à la génération n+1. C'est ce qui va être utilisé lors de notre modélisation pour générer une nouvelle opinion de la communauté au fil des mois. Après cela, j'ai également programmé la fonction qui va nous permettre de caractériser chaque type de joueur, c'est à dire les Gamers, les joueurs lambda et les influenceurs. Le statut de chaque joueur aura ensuite une importance sur le degré d'influence dont il pourra bénéficier.
En termes de connaissances acquises sur un projet de programmation de groupe je pense que la mission est réussie. Il s'agit d'un atelier riche en apprentissage.

Alexis : (Network X, evolution de l'opinion en fonction de l'influence)

Mon objectif étant de modéliser les résultats obtenus grâce aux programmes réalisés par mes deux collègues, j'ai eu besoin de me renseigner et d'apprendre a utiliser la bibliothèque python NetworkX pour pouvoir réaliser des graphiques mettant en scène notre communauté de joueurs représentée par des points (nodes) et, dans certains graphiques, reliés entre eux par des liens d'influences (strings). Je me suis donc chargé de modéliser graphiquement l'évolution de l'opinion personnelle des joueurs a l'aide des programmes élaborés par mes collègues et me suis également chargé de la majeure partie de la mise à jour de l'opinion de chaque joueur en fonction des joueurs auxquels il est relié, nécessitant au préalable la

création d'un dictionnaire représentant le poids de l'opinion de chaque type de joueur par rapport à un autre, pour pouvoir en effectuer la moyenne pondérée. La partie la plus difficile aura été de prendre en compte les différents degrés d'influences que font les joueurs entre eux et les prendre en considération lors de la mise à jour de l'opinion, car dans nos essais de base, les liens d'influences étaient bien affichés sur l'image du graphique mais n'étaient pas rajoutes dans le graphique en lui meme, erreur difficile à détecter, et qui faussait nos résultats. Une recherche sur les liens d'influences NetworkX sur internet aura permis de regler ce probleme.

IV) ANALYSE DES RÉSULTATS :

Après avoir terminé l'élaboration de notre modèle à l'aide de la bibliothèque NetworkX, nous avons établi différents tests sur notre projet afin de constater l'évolution de la communauté en fonction de ces paramètres. Nous avons obtenu un graphique liant l'opinion personnelles des joueurs ainsi que la deuxième opinion qui est la conséquence des rapports d'influences avec les autres joueurs. Dans l'ensemble des tests effectués, globalement l'opinion personnelle moyenne tend vers l'insatisfaction. Nous en avons décidé ainsi car nous estimons qu'au bout d'un certain temps les joueurs sont lassé d'un jeu en question.

1) Tout d'abord, nous avons intégré dans la communauté un influenceur très content, c'est à dire avec une satisfaction de 100/100. Nous avons constaté de cela, qu'au bout de 300 mois (25 ans) les individus reliés aux influenceurs voient leur opinion augmenté dans la mesure où l'influenceur les impactent positivement. Les réseaux de joueurs contents continuent sur cette voie en restant content au cours du temps. Sur la courbe, nous voyons que malgré l'action de l'influenceur sur la communauté l'opinion personnelle l'emporte sur l'influence et donc l'opinion générale diminue. (60% vs 40%)

2) Dans le second test, on a disposé un influenceur 'Pas content', c'est à dire une satisfaction de 0/100. Comme attendu, l'opinion des individus liés à l'influenceur tend vers l'opinion de l'influenceur. Nous constatons sur la courbe que l'influence qui varie et très vite et tend à se stabiliser vers une valeur très basse. Suite à cela, c'est l'opinion personnelle qui détermine l'évolution et puisqu'elle diminue constamment, la courbe va forcément décroître. On constate une courbe exponentielle décroissante.

3) Dans ce troisième test, nous avons retiré la présence d'influenceurs, et observé l'évolution de l'opinion générale. Il est possible de constater une baisse de l'opinion générale, qui tend à se stabiliser avec le temps sur une opinion en dessous de 50.

4) Par analogie à la première expérience, nous avons ensuite introduit deux influenceurs content. Nous avons remarqué une diminution comme d'habitude dû aux probabilités de base que nous avons décidé au préalable. Cependant, lorsqu'on retrouve deux influenceurs contents dans la communauté, on constate une diminution moins rapide de l'opinion vers une opinion générale faible.

5) Ensuite, nous avons fait de même, en ce qui concerne l'introduction de deux influenceurs pas contents dans la communauté. Encore une fois, on remarque une décroissance exponentielle de l'opinion générale.

6) Dans ce troisième test, nous avons choisi de mettre en confrontation deux influenceurs, l'un avec une opinion de 0/100 et l'autre avec une opinion de 100/100. On observe que la majorité des joueurs liés aux deux influenceurs voient leur opinion modifiée autour de l'état 'Content', puis au final se modifient petit à petit pour atteindre l'état moyen 'Moyen content'. Cette variation se traduit par une baisse lente et linéaire de l'opinion générale au fil des générations, 50/100 lors de la génération 0 pour 40/100 lors de la génération 300. On constate une courbe décroissante linéaire. Cette courbe voit son opinion se dégrader, mais avec une vitesse moins élevée causé par la compensation entre les deux influenceurs.

7) Pour ce test, nous avons choisi d'augmenter la probabilité de variation croissante de l'opinion personnelle et, analogiquement, de baisser celle de variation décroissante ainsi que de stagnation de celle-ci. Nous avons donc retiré tous les influenceurs du graphique pour effectuer ce test. On observe que l'opinion personnelle varie assez rapidement vers des valeurs élevées à cause des nouvelles probabilités de variation, et notamment que l'opinion augmente beaucoup au cours des premiers mois, puis se stabilise et oscille autour d'une même valeur, à cause de la valeur moyenne de l'opinion personnelle qui se stabilise autour d'une valeur élevée assez rapidement, et de l'influence moyenne qui se stabilise également autour d'une valeur plutôt moyenne (environ 50/100).

8) (Bad Buzz) Pour ce test, nous avons intégré un influenceur content, avec l'apparition au mois 150, d'un bad buzz qui va influencer négativement les

joueurs. On constate sur la courbe que l'opinion baisse beaucoup lors du mois 150 quand bien même l'influenceur est présent. On obtient par la suite une courbe qui est stable au début et qui finit par décroître à partir du mois 150. Nous constatons sur la courbe, qu'après le mois nbuzz=150, on a une progression de l'opinion qui est causé par l'influenceur très content.

9) (Good Buzz) Pour ce test, nous avons intégré un influenceur pas content, avec l'apparition au mois 150, d'un good buzz qui va influencer positivement les joueurs. Nous voyons sur le graphique NetworkX un ensemble de points de couleurs vertes (Content) lors de l'apparition du good buzz. Sur la courbe, nous voyons un pic d'opinion positive au cours du mois nbuzz=150. Par la suite, on a une diminution de l'opinion, causé par l'influenceur pas content qui va impacter négativement l'opinion.

V) CONCLUSION :

Le modèle obtenu n'est pas totalement en adéquation avec celui souhaité dû à des problèmes d'effectifs et de temps. Nous avions mal jugé le travail nécessaire et la difficulté de notre idée initiale. Le manque de connaissances nous a empêché d'aboutir au projet souhaité au départ. De plus, les probabilités utilisés lors de la construction de nos dictionnaires des paramètres initiaux ne sont pas véridiques à 100%. Les incertitudes liés à ces probabilités pourraient être améliorés.
Nous aurions voulu avoir une certaine relation plus accentué entre l'opinion personnelle et l'influence des liens. Ce qui aurait pu donner, à notre goût, un modèle beaucoup plus abouti.

V) ANNEXES :

1°) Paramètres Initiaux :

```
p = {'Pas content' : 0.1,
     'Moyen content': 0.4,
     'Content': 0.3,
     'Très content': 0.2}

#Ces dictionaires représentent la probabilité que l'opinion d'un individu quelconque augmente
#(pCa, pMCa, pTCa, pPCa) ou au contraire baisse (pCb, pMCb, pTCb, pPCb) de 20%, 10%, 30% en fonction
#de l'opinion du joueur au départ

pCa={'20%': 0.5,
     '10%':0.3,
     '30%':0.2}

pCb={'20%': 0.3,
    '10%':0.5,
    '30%':0.2}


pMCa={'20%': 0.4,
     '10%':0.15,
     '30%':0.45}

pMCb={'20%': 0.4,
    '10%':0.15,
    '30%':0.45}


pTCa={'20%': 0.4,
     '10%':0.5,
     '30%':0.1}


pTCb={'20%': 0.45,
    '10%':0.4,
    '30%':0.15}


pPCa={'20%': 0.45,
     '10%':0.4,
     '30%':0.15}


pPCb={'20%': 0.4,
    '10%':0.5,
    '30%':0.1}

pp={'Gamer' : 0.4,
   'Joueur lambda': 0.599,
   'Influenceur':0.001}
#pp= répartition aléatoire du type de joueur arrivant sur la plateforme indiquant son taux d'attirance
```

```
#AAAP : dictionnaire generant l'opinion personnelle initiale de chaque personne de la communaute,
#utilise plus tard pour generer la base du programme generant le graphique representant l'evolution de
#l'opinion personnelle de chaque individu.
AAAP = avis(pi,p)


print(AAAP)

#AAA : dictionnaire generant l'avis initial de chaque personne de la communaute initiale,
#utilise plus tard pour generer la base du programme generant le graphique representant l'influence des liens
#entre individus.
AAA = avis(pi,p)
print(AAA)

#copie du dictionnaire generant l'opinion personnelle initiale, utilise plus tard pour ne pas modifier la
#reference du test.
AAAPP = AAAP.copy()
print('y',AAAPP)


#II : dictionnaire associant a chaque lien entre type de joueur un poids qui sera pris en compte
#dans la mise a jour de l'influence des liens pour chaque individu de la communaute.
II = {'Gamer' : {'Gamer' : 1.5,
                 'Joueur lambda' : 0.7 ,
                 'Influenceur' : 50 },
     'Joueur lambda' : {'Gamer' : 1.5 ,
                 'Joueur lambda' : 1.1 ,
                 'Influenceur' : 50 },
     'Influenceur' : {'Gamer' : 0,
                 'Joueur lambda' : 0 ,
                 'Influenceur' : 1.2 }}
```

2°) Programmes concernant l'opinion des joueurs :

```
def opinion_utilisateur(p):
    """retourne l'opinion d'un utilisateur"""
    u = np.random.random()
    s = 0
    liste= ['Pas content','Moyen content','Content','Très content']
    for k in liste:
        s += p[k]
        if u < s:
            indice = k
            break

    if indice=='Pas content':
        return(random.randint(0,24))
    if indice=='Moyen content':
        return(random.randint(25,49))
    if indice=='Content':
        return(random.randint(50,74))
    return(random.randint(75,100))
print(opinion_utilisateur(p))

4.

def avis(n,p):
    """retourne un dictionnaire représentant la proportion de chaque catégorie d'individu"""
    z=dict()
    for i in range(1,n):
        z[i]=opinion_utilisateur(p)
    return z
print(avis(10,p))

def dic_statut(D):
    'renvoie un dictionnaire representant l etat de satisfaction de chaque individu'
    Etat = dict()
    for i in D:
        if 0<=D[i] and D[i]<=24 :
            Etat[i] = 'Pas content'
        elif 24<D[i] and D[i]<=49 :
            Etat[i] = 'Moyen content'
        elif 49<D[i] and D[i]<=74 :
            Etat[i] = 'Content'
        else :
            Etat[i] = 'Très content'
    return Etat
```

```
def opinion_suivante(A,pCa,pCb,pMCa,pMCb,pTCa,pTCb,pPCa,pPCb):
    """retourne un dictionnaire représentant la proportion d'individus à la génération n+1"""
    z= dict()
    zz=A
    for i in zz:
        a=random.uniform(0,1)
        if zz[i]>=0 and zz[i]<=24 :
            if a>=0 and a<=0.33:
                zz[i]+=variation(zz[i],pPCa)
            if a>0.33 and a<=0.66:
                zz[i]=zz[i] - variation(zz[i],pPCb)

        if zz[i]>=25 and zz[i]<=49:
            if a>=0 and a<=0.33:
                zz[i]+=variation(zz[i],pMCa)
            if a>0.33 and a<=0.66:
                zz[i]=zz[i] - variation(zz[i],pMCb)

        if zz[i]>=50 and zz[i]<=74 :
            if a>=0 and a<=0.33:
                zz[i]+=variation(zz[i],pCa)
            if a>0.33 and a<=0.66:
                zz[i]=zz[i] - variation(zz[i],pCb)

        if zz[i]>=75 and zz[i]<=100 :
            if a>=0 and a<=0.33:
                zz[i]+=variation(zz[i],pTCa)
            if a>0.33 and a<=0.66:
                zz[i]=zz[i] - variation(zz[i],pTCb)
        if zz[i]>100:
            zz[i]=100
        if zz[i]<0:
            zz[i]=0
    return zz
```

3°) Programmes utilisant NetworkX :

```
    #condition de buzz et conséquences
        if e == nbuzz :
            for i in range(1,pi):
                rbuzz = random.uniform(0,1)
                if rbuzz<=0.6:
                    Opinion1[i] = 15

        Opinion1[79] = 75
        AAAL = Opinion1

        for i in Opinion1:
            S += Opinion1[i]
            compteur += 1

        FINALL= int(S/compteur)

        OPINION_FINALE_L_MOY[e]=FINALL

        EtatL = dict()

        EtatL = dic_statut(AAAL)



        plt.figure(figsize=(10, 10))
        nx.draw_networkx_nodes(I,pos,nodelist = [i for i in range(1,pi)],
                               node_size=50,
                               node_color= [E[EtatL[i]] for i in range(1,pi)])


        nx.draw_networkx_edges(I,pos,edgelist = W.edges(), alpha = 0.4)
        plt.xlim(-0.05, 1.05)
        plt.ylim(-0.05, 1.05)
        plt.axis('off')
        plt.savefig("Courbe OLiens"+str(e)+".png", format="PNG")
        plt.show()
        print(FINALL,OPINION_FINALE_L_MOY[e])
```

```
AAAPP = dict()
AAAPP = AAAP.copy()
#OPINION_FINALE_O_MOY : dictionnaire representant pour chaque mois l'opinion personnelle moyenne entre tous les indivi
OPINION_FINALE_O_MOY = dict()
#FINALO : Opinion personnelle moyenne
FINALO = dict()
S=0
compteur=0

EtatO = dict()
EtatO = dic_statut(AAAPP)

plt.figure(figsize=(10, 10))
nx.draw_networkx_nodes(I,pos,nodelist=[i for i in range(1,pi)], node_size = 50, node_color=[E[EtatO[i]] for i in range
plt.xlim(-0.05, 1.05)
plt.ylim(-0.05, 1.05)
plt.axis('off')
plt.show()

for i in AAAPP:
    S += AAAPP[i]
    compteur += 1

FINALO= int(S/compteur)

OPINION_FINALE_O_MOY[0]=FINALO


for e in range(1,n):
    EtatO = dic_statut(AAAPP)
    AAAPP = opinion_suivante(AAAPP,pCa,pCb,pMCa,pMCb,pTCa,pTCb,pPCa,pPCb)
    S=0
    compteur=0

    for i in AAAPP:
        S += AAAPP[i]
        compteur += 1

    FINALO= int(S/compteur)

    OPINION_FINALE_O_MOY[e]=FINALO
    plt.figure(figsize=(10, 10))
    nx.draw_networkx_nodes(I,pos,nodelist=[i for i in range(1,pi)], node_size = 50, node_color=[E[EtatO[i]] for i in r
    plt.xlim(-0.05, 1.05)
    plt.ylim(-0.05, 1.05)
    plt.axis('off')
    plt.savefig("Courbe Ogenerale"+str(e)+".png", format="PNG")
    plt.show()
```

```
##copie du dictionnaire generant l'influence des liens initiale, utilise plus tard pour ne pas modifier
#la reference du test.
AAAL = AAA.copy()

#n : nombre de mois
n = 300

#mois au cours duquel le buzz va arriver
nbuzz = 150
print(AAAL)

#influence des liens moyenne à chaque génération
OPINION_FINALE_L_MOY = dict()



T[79] = 'Influenceur'


for e in range(0,n) :
    S = 0
#FINALL : entier representant la moyenne de l'influence des liens moyenne, actualise a chaque generation.
    FINALL = 0
    compteur = 0
#Opinion1 : dictionnaire representant l'influence des liens sur chaque individu, actualise a chaque generation
    Opinion1 = dict()


#mise à jour de l'influence des liens
    for i in range(1,pi):
        Opinion1[i] = AAAL[i]
        compteur+=1
        for j in range(1,pi):
            if (i,j) in I.edges() :
                Opinion1[i] += II[T[i]][T[j]]*AAAL[j]
                compteur +=II[T[i]][T[j]]
        if compteur == 0 :
            Opinion1[i] = 0
        else :
            Opinion1[i] = int(Opinion1[i]/compteur)
        compteur = 0
```