## Gyanú felderítés online szerepjátékban solver segítségével

(Suspicion generation for online roleplay games based on Solver)

Kivonat: A gyanúgenerálás a játékok log-adatai alapján mindenkor lehetséges, legfeljebb ennek pontossága lehet adott esetben kérdéses. A gyanúskála az anti-diszkriminatív modellezés keretében áll elő, amennyiben beigazolódik valid (függvényszimmetria-támogatással), hogy nem minden objektum lehet másként egyforma.

Kulcsszavak: log, skála, hasonlóság, objektivitás, optimalizálás, automatizálás

Abstract: The suspicion generation for games based on log data is always possible. The scale of the suspicion can be defined based on similarities where the principle of the potential sameness of each object can not be derived and the anti-discriminative estimations are valid through function symmetries.

Keywords: log, scale, similarity, objectivity, optimization, automation

Szerző: Száraz Bence

## Bevezetés

2017-ben egy ismerősömmel indítottunk egy hobby projektet, minek célja egy minőségi MU Online (online szerepjáték) szerver felállítása és üzemeltetése volt. A kivitelezés sikeres volt, a termék +/- egy éven keresztül több ezer accountnyi felhasználóval üzemelt.

### Célok

Habár a szerver már bezárta kapuit, partneremmel sokat beszélgettünk egy esetleges újra nyitásról és új szolgáltatásokról.

Régebben felmerültek problémák a játékban előforduló mechanikák nem tervezett módon való kihasználásából vagy egyéb rosszindulatú előnyszerzési módokkal kapcsolatban.

A célja a projektnek az lenne, hogy a rendelkezésre álló játékszerver által generált adattömegben valamilyen logika alapján szűrni azokat a játékosokat, akik csalás vagy más nem kívánt tevékenység kihasználásával gyorsabban haladnak karakterfejlődésükben.

### Feladatok

#1 A Gameserver megadott sémában lenyomatot készít a játékosok közötti cserék, ki-be lépések, tárgy mentések és más fontos tranzakciók során, majd napi lebontásban, avagy maximum 2 GB file mérettel log-fájlokba menti ezt. Az első komplikációt ennek a file összességnek a feldolgozása jelenti, szükség lesz egy szoftverre, amely képes beolvasni és értelmezhető formába rendezni az adatkáoszt.

#2 Meg kell alkossunk egy adatmodellt, ami a karakter progresszió szempontjából jellemző tulajdonságokat a log-sorokból kinyerhető módon tartalmazza és ezen tulajdonságok az idő elteltével egyenesen arányos képet adnak a karakter erősségre.

#3 A beolvasó szoftvert ki kell egészítsük oly módon, hogy a modellünkre illeszkedő információs sorokká transzformálja a beolvasott információt.

#4 A rendelkezésre álló adattömeget feldolgozható méretűre kell redukálnunk.

#5 Vonjuk le a következtetéseket.

### Motivációk

A probléma alapból is érdekes számomra, nem csak a MU-szerver kapcsán. A megoldás felhasználható bármiféle későbbi projektemhez, hogy jobb minőségű terméket hozzak létre.

### Célcsoportok

Természetesen az eszköz nem feltétlen saját felhasználású. A MU-Online custom szervereknek megvan a saját rajongó bázisa, még manapság is több száz privát szerver online felhasználók százezreivel. Az összes jelentősebb létszámmal rendelkező szerver üzemeltetői potenciális vevők.

### Hasznosság

Mint hajdani szerver tulajdonos nyilatkozom, egy ilyen tool-ért némi pénzt fizetni jól megtérülő befektetés. Nem csak a szerver minőségét dobja meg, ha szűrni tudjuk a fekete bárányokat, de az esetleges problémákra is hamarabb fény derül és megelőzhetjük a különböző „abuse” drámákat, ha monitorozzuk az emberek fejlődését.

### A probléma/jelenség története

A piacon rengeteg vállalkozás foglalkozik egy szerver futás közben generált adatainak tárolásával, jellemzően e-közben teljesítményt monitoroznak, majd ezeket az eltárolt információkból grafikonokat hoznak létre.

Léteznek olyan játékok, ahol az üzemeltetők eltárolják az összes felhasználóktól érkező inputot, amelyek ismeretében a játékmenet teljes egészében rekonstruálható (pont olyan elven, ahogy egy sakk meccs rekonstruálható a felírt lépések újra játszásával). Ezeket az inputokat jellemzően replay szolgáltatásokra használják. Találtam néhány tanulmányt (vö. [Referenciák fejezet](#_Referenciák:)), ahol ezeket a log-okat First Person Shooter (FPS mostantól) játékokban csalók kiszűrésére használják jellemzően machine learning algoritmusok segítségével.

Olyan példát sajnos nem találtam, hogy szerepjátékok esetében készült ilyen feladatot ellátó termék vagy tanulmány. Valószínűsítem, hogy a nagyobb játék fejlesztő vállalatok rendelkeznek valamiféle belső fejlesztésű analitikai programmal, de ezek nem publikusan elérhetőek.

### A probléma/jelenség aktuális állapota

Az FPS játékok csalás detektáló esete hasonló probléma az enyémhez, de a bemenő adatok jellege teljesen eltérő. Az fps-ekben jellemzően körök, meccsek vannak, sebesség, célzási vonal, falak. Ez összehasonlíthatatlan egy szerepjátékban értelmezett információtömeggel. Sajnos így a modell megalkotásával magamra vagyok hagyva.

## Adatok és módszerek

### Saját adatvagyon

Az adatok, amiből én kiindultam, egy három hónapos időtartamot fednek le egy még ki nem forrott szerverkonfiguráció élettartamának elejéről. A rendelkezésre álló log-ok így több szempontból is szerencsések:

1. A tökéletlen konfiguráció azt jelenti, hogy vannak nem tervezett hibák, amiket a felhasználók egy része biztosan ki is használt és azt várjuk, hogy ez valahogy megjelenjen a kimutatásban.
2. A szerver kiinduló fázisa jó eséllyel jobban reflektálja a karakterfejlődést és annak sebességét ahhoz képest, ha egy olyan szakaszt vizsgálnánk, ahol a felhasználók többsége már az utolsó százalékokat szedi össze.
3. További előny az adat mélyenszántó ismerete, emlékek a felhasználókkal való interakciókra és tippek a várható kimenetelre az elemzés szempontjából.

A log-fájlok minden változást tartalmaznak egy karakter életciklusában. Amint a felhasználó szintet lép, tárgyakat cserél, ki-be lép a szerverre egy teljes jegyzőkönyvet kapunk. A struktúra minta képpen ehhez hasonló:

[00:01:43] [JahMan][JahMan] SI[236,Muren's Book of Magic,6,0,0,0]serial:[5487168] dur:[5] Ex:[0,0,0,0,0,0,0] Set[0] 380:[0] HO:[0,0] E:[18]

[00:01:43] [JahMan][JahMan] CharInfoSave : Class=50 Level=400 LVPoint=21 Exp=-472819216 Str=700 Dex=10500 Vit=500 Energy=15572 Leadership:0 Map=56 Pk=3

[00:04:22] [blackshar8][loktide] SI[0,Soleil Scepter,9,1,1,1]serial:[4388884] dur:[0] Ex:[1,0,0,0,0,0,0] Set[0] 380:[0] HO:[0,0] E:[0]

[00:04:22] [blackshar8][loktide] SI[1,Dark Raven,0,0,0,0]serial:[5228414] dur:[244] Ex:[0,0,0,0,0,0,0] Set[0] 380:[0] HO:[0,0] E:[0]

Az pillanat mentése több sorra van leképezve. Minden sor tartalmazza a timestamp-et amikor készült, valamint a save típusát, karakter accountját, nevét stb.

A játék szempontjából leglényegesebb információk például a karakter szintje, elosztott pontjai és a nála lévő tárgyak összessége. A tárgyak tulajdonságai ugyan-így fel vannak sorolva. Érdemes a tárgyakat ritkaság és mágikus tulajdonság alapján megkülönböztetni, de az ökölszabály az, hogy minél magasabb szintű (1-től 15) és minél magasabb besorolású tulajdonsága van (Tier1>Tier2>Tier3), annál jobb.

### Saját módszertan

A problémát abból az oldalból közelítettem meg, ha sikerül egy jó adat modellt létrehozzak, melynek sorai az egyes játékosok, akkor egy Solver algoritmus (anti-diszkriminatív elvek mentén) meg tudja mondani azok közül melyek a leggyanúsabbak (vö. <https://miau.my-x.hu/myx-free/>).

A nyers log beolvasása c# objektumokba

A szerverfájlok birtokában pontosan meghatározható a minta, amit a kiírt sorok követnek. Ennek tudatában a c# erős eszközöket nyújt a feldolgozásra. Kézenfekvő választás az objektum orientáltság (amikben majd felfogható lesz az adat), könnyű file kezelés és a „RegularExpressions” library miatt.

Amit végül is szeretnénk az a karaktereket tartalmazó időpecséthez tartozó objekutok sokasága. Ehhez először is definiáltam egy egyszerű struktúrát, ami képes ilyen módon tárolni a beolvasott sorokat. Született egy „Item” osztály a tárgyak játék béli tulajdonságaival és egy „Char” osztály a kraktereknek, melyek tetszőleges mennyiségű tárgyat is tartalmazhatnak.





A program egy mappában megtalálható összes log-ot beolvassa időrendben és soronként egy „LineParser”-nek átadja, melynek az a feladata a szekvenciálisan érkező adat Regexel való értelmezése és szegmentálása a különböző karakterekre vonatkozó információkra.

A felhasznált beolvasó pattern-ek:

private readonly Regex gObjItemTextSave = new Regex(@"\[(.+)\] \[(.+)\]\[(.+)\] SI\[(\d+),(.+),(\d+),(\d+),(\d+),(\d+)\]serial:\[(\d+)\] dur:\[(\d+)\] Ex:\[(\d+),(\d+),(\d+),(\d+),(\d+),(\d+),(\d+)\] Set\[(\d+)\] 380:\[(\d+)\] HO:\[(\d+),(\d+)\] E:\[(\d+)\]");

private readonly Regex gObjStatTextSave = new Regex(@"\[(.+)\] \[(.+)\]\[(.+)\] CharInfoSave : Class=(\d+) Level=(\d+) LVPoint=(\d+) Exp=(-?\d+) Str=(\d+) Dex=(\d+) Vit=(\d+) Energy=(\d+) Leadership:(\d+) Map=(\d+) Pk=(\d+)");

private readonly Regex gObjWarehouseTextSave = new Regex(@"\[(.+)\] \[(.+)\]\[(.+)\] SW \[(\d+),(.+),(\d+),(\d+),(\d+),(\d+)\]serial:\[(\d+)\] dur:\[(\d+)\] Ex:\[(\d+),(\d+),(\d+),(\d+),(\d+),(\d+),(\d+)\] Set:\[(\d+)\] 380:\[(\d+)\] HO:\[(\d+),(\d+)\] E:\[(\d+)");

Az így tárolt adattal már bármit kezdhetünk!

Adatmodell megalkotása

Talán a legfontosabb lépés a precíz végeredmény szempontjából, hogy beszédes oszlopokat vegyünk fel az OAM-ba. A játékkal való tapasztalatomra hagyatkozva és a modellre vonatkozó célszerűségből (pl: Jó ötletnek tűnik a tárgyak tulajdonság szerinti szétbontása. Ezzel a módszerrel a kevésbé értékes ingóságok is részletesen reprezentálva lesznek a statisztikában. Továbbá ezek kombinációja így több információt tartalmaz, mintha valami konkrét dolog meglétét vizsgálnánk (pl: van-e tökéletes szettje = min 5db 15ös és 2xt1 cucc)) fakadóan a következő lett a végeredmény:



1. Ábra: Adatmodell, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, Data-munkalap) – jelmagyarázat: az első sor az irányokat tartalmazza

Account – Account neve (kulcsmező)

Karakter – Karakter neve (kulcsmező)

Rekord ideje – lenyomat ideje (kulcsmező)

Summa stat – a karakter tapasztalatpontjaiból kiváltott tényezők összessége. Valójában ennek az összegnek egy konstansod része a karakter által lépett szintek összessége, ami talán az egyik legjobb fejlődési jelző.

Item 1-9 – Tárgyak darabszáma lvl 1 és 9 között

Item 10-12 – Tárgyak darabszáma lvl 10 és 13 között

Item 13-15 – Tárgyak darabszáma lvl 13 és 15 között

Item with T1 – Ritka tárgy tökéletes tulajdonsággal

Item with T2 – Ritka tárgy elfogadható tulajdonsággal

Item with T1 and T2 – Ritka tárgy tökéletes és elfogadható tulajdonsággal is.

Item with 2x T1 – Ritka tárgy kettő tökéletes tulajdonsággal. Ennél jobb nincsen.

Szárny t2 – második generációs szárnyak száma (A szárny egy meghatározó tárgyfajta)

Szárny T3 – harmadik generációs szárnyak száma

Szárny T4 – negyedik generációs szárnyak száma

Szárny lvl – legjobb szárny szintje

Kristályok – A játékban fizetőeszköznek használt kristályok darabszáma

Talizmán – A kristályoknál jóval értékesebb másik fizetőeszköz darabszáma

Objektumok leképezése az adatmodellre

A beolvasott logot az OAM-re illeszkedő, tabulátorokkal ellátott sorokba írtam, így a transzformált adat beimportálható legyen egy Excel táblába. Az Excelt használni kimutatásokra és szelekciókra jelentősen egyszerűbb, mint lekódolni azokat a funkciókat.

Egy karakterre vonatkozó sort a következő kódrészlet adja vissza:



Az így keletkezett c# program az adattömeget 1-2 óra lefolyása alatt generálta le. Némi optimalizálással ezen lehetne még javítani, de ez most nem volt cél.

Adatszűrés

A három hónapnyi logból 93656 karakter-állapot generálódott.

Ez a mennyiség meghaladja azt a keretet amire számítási kapacitásunk van, így muszáj szűkítenünk az állományt.

Célszerű azokkal az emberekkel foglalkozni, akikről az átlagnál több adatunk van (Így az átlagosnál többet is játszottak). Azok, akik csak pár órát játszottak nem is szignifikánsak a megoldandó kérdés szempontjából.

Az adatokat „Account” -ra szűrve azt kapjuk, hogy a 93656 bejegyzés 162account között oszlik meg.



2. Ábra: Account-szűrés, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Accounts” munkalap)

Az accountra való szűrés viszont pontatlan lehet a „kis-karakterek” miatt, melyeket kipróbálás vagy más célból hoznak létre a játékosok.

Ezen felülkerekedve szűrhetünk karakterekre is, ahol tovább szűkíthetjük a teret, ha megflageljük azokat a neveket, akikről tegyük fel több mint 1000 bejegyzés készült plusz kontrolnak betettem két karaktert, amivel a szerver adminok játszottak kontrol gyanánt. Ez már egy sokkal barátságosabb 34 elemes csoportot ad.



…



3. Ábra: Karaktereként való szűrés, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Char” munkalap)

Összevonás

Habár a résztvevők számát lecsökkentettük, még mindig sorok ezreiről van szó. Az eredeti adatot tartalmazó panelen létrehoztam egy szűrő-mezőt, ami a „karakter” -szűrt tábláról név, mint kulcs mező által referált flag-eket tartalmazza. Így látszik mely sorokat „tartjuk meg” (vö. 4. ábra).



4. Ábra: Adatsorok flag-elése, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Data” munkalap, fókuszban az A-oszlop)

Hogy a karakterekről tudjunk egy sorban beszélni a rájuk vonatkozó adatokat érdemes azok átlagával jellemezni, ez reprezentálja legjobban az idő tényezőt az összevonási módszerek közül.

A flag-gel ellátott 34 „versenyző” -táblázata a következőképpen alakul a szűrés és átlagolás után:



5. Ábra: Kiátlagolt adatsorok, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Kimutatás2” munkalap)

Kimutatás

Utólag a modellhez felvettem még két érdekesnek mutatkozó oszlopot, „item meredekség”( =MEREDEKSÉG(B4:D4,$C$1:$E$1) item9-13-és15) és „wing meredekség”( =MEREDEKSÉG(H4:I4,$H$1:$I$1)) néven. Ez a meredekség egy jó mérőszám lehet a karakter általános tárgy fejlesztési ütemére (vö. 6. ábra).



6. Ábra: Meredekség oszlopok a modellben, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Kimutatás2” munkalap, N és M oszlop)

Miután minden érték adott volt. Sorszámoztam az oszlopokat, hogy látszódjon melyik karakter hányadik „helyezést” érte el az adott szempont szerint (vö. 7. ábra).



7. Ábra: Sorszámozott átlagok, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Kimutatás2” munkalap, R-től AB oszlop)

A karakterek helyezéseinek átlagolásával kapunk egy naiv becslést a gyanús fejlődési ütem mértékére (vö. 8. ábra):



8. Ábra: Naiv becslés, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Kimutatás2” munkalap, AD oszlop)

1000-es Y0 értékek mellett a COCO solver (<https://miau.my-x.hu/myx-free/coco/beker_y0.php>) a következő esztimációt generálja (vö. 9. ábra):



9. Ábra: COCO solver output, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Y0-2” munkalap, 115. sortól)

A becslés eredményeket visszareferálva a táblázatunkba a kimutatás ezt az eredményt adta (vö. 10. ábra):



10. Ábra: Optimum, forrás: (<https://miau.my-x.hu/miau/279/esport.xlsx>, „Kimutatás2” munkalap, AE oszlop)

## Eredmények

### Hipotézisek/elvárások/kérdések

A kimutatás egyértelműen anomáliát vél felfedezni az „UNDISPUTED” nevű felhasználónál. Ez számunka rendkívül jó hír, ugyan is ez az account volt az egyetlen, amit ismételt abuse és csalás miatt ki kellett tiltani a szerverről.

Ha a naiv becslést összehasonlítjuk a Solver által számított optimummal érdemes néhány jelenségre rámutatni:

1. A naiv szerint legártalmatlanabb emberek a Solver szerint is ártatlanok.
2. A leggyanúsabb ember a solver szerint is a leggyanúsabb.
3. A két game master karakter viszont a Solver szerint inkább közömbös, mint gyanús.

A harmadik pont igencsak jó hír. Habár azok a karakterek generálva lettek és semmi sem „legit” rajtuk, jó reprezentációk arra az esetre, ha már egy játékos elérte közel a maximumot. Önmagában a magas értékeknek nem szabadna gyanúsnak lenni és ezt a Solver érzi is.

### Válaszok/állapotok

Az eredmény elemzése után a rendszer alkalmasnak tűnik a rosszindulatú tevékenységeket folytató felhasználók kiszűrésére.

## Vita

Az online szerepjátékok nagy része hasonló fejlődési elvet követ. Ebből kiindulva a rendszer modell és gyanúkeltő része alkalmazható minden olyan más játék esetében is, ahol létezik egyértelmű leképezés azon játék karakter felszerelés modelljének oszlopai és a fent használtak között.

Feltehető a kérdés, mik azok az akciók, amiket végre merünk hajtani az eredmény birtokában? Például csak az egyértelmű gyanú elegendő-e arra, hogy kitiltsuk a felhasználót?

Jó lenne a rendszert nagyobb adatvagyonon mondjuk egy évnyi adaton átfuttatni több időszelvényben is, hogy látszanak az esetleg kritikus szakaszok. Sajnos ehhez jelenleg nincs rendelkezésre álló log-file.

## Következtetések

A módszer legszebb részei, hogy az első ránézésre gyanúsnak tűnő elemeket is elfogultság nélkül tudja elemezni (game master karakterek esete) és a rossz indulatúakat összekeverhetetlenül mutatja.

## Jövőkép

Ha a c# részt optimalizálnám lehetőség lenne a rendszeres vagy folyamatos futtatásra. A napi log-ok lementése után így rögtön kiderülne, ha valami nincs rendben.

Az Excelben lévő szűréseket és adat transzformációkat le lehetne kódolni és integrálni a logReader programba, így megspórolnánk a Microsofttól való függést. A készen álló adatot automatikusan elküldeni a Solver-nek, majd az eredményt egy html oldalon formázni és megjeleníteni.

## Mellékletek

### Tartalomjegyzék

[Gyanú felderítés online szerepjátékban solver segítségével 1](#_Toc90497662)

[Bevezetés 1](#_Toc90497663)

[Célok 1](#_Toc90497664)

[Feladatok 1](#_Toc90497665)

[Motivációk 1](#_Toc90497666)

[Célcsoportok 1](#_Toc90497667)

[Hasznosság 2](#_Toc90497668)

[A probléma/jelenség története 2](#_Toc90497669)

[A probléma/jelenség aktuális állapota 2](#_Toc90497670)

[Adatok és módszerek 3](#_Toc90497671)

[Saját adatvagyon 3](#_Toc90497672)

[Saját módszertan 4](#_Toc90497673)

[Eredmények 15](#_Toc90497674)

[Hipotézisek/elvárások/kérdések 15](#_Toc90497675)

[Válaszok/állapotok 15](#_Toc90497676)

[Vita 15](#_Toc90497677)

[Következtetések 15](#_Toc90497678)

[Jövőkép 15](#_Toc90497679)

[Mellékletek 16](#_Toc90497680)

[Tartalomjegyzék 16](#_Toc90497681)

[Rövidítések, szómagyarázatok: 16](#_Toc90497682)

[Referenciák: 17](#_Toc90497683)

[Behavioral-based cheating detection in online first person shooters using machine learning techniques 17](#_Toc90497684)

### Rövidítések, szómagyarázatok:

* Custom = egyedi, mint „custom szerver” az eredeti forgalmazótól külön való szerver megoldás.
* Tool = szoftver eszköz.
* Abuse = szoftver értelemben: nem rendeltetés szerű használatból kovácsolt előnyszerzés.
* Pattern = mintázat.
* OAM = objektum-attribútum-mátrix
* Flag = két különböző lehetséges állapotot megengedő adattároló egység (1-0, igaz vagy hamis).
* Admin = felelős adminisztrátor.
* Legit = bizonyíthatóan szabályszerű.

### Referenciák:

# [Behavioral-based cheating detection in online first person shooters using machine learning techniques](https://www.researchgate.net/publication/261497438_Behavioral-based_cheating_detection_in_online_first_person_shooters_using_machine_learning_techniques)

[Cheat Detection using Machine Learning within Counter-Strike: Global Offensive](https://openworks.wooster.edu/cgi/viewcontent.cgi?article=11803&context=independentstudy)