Preporuka za odabir heroja u MOBA igrama (DOTA 2)

Članovi tima:

1. Nenad Raković, SW45-2019.

Motivacija:

Kao strastveni igrač MOBA (Multiplayer Online Battle Arena) igara, susreo sam se više puta sa problemom odabira heroja. Potrebno je određeno vreme da se sama igra i funkcionisanje svih heroja nauče. Tu nastaje problem u slučaju DOTA 2 igre.

DOTA 2 igra nije ni malo laka. Novi igrači dolaze, pokušaju da se snađu u njoj, ali velika većina nakon kratkog vremena jednostavno odustane. Smatram da jedan od većih faktora predstavlja I sam odabir heroja što je zapravo prvi korak ka pristupanju samoj igri.

Verujem da bi algoritam za preporuku heroja drastično smanjio problem malog broja novih igrača I učinio samu igru boljom.

Pregled problema:

Na sreću igrača, takvi algoritmi već postoje, ali smatram da nisu u dovoljnoj meri precizni I da postoji prostor za njihovo poboljšanje.

Jedan takav primer predstavlja sajt "*Dota 2 Hero Picker*" (*http://dotapicker.com*). Sajt iz datog domena očekuje izabrane heroje kao ulazne vrednosti, isto tako I heroje koji su banovani, tj. ne mogu biti izabrani u trenutnoj partiji.

Algoritam funkcioniše po principu da poseduje pojedina pravila prilikom preporuke heroja, neki od parametara pri preporuci su:

1. Currently picked heroes
2. Banned heroes
3. Role
4. Counter-pick
5. Combo-pick
6. Hero familiarity

7*. Meta

8*. Rank

Currently picked heroes: Ako je neki heroj već izabran, njega opet ne možemo izabrati stoga ga izbacujemo iz preporuke.

Banned heroes: Ako su neki heroji banovani pre početka igre, ne mogu se izabrati stoga se ne preporučuju.

Role: Uloga heroja u timu. Igra sadrži 5 uloga: Mid, Carry, Support, Tank i Jungle. Tim je kompletan ako se izabere po jedan heroj iz svake uloge, zbog toga ako je neki heroj već izabran, skup preporuke se sužava isključivši sve izabrane uloge.

Counter pick: Pojedini heroji su veoma loši kada protiv njih igra neko od heroja iz counter-pick liste, tu informaciju treba iskoristiti u preporuci da igrač baš izabere tog heroja kako bi stekao prednost.

Combo-pick: Neki heroji veoma dobro funkcionišu u kombinaciji sa drugim herojima u timu, za svakog heroja će se u bazu znanja ubaciti

koji su mu podložni ostali heroji za combo, tada se igraču predlažu više ti heroji, pod ispunjenim prethodnim uslovima.

Hero familiarity: Nakon dobijene prve verzije outputa, svi heroji koji mogu biti izabrani, sledi izračunavanje vrednosti za svakog heroja koliko je dobro da je izabran, jedan faktor predstavlja koliko je igrač upoznat sa samim herojem.

Na papiru, ovo predstavljaju razumne I dobre odluke za parametre, ali u praksi vlada drugačiji pristup.

Igra se vremenom menja, neki heroji postaju slabiji, neki jači. To se dešava na način što programeri ručno menjaju specifikacije samih heroja. U jednom momentu, programeri ne mogu shvatiti da li su napravili dobre ili "loše" izmene.

Nakon izmene I objave nove verzije igrice, igrači isprobavaju razne varijacije I traže "rupe" u igrici, što je zapravo slučaj u skoro svakoj verziji. Veoma se često desi da su neki heroji jednostavno mnogo "jači" od ostalih, pa je slučaj da se u svakoj partiji moraju ili izabrati ili banovati. Takve heroje nazivamo Meta herojima.

Tu nastaje scenario gde bi novi igrač znajući da je pojedini heroj "jači" od ostalih izabrao baš njega I stekao prednost. Igračima bi tada igra bila zanimljivija I veće su šanse da će igrač odigrati više partija, dodatno naučiti pravila igre I verovatno ne odustati nakon kratkog vremena.

Trenutni algoritmi ne poseduju tako nešto, a smatram da je veoma važan faktor koji treba dodati.

Moj algoritam će sadržati svih 6 navedenih parametara, plus parametar za Meta heroje, I za kraj još jedan novi parametar koji je isto bitan jeste sam Rank igrača. Jednostavno neki

heroji koji se koriste u nižim rankovima ne funkcionišu tako dobro u višim rankovima I treba ga uzeti u obzir.

Input sistema:

Input sistema bi predstavljao listu heroja koji su trenutno izabrani I koji su banovani, sa pretpostavkom da je trenutno igračev potez, ako je prazna lista smatra se da igrač prvi bira.

Output sistema:

Output sistema bi predstavljao listu heroja koji su soritrani po tome koji su najbolji odabiri koje korisnik može izabrati.

Baza znanja:

U bazu podataka potrebno uneti sve heroje, navesti njihove uloge (Role), heroje koji ga kontriraju (counter-pick), heroje sa kojima pravi dobar spoj (combo-pick), meta vrednost (što veći broj predstavlja da je heroj u ovoj verziji jači od ostalih) I upoznatost igrača sa herojem, bazira se na igračevom ranku koji unosi na početku rada programa, a može dodatno da izabere heroje sa kojima je veoma upoznat I polje da li se trenutni heroj više bira u nižim ili višim rankovima (isto zavisi pri unosu ranka samog igrača)

Način funkcionisanja:

Način funkcionisanja predstavlja ručno upisana pravila. Pravila će sadržati prioritet u izvršavanju (forward chaining). Na nivou istog prioriteta, output pravila će predstavljati broj, veći broj će predstaviti veći prioritet. Konkretno u zadatku, nivo gde se spominje brojna vrednost je meta, upoznatost igrača sa herojem I da li se trenutni heroj više bira u nižim ili

višim rankovima. Na ovom nivou dobiće se vrednost od svakog faktora, konačni bolji rezultat predstavlja sumu navedene 3 vrednosti, gde bolji output predstavlja veća vrednost.

Dodatan nivo kompleksnost donosi uslov inputa. Šta to predstavlja? U zavisnosti od trenutno izabranih heroja u partiji, može se promeniti prioritet koji će se prvo gledati prilikom preporuke.

Primer 1: Izabrano 4 heroja u timu, igrač poslednji bira. Prioritet predstavlja uloga heroja u timu. Ako se desi scenario da nedostaje jedna od uloga, skup preporuke se smanjuje na sve heroje koji su sa zadatom ulogom, nakon toga ulančavanje se nastavlja.

Primer 2: Izabrano 0 heroja, igraču su tad u opticaju svi heroji I sve uloge, tada se tad prvi stepen zanemaruje, gde do izražaja dolazi heroji koji su meta (jači od ostalih).

Primeri rezonovanja:

```
rule "Check if role is missing and suggest it if so"
when
   $myTeam: List()
   $roles: Set() from accumulate(Hero($role: role) from $myTeam,
collectSet($role))
   $missingRole: String() from ["JUNGLER", "MID", "CARRY",
"TANK", "SUPPORT"]
   not Hero(role == $missingRole) from $myTeam
then
   boolean added = false;
   ableToSort.setValue(false);
   for (Hero hero : allHeroes) {
       if (hero.getRole().equals($missingRole) &&
!pickedHeroes.contains(hero.getLocalized_name()) &&
!recommendedHeroes.contains(hero)) {
           hero.addScore(100);
           recommendedHeroes.add(hero);
           added = true;
       }
   }
   System.out.println("Recommended picks: ");
   for (Hero hero : recommendedHeroes) {
       System.out.println("\t" + hero.getLocalized_name() + "(" +
hero.getRole() + "), Score: " + hero.getScore());
   }
   if (added) {
       flag.setValue(true);
       System.out.println("These picks were recommended because
your team does not have the role: " + $missingRole);
   }
end;
```

When in team there is some role/roles then don't recommend that/those roles. (Pojašnjenje: Izabrane su sve uloge osim supporta u timu, preporuka se sužava na sve heroje koji su support uloge). *Ovo predstavlja uopšten primer koji će morati posebno da se napiše za svih pet uloga.*

```
rule "Cut some heroes from recommended list with counter picks"
when
   eval(!counterPicks.isEmpty() && flag.getValue())
then
   List<Hero> newList = new ArrayList<>();
   for (Hero hero : recommendedHeroes) {
       if (counterPicks.contains(hero.getLocalized_name())) {
          hero.addScore(200);
           newList.add(hero);
       }
   }
   if (!newList.isEmpty()) {
       System.out.println("\nBest picks list (COUNTERS):");
       for (Hero hero : newList) {
          System.out.println("\tHero name: " +
hero.getLocalized_name());
          for (String s : hero.getGoodAgainst()) {
               for (Hero oppositeHero: oppositeTeam) {
                    if
(oppositeHero.getLocalized_name().equals(s)) {
                         System.out.println("\tCounters: " +
oppositeHero.getLocalized_name());
                    }
               }
          }
         System.out.println("\tScore: " + hero.getScore());
          System.out.println("");
       }
       ableToSort.setValue(true);
       System.out.println("These picks were recommended because
you can counter enemy hero by picking one of these, which is much
better than any other picks");
   }
end;
```

When in opponent team is hero which has counter pick heroes and team is missing role which can be that over pick then recommend the most that hero. (Pojašnjenje kroz primer: U protivničkom timu je heroj Dazzle, njegov kontra heroj je Axe. Axe je uloge tank, tank fali u timu igrača, zbog toga igraču se predlaže najviše da uzme tog heroja). *Ovo predstavlja uopšten primer koji će morati posebno da se napiše za svaki slučaj, ima ih ukupno oko 15-20.*

```
rule "Suggest the most if combo pick is available"
when
   eval(!combos.isEmpty() && flag.getValue())
then
   List<Hero> newList = new ArrayList<>();
   for (Hero hero : recommendedHeroes) {
       if (combos.contains(hero.getLocalized_name())) {
          hero.addScore(100);
           newList.add(hero);
       }
   }
   if (!newList.isEmpty()) {
       System.out.println("\nBest picks list (COMBOS):");
       for (Hero hero : newList) {
           System.out.println("\tHero name: " +
hero.getLocalized_name());
           for (String s : hero.getCombo()) {
               for (Hero heroMyTeam : myTeam) {
                    if (heroMyTeam.getLocalized_name().equals(s))
{
                        System.out.println("\tCombo with: " +
heroMyTeam.getLocalized_name());
                    }
               }
           }
           System.out.println("\tScore: " + hero.getScore());
           System.out.println("");
       }
       ableToSort.setValue(true);
       System.out.println("These picks were recommended because
you can create a combo by picking one of these, which is much
better than any other picks");
   }
end;
```

When in team is hero which has a combo with some other hero and that hero role is missing in team then recommend the most that hero. (Pojašnjenje kroz primer: U igračevom timu je heroj

Juggernaut, njegov heroj za combo je Crystal Maiden, Crystal Maiden je uloge support I support uloga fali u timu, tada se igraču najviše predlaže da izabere tog heroja). *Ovo predstavlja uopšten primer koji će morati posebno da se napiše za svaki slučaj, ima ih ukupno oko 20.*

```
rule "Add meta points to score"
when
     eval(ableToSort.getValue())
then
     System.out.println("\nAdded meta points to heroes: ");
     for (Hero hero : recommendedHeroes) {
          hero.addScore(hero.getMeta());
          hero.addScore(hero.getMeta());
     }
     Collections.sort(recommendedHeroes);
     for (Hero hero : recommendedHeroes) {
           System.out.println("\t" + hero.getLocalized_name() +
" (" + hero.getScore() + ")");
       }
end;
```

Dodavanje meta poena za svakog heroja

```
rule "Add player skills points to score"
when
     eval(ableToSort.getValue())
then
     System.out.println("\nAdded player skills points to heroes:
");
     for (Hero hero : recommendedHeroes) {
          hero.addScore(hero.getPlayer_skills());
     }
     Collections.sort(recommendedHeroes);
     for (Hero hero : recommendedHeroes) {
           System.out.println("\t" + hero.getLocalized_name() +
" (" + hero.getScore() + ")");
       }
end;
```

Dodavanje licnih poena u celoukupan rezultat

```
rule "Add 50 points if hero is in correct rank where played"
when
     eval(ableToSort.getValue())
then
     System.out.println("\nAdded 50 points if hero is played in
correct rank where played: ");
     for (Hero hero : recommendedHeroes) {
          if (hero.getRank_play().equals(player_rank)) {
               hero.addScore(50);
          }
     }
     Collections.sort(recommendedHeroes);
     for (Hero hero : recommendedHeroes) {
           System.out.println("\t" + hero.getLocalized_name() +
" (" + hero.getScore() + ")");
       }
end;
```

Dodavanje 50 poena u celoukupan rezultat ako je rank igrača jednak rangu heroja u u kom se rangu taj heroj najviše igra.

Ovo su predstavljali slučajevi na nižim nivoima, naravno treba implemenitrati I načine davanja bodova za svakog heroja koji je u preporuci finalnoj gde se može desiti pomeranja. Tok programa je da se svaki put sužava skup ponuđenih vrednosti gde se u jednom momentu dobije lista preporučenih heroja. Tada se prolazi kroz sve preporučene heroje I dodaju vrednosti za svaki, vrednosti predstavljaju: u kojoj meri je heroj meta, lične preferencije za heroja tj. Koliko je upoznat sa herojem, potom da li se heroj više igra u nižim ili višim rankovima. Na kraju će se lista preporučenih heroja sortirati po vrednosti, što je veća vrednost to je preporučniji heroj za izabrati I to predstavlja finalni izlaz iz sistema.