Opis in načrt dela

tim.mulej

April 2022

1 Uvod

Za projektno nalogo sem si izbral Wordle s slovenskimi besedami. Moj plan je, da najprej igro repruduciram, kot spletno aplikacijo v ASP.NET okolju, kjer bom za programski jezik uporabil C#. Besede bom shranil v bazi, do katere bom dostopal s programskim jezikom SQL. Nato bom za vse besede ocenil, koliko informacije ti dajo o iskani besedi, in potem izbral tisto, ki ti da največ. Oceno bom konstruiral kot pričakovano vrednost informacije, ki ti jo ta poda.

2 Opis dela

Cilj besedne igre Wordle je, da s čim manj poizkusi uganeš izbrano besedo s petimi črkami. Po vsakem poizkusu ti računalnik pove, katere črke si zadel, in katerih črk iskana beseda nima. Torej, če se ti katera od črk obarva zeleno, to pomeni, da je tudi v izbrani besedi ta črka na tem mestu. Če se ti črka obarva rumeno, pomeni da si črko zadel, vendar je njeno mesto napačno, če pa se črka obarva sivo, pa pomeni, da te črke ni v iskani besedi. Če se ti vseh pet črk obarva zeleno, si zmagal igro.

Ta algoritem za igranje Wordla temelji na tem, da poizkuša z vsakim poizkusom čim bolj zmanjšati število možnih pravilnih odgovorov. Število možnih odgovorov najbolj elegantno opišemo z informacijo, ki je definirana na način

$$I = \log_2 \left(\frac{n}{m} \right),$$

kjer je n število vseh možnih besed in m število še možnih pravilnih besed. Ta način je intuitiven, ker če dobimo na primer 8 informacije iz ene besede, to pomeni da smo število možnih odgovorov zmanjšali za faktor 8. Poleg tega pa je informacija aditivna, torej če nam da prva beseda 6 informacije in druga 4, smo zmanjšali število možnih odgovorov za 10-krat. Ta algoritem za vse besede izračuna pričakovano informacijo, na način

$$E(I) = \sum_{i=1}^{n} p_i \log_2 \left(\frac{1}{p_i} \right),$$

kjer je p_i verjetnost, da se nam bo pojavil i-ti vzorec, na primer rumena, siva, siva, siva, zelena, potem pa izbere besedo z največjo pričakovano vrednostjo.

3 Načrt dela

Najprej sem si moral naložiti programe, v katerih sem delal, kot so MS SQL strežnik in MS SQL Manegement studio za obdelavo podatkovnih baz, potem pa še Visual Studio z paketom ASPNET and web development.

Potem sem iz spletne strani http://bos.zrc-sazu.si/sbsj.html pobral vse besede z petimi črkami, ki sem jih shranil v bazo tako, da je bila vsaka črka v svojem stolpcu. To mi je potem pomagalo pri iskanju besed. Čeprav sem imel v bazi samo besede s petimi črkami, sem moral nekatere izbrisati, ker so vsebovale številke, vezaje in pike.

Moja aplikacija je sestavljena iz štirih glavnih map. Prva mapa se imenuje wwwroot in v njej so shranjene vse datoteke, ki vsebujejo grafične elemente, na primer škatle okoli črk. Druga mapa se imenuje Models, notri imam shranjen razred Beseda, katerega uporabljam kot osnovno podatkovno strukturo. Ta ima 12 atributov. V prvih 10 imam shranjene črke in njihove stopnje, zadnje dva atributa pa vsebujeta povprečno informacijo, ki naj bi jo ta beseda prinesla v danih okoliščinah in dejansko informacijo, ki smo jo dobili iz te besede. Tretja mapa se imenuje Pages, v njej so shranjeni cshtml teksti strani in njihovi controllerji. Četrta mapa se imenuje services v njej imam shranjene vse funkcije, ki jih moja aplikacija počne. Poleg teh glavnih map pa aplikacija vsebuje še Connected Services, Dependencies, Properties, appsettings.json in Program.cs, ki skrbijo, da se aplikacija pravilno zažene.

Največ dela sem vložil v mapo services, v kateri sem ustvaril tri datoteke. V datoteki GetServices sem napisal funkcije, s katerimi sem pobral, prefiltriral in prenesel podatke. Naslednja datoteka GameServices vsebuje vsa pravila igre Wordle. Tukaj sem imel največ problemov s tem, kako pobarvati črke. Paziti sem moral na vrstni red in število ponavljajočih se črk. Saj če uporabnik poizkusi besedo, ki ima dve isti črki in je druga pravilna, se mora prva pobarvati sivo in ne rumeno. Tretjo datoteko pa sem poimenoval BotServices. V njej so funkcije, ki mi pomagajo izračunati optimalno besedo. Tu mi je največ problemov povzročalo sestavljanje wherestringa, s katerim sem hotel zajeti samo besede, ki zadoščajo določenemu vzorcu. Drugi način, ki ne bi zahteval wherestringa, je, da bi za vsako besedo poizkusil, kako se obarvajo vse ostale besede, in potem bi preštel, kolikokrat se vsak vzorec obarvanja pojavi, vendar bi za ta postopek moral za vsako besedo iti skozi celoten seznam besed, kar pa bi porabilo preveč časa. Pri sestavljanju wherestringa sem moral paziti pri besedah, ki vsebujejo več enakih črk. Na primer, če beseda vsebuje dva a-ja, jaz pa bi hotel najti vse besede, ki ustrezajo vzorcu, da je prvi a rumen in drugi siv, bi moral poiskati samo besede z enim a-jem. Če bi bile barve a-jev zamenjani, pa temu vzorcu ne bi ustrezala prav nobena beseda, ker wordle vedno pobarva samo prvo možno črko. Čeprav sem naredil izboljšavo z wherestringom, moja aplikacija še vedno potrebuje veliko časa, da izračuna optimalno besedo, ker je vseh vzorcev obarvanja besede $3^5 = 243$, moram tolikokrat iti skozi seznam besed, za vsako besedo. Moj računalnik zmore predelati približno eno besedo na sekundo, torej bi mu za vse besede vzelo okoli 12000 sekund, kar je malo več kot 3 ure.

Poleg tega, da ti program predlaga optimalno besedo, mislim še narediti, da oceni vsak poizkus uporabnika, s tem da mu pove, koliko je bila pričakovana informacija njegove besede in nato mu še pove, koliko dejanske informacije je pridobil s tem poizkusom.