Szegedi Tudományegyetem

Informatikai Tanszékcsoport

Minőségkinyerés borkóstolási adatokból, web és android alkalmazás fejlesztés

Szakdolgozat

Készítette:
Varga Tamás
Programtervező Informatikus hallgató

Témavezető:
Dr. Csendes Tibor
tanszékvezető egyetemi tanár

Szeged
2015

Tartalomjegyzék

Feladatkiírás

A hallgató feladata egy olyan web és hozzá tartozó android alkalmazás készítése, amely képes borkóstolás során gyűjtött bor értékelések alapján több különböző algoritmus használatával a kóstolók rangsorolására. A web és android alkalmazásoknak képeseknek kell lenniük egymás közötti szinkronizációra, több felhasználó kezelésére valamint a bevitt adatok és az alkalmazás állapotaink tárolására. Az android alkalmazásnak offline módban is használható kell hogy legyen.

Tartalmi összefoglaló

A téma megnevezése ♦ Minőségkinyerés borkóstolási adatokból, web és android alkalmazás fejlesztés

A feladat megfogalmazása ♦ Web és android alkalmazás fejlesztése, amely képes bor értékelések alapján már létező algoritmusok által rangsorolni a borkóstolókat. A mobil alkalmazás felhasználó barát megvalósítása, mely bármilyen körülmény mellet valós idejű adat bevitelt is garantál.

A megoldási mód ♦ A web alkalmazás a következő címekről érhető el:

- `http://bor.tvarga.hu`

Az android alkalmazás letölthető bármelyik fent említett URL -ről a jobb oldali floating menün keresztül, vagy az alábbi közvetlen linken: borkostolas.apk

Alkalmazott eszközök, módszerek ♦

- Web alkalmazás:

  Programozási nyelvek: PHP, JavaScript, MySQL

  Fejlesztői környezetek: SublimeText, PhpStorm

- Android alkalmazás:

  Programozási nyelvek: Java

  Fejlesztői környezetek: Android Studio

- Verziókezelés: GitHub

Kulcsszavak ♦ Android, PHP, JavaScript, Borkóstolás

Bevezetés

Borkóstolók rangsorolásával illetve minőségkinyeréssel valamint borversenyek lebonyolításával már előttem is többen foglalkoztak. Bár még van kutatni való ezen a területen, jól működő algoritmus már több is ismert ilyen jellegű feladatok megoldására. Ezért ennek a szakdolgozatnak nem is újabb algoritmusok kifejlesztése vagy már meglévők továbbfejlesztése a fő célja. Mivel található több olyan web alkalmazás is, mely egy specifikus egyszerű algoritmust vagy talán egy sokat tesztelt komplexebbet használ, mégsem létezek egy közismert, modern és felhasználóbarát alkalmazás, amely egy igényes felhasználói felületen keresztül bemutatná ezen algoritmusok működését lehetővé téve hogy nyomon kövessük akár saját laikus eredményeinket neves szakértőkhöz viszonyítva.

Mivel a mai világban a mobil eszközök egyre nagyobb teret hódítanak, az is célom volt hogy egy olyan mobil (android) alkalmazás társuljon a web alkalmazáshoz mely egy könnyen kezelhető intuitív felületen keresztül lehetővé tegye a rendszer használatát, bizonyos funkciókat még offline üzemmódban is elérhetővé téve, kielégítve napjaink átlagos felhasználóinak igényeit.

A dolgozat elején röviden bemutatom az alkalmazás által használt algoritmusokat. Majd rátérek a weboldal és a mögötte álló rendszerek felépítésére, a hozzájuk társuló fejlesztői környezetek és eszközök rövid ismertetésére. Ezt majd a mobil (android) alkalmazás bemutatás követ, melynél szintén kitérek a rendszer felépítésére illetve a megvalósítást lehetővé tevő eszközökre, erőforrásokra. Logikusan ezt a két alkalmazási felület (web és mobil) összefűzésének megvalósítása követi. Az összefoglalás előtt néhány teszten keresztül bemutatom a rendszer megbízhatóságát. Legvégül pedig kitérek néhány olyan dologra mely segítette a munkámat, mint például verziókezelő rendszerek használata.

1. fejezet

Borkóstoló algoritmusok

Ezen szakdolgozat keretében a CoHITS algoritmus került implementálásra, mely mellé csatlakozott néhány Borkóstolás projekt[1] keretében használt már Szilárd Iván által implementált algoritmus, mint a Hamming, Koszinusz, Precedencia, Összefüggőségi. Ezeknél az algoritmusoknál a PageRank egy hasonlósági mátrixon propagál, ennek elemeit a borkóstolók páronkénti értékeléseinek inverz távolságából kapjuk a hozzájuk illő távolság mértékek alkalmazásával (Hamming távolság, koszinusz távolság, precedencia távolság, összefüggősségi távolság)[1].

1.1. CoHITS

A CoHITS algoritmus a PageRank és a HITS algoritmuson egy kiterjesztett változata [5]. A PageRank algoritmust Sergey Brin és Larry Page fejlesztette ki a Google kereső használatához[2] keresési találatok fontossági rangsorolásnak meghatározása céljából. Tőlük függetlenül Jon Kleinberg egy hasonló koncepcióval állt elő mely ugyan ilyen jellegű feladatot látott el[3].

Legyen X és Y a kóstolók és borok. Ugyan abból a p^0 értékből indulunk ki minden $x_i \in X$ kóstolónál. Ekkor legyen $w'\left(\overrightarrow{x_i y_j}\right)$ az az értékelés amit y_j bor kapott az x_i kóstolótól és legyen $w\left(\overrightarrow{x_i y_j}\right) = w'\left(\overrightarrow{x_i y_j}\right) / \sum_{j \in Y} w'\left(\overrightarrow{x_i y_j}\right)$ a normalizáltja. Továbbá legyen q_j^0 érték (az y_j borra) az értékelések átlaga. Majd definiáljuk a $w\left(\overleftarrow{x_i y_j}\right)$ az alábbi képen: tegyük fel hogy y_j bort l különböző kóstoló kóstolta és legyen

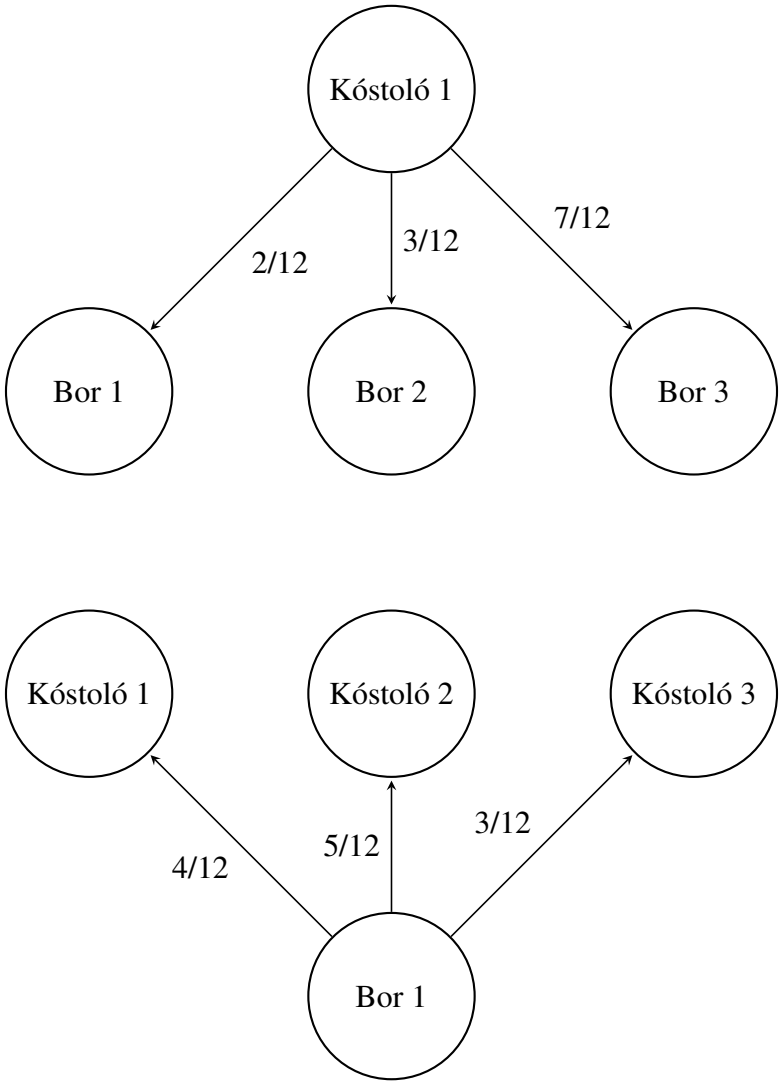
$$D = \sum_{i \in X} |q_j^0 - w'\left(\overrightarrow{x_i y_j}\right)|, \tag{1.1}$$

az összegek különbsége az átlagtól az y_j borra tekintve. Végül, legyen

$$w\left(\overleftarrow{x_i y_j}\right) = \frac{|\sum_{i \in X} |q_j^0 - w'\left(\overrightarrow{x_i y_j}\right)||}{(l-1)\, D} \tag{1.2}$$

Ekkor $\sum_{i \in X} w\left(\overleftarrow{x_i y_j}\right) = 1$ tehát minden $w\left(\overleftarrow{x_i y_j}\right)$ súlyra lehet úgy tekinteni mint egy átmeneti valószínűség y_j-ből x_i-be. Az 1.1 ábra a súlyok kiszámítására mutat egy konkrét példát.

A két kóstoló közti súly x_i és x_j úgy definiálható mint egy rejtett átmeneti valószínűség 1 által definiálva. Ekkor a $p = W_{xp}$ HITS egyenlet $p = (p_1, p_2, ..., p_m)$ megoldás megadja a kóstolók rangsorát.



1.1. ábra. A gráf súlyai amikor a Kóstoló 1 a 20, 30 és 70 értékeléseket adott a Bor 1, Bor 2 és Bor 3 borokra, valamint a Bor 1 a 20, 30 és 70 értékeléseket kapott a Kóstoló 1, Kóstoló 2 és Kóstoló 3 aktól.

Tehát látható, hogy az így elő állított algoritmust alkalmazni lehet egy borkóstolási gráfra kóstolók rangsorolásának céljából az ő képességeik és szakmai hozzáértésük alapján. Általánosságban az figyelhető meg, hogy az így előálló rangsor jobban teljesít mint más egyszerű statisztikai algoritmusok. Ki tudja szűrni a hozzá nem értő kóstolókat[5].

1.2. Hamming

A Hamming-távolságot (1.3) többnyire vektorok, karaktersorozatok valamint bitminták közötti eltérések kimutatására használják. A Hamming-távolság csak azt mondja meg, hogy hány helyen nem illeszkedik az egyik minta a másikra. Legyen S_1 és S_2 két azonos hosszú minta.

$$D_{Hamming}\left(S_1, S_2\right) = \sum_{p=0}^{|S_1|} S_1\left(p\right) \neq S_2\left(p\right) \tag{1.3}$$

Ez a távolság nem lesz megfelelő mérték vizsgálathoz mivel nem túl informatív.

1.3. Koszinusz

Legyen π^1 és π^2 két permutáció-vektor. Ekkor ezek koszinusz távolságát az (1.4) képlet szerint számíthatjuk ki, ahol $||\cdot||$ az euklideszi normát jelöli.

$$D_{cos}\left(\pi^1, \pi^2\right) = 1 - \frac{\pi^1 \cdot \pi^2}{||\pi^1|| \cdot ||\pi^2||} \tag{1.4}$$

1.4. Precedencia

A precedencia távolság meghatározásához definiálnunk kell először a precedencia mátrix fogalmát. Egy π, n hosszúságú permutáció precedencia mátrixa egy olyan bináris elemekből álló $n \times n$-es P mátrix, melyben $P_{\pi_i \pi_j} = 1, i < j$ egyébként $P_{ij} = 0$, azaz a permutáció elemeinek sorrendjét kódoljuk egy bináris mátrixszal. A precedencia távolságot két π^1 és π^2 permutáció között (P_1 és P_2 a nekik megfelelő precedencia mátrix) a precedencia mátrixuk egyező nem nulla elemeinek számából számíthatjuk ki (1.5) szerint.

$$D_{prec}\left(\pi^1, \pi^2\right) = \frac{n\left(n-1\right)}{2} - \sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}^1 P_{ij}^2 \tag{1.5}$$

1.5. Összefüggőségi

Az összefüggősségi távolság a szomszédos elemek azonos előfordulásából számítható. Legyen a két permutációnk π^1 és π^2, n hosszúságúak, és a szomszédos elemek azonos előfordulásának száma n_{adj}.

Szemléletesen: Legyen N_π a π permutáció összefüggőségi mátrixa. Kezdetben minden eleme legyen 0, majd a π permutáció minden i elemére az $N_{pi}\left(\pi_i, \pi i + 1\right) = 1$. A π^1

és π^2 permutációra kiszámoljuk az N_{π^1} és N_{π^2} mátrixokat. n_{adj} az N_{π^1} ésN_{π^2} mátrixban azonos cellában lévő 1-esek száma.

Ekkor az összefüggősségi távolság (1.6) szerint számítandó.

$$D_{adj}\left(\pi^1, \pi^2\right) = n - n_{adj} - 1 \tag{1.6}$$

[1]

2. fejezet

A weboldal

2.1. Iterációk

2.2. PHP

2.3. JavaScript

2.4. Grafikonok

2.4.1. Google Charts

2.5. Statikus tartalom

2.5.1. index.html - Főoldal

2.5.2. borkostolasEredmenyek.html - Eredmények

2.5.3. modszerek.html - Módszerek

2.5.4. kapcsolat.html - Kapcsolat

2.6. Dinamikus tartalom

2.6.1. Regisztráció

2.6.2. Bejelentkezés

2.6.3. demo.html - Demó

3. fejezet

A mobil alkalmazás

Android alkalmazás fejlesztés

3.1. Android

Android

4. fejezet

A weboldal és a mobil alkalmazás összefűzése

A weboldal és a mobil alkalmazás összefűzése

5. fejezet

Tesztelés

5.1. Regisztrációs és bejelentkeztető rendszer

5.2. Demo adatkezelésének ellenőrzése

5.3. Algoritmusok ellenőrzése kis adatokon

5.4. Algoritmusok ellenőrzése ismert eredményekkel

6. fejezet

Összefoglalás

7. fejezet

Egyebek

7.1. Verziókezelés

Verziókezelés röviden

Ez alatt több verzióval rendelkező adatok kezelését értjük. Leggyakrabban a szoftverfejlesztésben használnak verziókezelő rendszereket fejlesztés alatt álló dokumentumok, tervek, forráskódok és egyéb olyan adatok verzióinak kezelésére, amelyeken több ember dolgozik egyidejűleg vagy amelyen több fizikai helyről dolgoznak.

7.1.1. Git

A **Git** egy nyílt forráskódú, elosztott verziókezelő szoftver, amely a sebességre helyezi a hangsúlyt melyet eredetileg Linus Torvalds fejlesztette ki a Linux kernel fejlesztéséhez. Az elosztottság abban valósul meg, hogy a Git minden fejlesztő helyi munkaváltozatában rendelkezésre bocsátja a teljes addigi fejlesztési történetet, és a változtatások másolása mindig két repository között történik. Ezeket a változtatásokat mint külön ágakat importálják és összefésülhetőek, hasonlóan a helyben létrehozott fejlesztési ágakhoz. Ez azért jó, mert így minden munkamásolat egy teljes értékű repository teljes verziótörténettel és teljes revíziókövetési lehetőséggel, amely nem függ a hálózat elérésétől vagy központi szervertől.

7.1.2. GitHub

A GitHubot eredetileg Tom Preston-Werner, Chris Wanstrath és PJ Hyett hozta létre a kódmegosztási procedúra szimplifikálásának érdekében, mára a világ legnagyobb kód távoli kód repository szolgáltatójává nőtt. [4]

7.1.3. A választás

Már a kezdetektől nyilvánvaló volt számomra, hogy valamilyen verziókezelő rendszer használata elengedhetetlen lesz, mivel gyakran nem csak egy specifikus munkaállomásról szoktam dolgozni. Ez nem csak azért jó, mert nagyobb flexibilitást nyújt térben és időben, hanem azért is mert valamilyen szinten szimulálja azt amikor valaki egy projekt munkában egy közös kódbázison dolgozik. Korábbi munkáim és tanulmányaim során már néhány ilyen rendszerrel is megismerkedtem, többek között SVN, Git, Mercurial. A szakdolgozathoz tartozó alkalmazások készítéséhet végül a Git -et választottam mint verziókezelő rendszer, melyhez távoli repository szolgáltatónak a GitHub társult. A választás fő támpontjai között talán a felhasználó barátságot és könnyű automatizálhatóságot említeném meg.

7.2. Környezetek

SublimeText - linterek

PhpStorm

Android Studio

Külön fájlban elkészített grafika beillesztését a

8. fejezet

Függelék

8.1. A program forráskódja

Nyilatkozat

Alulírott Varga Tamás programtervező informatikus BSc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Tanszékcsoport Számítógépes Optimalizálás Tanszékén készítettem, programtervező informatikus BSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Informatikai Tanszékcsoport könyvtárában, a helyben olvasható könyvek között helyezik el.

Szeged, 2015. április 27. .....

aláírás

Irodalomjegyzék

[1] Borkóstolás projekt - http://www.inf.u-szeged.hu/ tnemeth/osa/

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, vol. 30, no 1, 107-117, 1998

[3] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, vol. 46, no. 5, 604-632, 1999

[4] GitHub - https://github.com/about

[5] London András és Csendes Tibor. HITS based network algorithm for evaluating the professional skills of wine tasters