Tartalom-alapú tantárgyajánló-rendszer létrehozása

Nyiri András (HH4NO1), Szepes Nóra (N2WIN2) Konzulens: Gézsi András Mesterséges Intelligencia házi feladat 2013/2014/1

Tartalom

Specifikáció	. 3
Program működése	. 4
Az algoritmus működése	. 7
Eredmények értékelése	. 0

Specifikáció

A feladatunk egy tartalom-alapú tantárgyajánló-rendszer létrehozása. A tartalom-alapú ajánlórendszerek az egyes termékek tulajdonságai alapján egy profilt állítanak fel a termékekről, majd ezen tulajdonságok alapján meghatározzák a termékek egymáshoz viszonyított hasonlóságát. Az ajánlás folyamata során a felhasználó kiválaszt néhány terméket, illetve kiválasztja, hogy az adott tulajdonságok milyen súllyal szerepeljenek az ajánlás során, majd a rendszer a kiválasztott termékekhez hasonló tulajdonságú termékeket javasol. A programunkhoz az adatbázist a www.vik.bme.hu/kepzes/targyak honlapról nyertük.

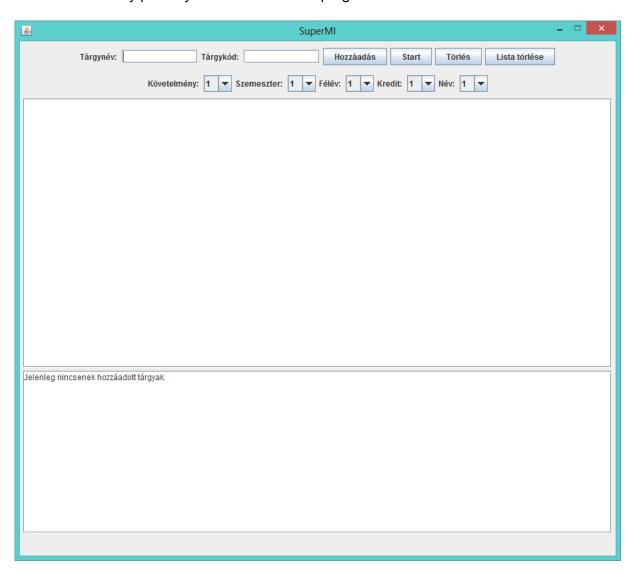
A mai internetes marketing nagyban épít az ajánló rendszerekre, mivel a felhasználókat úgy lehet hatékonyan egy adott holnapon huzamosabb ideig tartani, hogy ha folyamatosan találnak számukra releváns tartalmat. Ez a feladatuk ajánlórendszereknek. A felhasználó előzetes tevékenységeit, szokásait figyelembe véve próbálnak releváns tartalmat szolgáltatni. Így aztán ez egy olyan fontos, központi funkció amely nagyban befolyásolhatja az adott honlap látogatottságát vagy egy szolgáltatás népszerűségét.

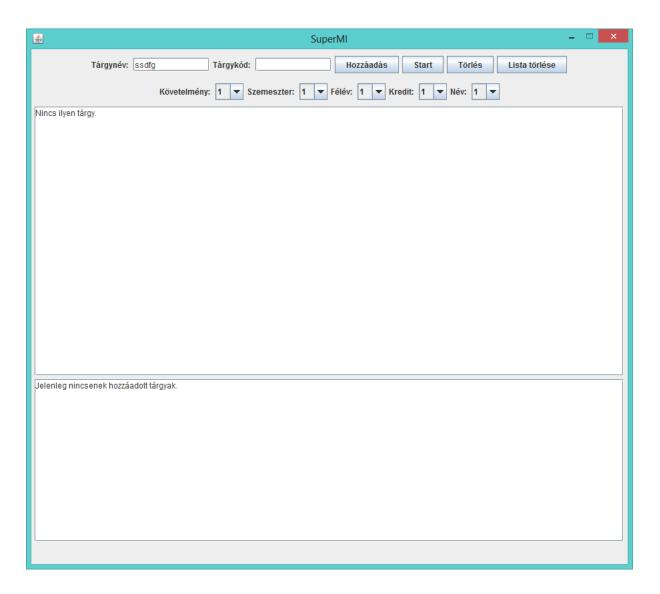
A legnagyobb és legnépszerűbb szolgáltatások, ahol ilyen ajánló rendszerek működnek, pl. a YouTube, Amazon, Facebook. Az alábbi képen pl. látható, hogy az Amazon megjegyezte, hogy megtekintettem az Effective Java c. könyvet és tőle jobbra a "Similar items avaliable…" címszó alatt ajánlott ehhez hasonló könyveket.



Program működése

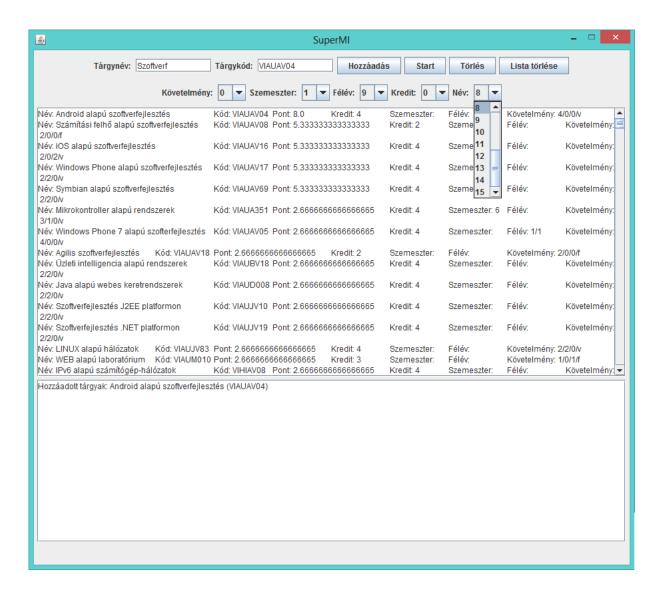
A program elindulásakor ellenőrzi, hogy megvan-e az adatbázis fájl. Az adatbázist kiszerializálva tároljuk egy savedTargyak.txt fájlban. Ha a program megtalálja a fájlt, abból betölti az adatokat, egyébként pedig egy HTML parserrel (Parse HTML osztályban JSoup parserrel) letölti az adatokat a fentebb megadott honlapról és kiszerializálja őket. Ezután a Window osztály példányosításával elindul a program GUI része.





Maga a GUI 3 fő részből áll. Az első a két beviteli mezőt, és a gombokat tartalmazza. A két beviteli mező segítségével tudunk főbb interakciókat végrehajtani. Ha a Tárgynév mezőbe beleírunk valamit, és entert nyomunk, akkor a program az adatbázisból keres olyan tantárgyakat, amelyek nevében benne van a keresett kifejezés. Találat esetén kilistázza a fentebbi mezőbe, különben pedig odaírja, hogy nincs ilyen tárgy.

Ha a tárgykód mezőbe beillesztjük a kívánt tárgykódját, akkor a hozzáadással tudjuk kiválasztani, hogy milyen tantárgyakhoz hasonló tárgyakat szeretnék kiválasztani. Amíg nincs hozzáadott tárgy (alsó mező), addig nem indíthatjuk el az ajánlást sem. Amint hozzáadunk egyet, megjelenik a lenti listában. A tárgyak paramétereit a Targy osztályban határozzuk meg, az adatbázist illetve a kiválasztott tárgyakat pedig két Targyakat tartalmazó ArrayListben tároljuk. A hozzáadott tárgyat természetesen törölni is lehet, a kívánt tárgykód beírásával és a törlés gomb megnyomásával. A lista törlése gomb hatására pedig az összes kiválasztott tárgyat törölhetjük.



A második fő rész a két mezőt tartalmazza, ahova kiírjuk a az interakciók eredményeit. A harmadik pedig a súlyok meghatározására szolgáló JComboBoxokat tartalmazza. Alap helyzetben beállítottuk az 1-es súlyt minden attribútumhoz. Az attribútumokat az egyes tárgyhonlapok tetején lévő táblázat alapján választottuk. A súlyokat 0-tól 15-ig terjedő skálán választhatjuk ki. A start gomb lenyomására a SimilarityFinder osztály megkapja a kiválasztott tárgyak listáját, illetve a kiválasztott súlyt (Suly osztály egy példányának formájában), és ez alapján a kapott pont alapján csökkenő sorrendben kijelzi a tantárgyakat, legfelül azt, amit leginkább ajánl.

Az algoritmus működése

Az algoritmus a SimilarityFinder nevű osztályban van implementálva. Ebben az osztályban kell meghívni a kívülről látható (publikus) findSimilar() metódust, aminek paraméterként meg kell adni a kiválasztott tárgyak listáját és a kívánt súlyozást. A metódus egy pontszám szerint csökkenő sorrendbe rendezett tárgy listát ad vissza.

A metódus futásának kezdetén elvégzi a részletes tárgyadatok kigyűjtését az inicializáló metódus meghívásával. Ez a fent már említett metódus annyit csinál, hogy megnézi, hogy szerializálva rendelkezésre állnak-e a tárgyak és ha nem akkor a parser segítségével letölti az internetről őket (ez internet kapcsolattól függően akár 5 percig is eltarthat). Ez után következik a tárgyak hasonlóság alapján történő pontozása, mivel az algoritmus pontszámok összegzésén alapuló hasonlóság mérést alkalmaz. A következő paraméterek vannak itt figyelembe véve: tárgynév, követelmények (félévközi, vizsgás, aláírás), szemeszter sorszáma, kreditszám. Minden egyes paraméter esetében legfeljebb 1 pont adható a teljes hasonlóság esetén, illetve ez a pontszám be van szorozva az adott paraméterhez tartozó súllyal is. Részleges hasonlóságnál az algoritmus elég szigorúan "büntet", az 1 pontnak ilyenkor valamilyen töredékét adja csak meg.

Példa, tárgynév súlyozása:

```
1. for (Targy t : mFullTargyak) {
2.
                       double sum = 0;
3.
                       //----név súlyozása-----
4.
                       double nevNum = 0;
5.
                       List<String> kivTargyNev = Arrays.asList(targy.mNev.split(" "));
6.
                       List<String> tmpTargyNev = Arrays.asList(t.mNev.split(" "));
7.
                       for (String s1 : kivTargyNev) {
8.
                            for (String s2 : tmpTargyNev) {
9.
                                 if(!s1.equalsIgnoreCase("és")){
10.
                                       if(s1.equalsIgnoreCase(s2)){
11.
                                            nevNum += (1.0/kivTargyNev.size());
12.
                                       }
13.
                                 }
14.
                            }
15.
                       }
16.
17.
                       sum += suly.mSNev * nevNum;
```

Magyarázat:

A sum változó tárolja az éppen aktuális tárgynak az összes pontszámát. Majd a ciklusban a tárgynévben levő összes szót összehasonlítja az összes többi tárgynévben levő összes szóval. Az "és" szót nem tekintjük hasonlóság növelő tényezőnek, azonban ha egy szó hasonlít akkor az összehasonlítandó (kiválasztott) tárgy nevében levő szavak számának az inverzét adjuk hozzá a pontokhoz. Így garantálható, hogy nem lépi át az adott paraméterre adott pontok száma az 1-et. Ezt a pontszámot tehát még beszorozzuk a súllyal, majd hozzáadjuk a kiválasztott tárgy összes pontszámához.

Az algoritmus végén rendezzük pontszám szerint a tárgylistát majd visszatérünk vele:

```
    Collections.sort(mFullTargyak);
    return mFullTargyak;
```

A rendezést megvalósító komparátor:

```
    @Override
    public int compareTo(Object t) {
    Targy tmp = (Targy)t;
    return (int) (1000*(tmp.mSumSuly - mSumSuly));
    }
```

Eredmények értékelése

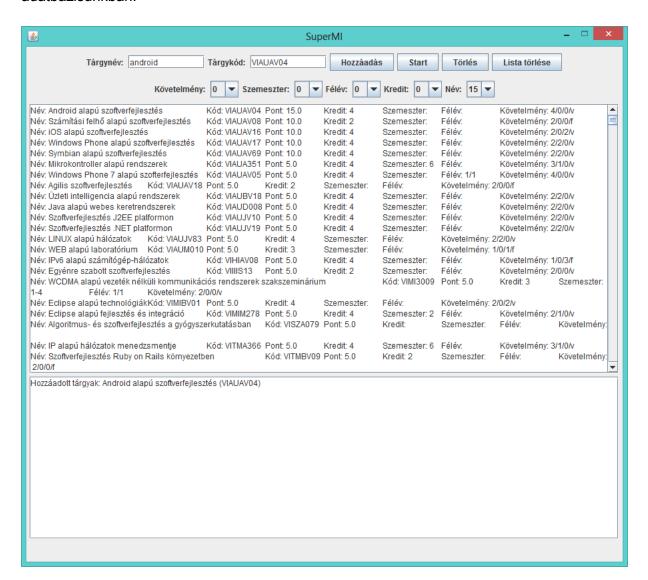
Csak név súllyal való tesztelés

Kiválasztott tárgy:

VIAUAV04, Android alapú szoftverfejlesztés

Elvárás:

A legtöbb pontszámot nyilvánvalóan a kiválasztott tárgy kapta, és alatta olyan tárgyak vannak, amikben a nevében vagy az "alapú" vagy a "szoftverfejlesztés" szó szerepel. Azonos pontszámmal rendelkező tárgyak között az dönt, hogy melyik van előrébb az adatbázisunkban.



Eredmény értékelése: Sikeres, hiszen hasonló nevű tárgyakat keresünk, így ha már egy szó egyezik a tárgyak nevében, akkor az már hasonlónak minősül jelen esetben. Nyilván minél több szó egyezik, annál nagyobb pontszámot fog kapni az adott tárgy.

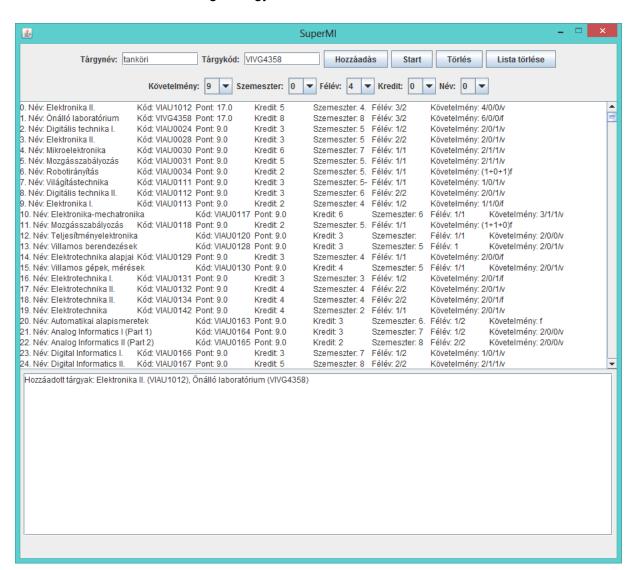
Követelmény és félév közös tesztelése

Kiválasztott tárgy:

VIAU1012, Elektronika II. VIVG4358, Önálló laboratórium

Elvárás:

Azért ezt a két tárgyat választottam, mert félévnél mind a kettő 3/2-es, és nincs is más 3/2-es tantárgy, viszont ezeknek különböző a követelménye, mert az egyik vizsgás, a másik pedig félévközis. Elvárásaim szerint az első két helyen azonos pontszámmal kellene elhelyezkedniük a kiválasztott tantárgyaknak. A további helyen pedig össze-vissza jönnének mind a félévközis, mind a vizsgás tárgyak.



Eredmény értékelése: Sikeres, az első két kapott tárgy tényleg a két kiválasztott, utána pedig azonos pontszámmal tényleg felváltva jönnek mind félévközis, mind vizsgás tantárgyak.

Kredit és szemeszter együttes előfordulása

kiválasztott tárgy: VIAU0164, Analog Informatics I (Part 1)

Kredit:3, szemeszter:7

Ugyan ilyen kredit és szemeszterszámmal hasonló tárgyak:

VIAU0166, Digital Informatics I.

VIFO4054, Folyamatirányítás I. laboratórium

VIMH4050, Méréstechnikai laboratórium

VIVG4043, Laboratórium II. (Energiaátalakító rendszerek)

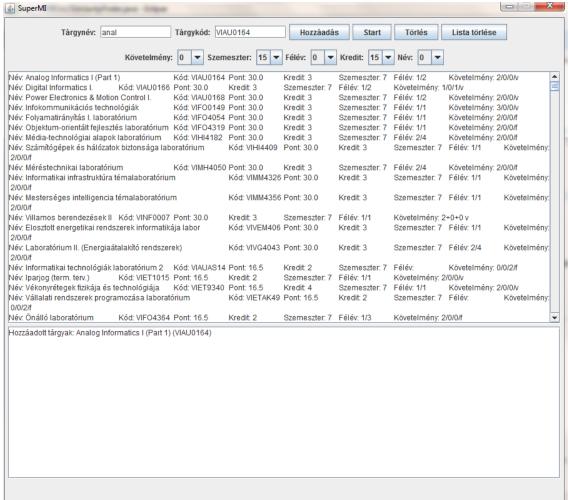
1. teszt

Helyes működés vizsgálata

Elvárás:

A lista elejére vonatkozó elvárás, hogy egyformán a legmagasabb pontszámmal szerepeljenek a hasonló tárgyak.

Teszt eredmény:



Eredmény értékelése: Sikeres mivel az elvárt tárgyak valóban a lista élén szerepelnek maximum pontszámmal.

2.Teszt

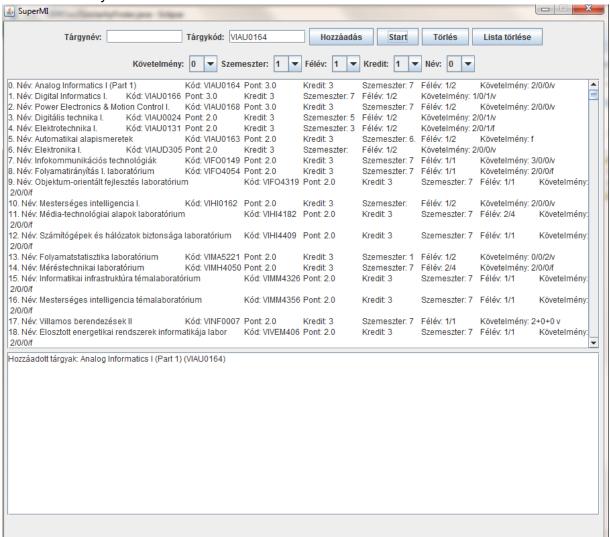
Torzítás vizsgálata

A két paramétert 1-1 súllyal véve és hozzájuk véve egy 3. paramétert ismét 1 súllyal a 3. paraméter ne befolyásolhassa a lista legtetejét érdemben.

Elvárás:

Legalább az első 20 elem között szerepeljenek a hasonló tárgyak.

Teszt eredmény:



Eredmény értékelése:

Tárgykód	Helyezés
VIAU0166	1.
VIFO4054	9.
VIMH4050	14.
VIVG4043	20.

Tehát mindegyik benne van az első 20 találatban, ami megfelel a teszt elvárásainak.