SZÖVEGOSZTÁLYOZÁSI MÓDSZEREK A WEKA ADATBÁNYÁSZATI SZOFTVER SEGÍTSÉGÉVEL

SUBECZ ZOLTÁN

Összefoglalás

A tanulmányban a Weka adatbányászati szoftver használata és a szövegosztályozás alapelvei kerülnek bemutatásra. Egy gyakorlati példán keresztül, amiben Internetről letöltött 4000 db ingatlanhirdetési szöveget dolgoztam fel, több szövegosztályozási módszert megvizsgáltam. Voltak olyan módszerek, amelyekhez a Weka beépített algoritmusát használtam fel, és előfordultak olyanok is, amelyekhez saját programot készítettem. Több módszert is részletesen elemeztem a paraméterek beállításának változtatásával. Az egyes módszerek eredményeit összehasonlítottam az osztályozási pontosság és a futási idő szerint. A feladatokhoz a programokat Java nyelven írtam meg.

Kulcsszavak: *szövegosztályozás, információkinyerés, adatbányászat, szövegbányászat, mesterséges intelligencia, programozás, Java programozási nyelv*

Text classification methods with WEKA data-mining software

Abstract

In my work I presented the usage of the Weka data-mining software and the principles of text classification. I examined several text classification methods with the help of a practical example, where I processed 4000 real estate advertisements from Internet. I used the Weka built-in algorithms for some methods and I wrote programs for the others. I analyzed some methods in detail with different parameters. I compared the results of the methods from the point of view of precision and execution time. I wrote the programs in Java language for the tasks.

Keywords: *text classification, information extraction, data mining, text mining, artificial intelligence, programming, Java programming language*

Bevezetés

Az utóbbi években az informatika egyik leggyorsabban fejlődő részterülete az *adatbányászat* lett. Ez az új tudományág szolgál a nagy mennyiségű adathalmazban rejlő információk automatikus feltárására mesterséges intelligencia algoritmusok alkalmazásával. Az adatbányászat egyik igen fontos részterülete a *szövegbányászat*, amely a strukturálatlan elektronikus szöveges állományokban található információk kinyerését jelenti. Az új alkalmazási lehetőségek közül a *web-bányászat* az egyik legígéretesebb, mivel a világ legnagyobb és leggyorsabban bővülő adattárát az Internetet használja. A web-bányászat célja, hogy az internethez kapcsolható dokumentumokból (honlapok, e-mailek, blogok, fórumok stb.) hasznos információkat gyűjtsön össze és feldolgozza azokat. Az egyik legalapvetőbb szövegbányászati feladat a dokumentumok tartalom szerinti rendszerezésének automatizálása, amelyet *szöveg-osztályozás*nak nevezünk. Ennek célja a szöveges dokumentumok előre definiált halmazból vett kategóriacímkékkel való ellátása. *A téma részletes ismertetése a http://kutat.uw.hu oldalon megtalálható.*

A Weka adatbányászati szoftver
A Weka Java nyelven írt adatbányászati szoftver. Gépi tanulási algoritmusok gyűjteménye adatbányászati feladatokhoz. Nyílt forráskódú programcsomag a GNU (General Public Licence.) szabályainak megfelelően. A programot a http://www.cs.waikato.ac.nz/ml/weka/ oldalról lehet letölteni ingyenesen.

A programot három fő módon lehet használni:
- Grafikus felhasználói felületen
- Parancssoros felületen
- Java programból, beágyazva bármilyen Java programba

A kisebb feladatokhoz a szemléletesebb, grafikus módszert érdemes választani. A nagyobb adatállományokat feldolgozó és nagyobb számítási igényű feladatokhoz a Java programba ágyazott módszert célszerű használni. Én magam is az utóbbi módszert használtam fel a szövegosztályozáshoz.

A felhasznált dokumentumhalmaz
A kutatási témám keretében az Interneten megtalálható ingatlanközvetítői hirdetések szövegét dolgoztam fel. Az Internetről letöltöttem 4000 ingatlanhirdetési oldalt. Ezek egyik fele lesz a tanító dokumentum, a másik fele a teszt dokumentum. A tanító és a teszt halmazban is 1000 családi házas és 1000 panellakásos hirdetés van. Az osztályozó feladata lesz a teszthalmaz minden hirdetéséhez a szövege alapján eldönteni, hogy családi házas vagy panellakásos hirdetés-e.

A Weka programba való beolvasáshoz a következő *elő-feldolgozási lépéseket* tettem meg. (Subecz, 2011)
- Az Internetes HTML oldalról kigyűjtöttem a csak szöveges hirdetési részt szöveges fájlokba (4000 db text fájl)
- Ezekből készítettem újabb 4000 db szöveges fájlt, amelyek már csak az adott dokumentum hasznos szavait tartalmazzák. Az írásjeleket és a felesleges karaktereket eltávolítottam.

A szavakra bontást a következő karaktereknél végeztem el: szóköz, tabulátor, ()[]{}?!/*=<>#&;-
A következő karaktereknél is daraboltam, kivéve, ha előtte és utána is számjegy áll: pont(.) vessző (,) kettőspont (:).
A dokumentum csak a szavakat tartalmazza pontosvesszővel (;) elválasztva.
Így 2 db dokumentumunk lesz: tanito.arff és tesztelo.arff fájlok. Mindkét fájl 2000-2000 ingatlanhirdetés szövegét tartalmazza. Egy hirdetés szövege egy sor a fájlban.

A Vektortér modell és súlyozási változatai
A dokumentumok tárolásához és az osztályozáshoz a szózsák (bag of words) modellt használtam. Ez az adott dokumentumnak csak a hasznos szavait tartalmazza. Ezen dokumentumok szavainak tárolására egy jól bevált modell a *Vektortér modell*, mely egy mátrixban tárolja a dokumentumok szavait. A mátrixnak annyi oszlopa van, ahány egyedi szó helyezkedik el a tanító dokumentumokban. Jelen esetben a 2000 tanító dokumentumban 20 969 egyedi szó található. A mátrix sorainak száma megegyezik a tanító dokumentumok számával (2000). Így a mátrix celláinak száma = 20 969 * 2000 = 41 938 000

A mátrixban a szavak fontosságának megválasztására több lehetőség is adódik. Ezek közül a következő három a leggyakoribb: szógyakorisági TD mátrix, bináris mátrix, súlyozott TD mátrix.
Részletesen: (Tikk, 2007; Subecz, 2011b)

A Vektortér modell elkészítése a Weka szoftverrel
Az előkészített fájlokat beolvastam a Weka programba. A Weka elő-feldolgozási modulja automatikusan el tudja készíteni a bináris és a szógyakorisági Vektortér modellt. A súlyozott TD mátrix elkészítéséhez egy átalakító metódust kellett készíteni.

Szövegosztályozás a Weka programmal
Azokat a bináris osztályozókat, amelyek az osztályozást az M dimenziós vektortér M-1 dimenziós szeparáló, vagy döntési hipersíkkal való kettéosztásával végzik el *lineáris osztályozónak* nevezzük. Egy d dokumentum c kategóriához való tartozását a szerint döntjük el, hogy a vektora a döntési hipersík melyik oldalára esik.

A Weka programcsomagban sokféle osztályozó algoritmus található. Ezek közül a következőket próbáltam ki: Döntési fa alapú, Valószínűség alapú, Legközelebbi szomszédokon alapuló, Szupportvektor-gépek. Ezeken kívül saját programot készítettem a következő módszerekhez: Rocchio osztályozó, Neurális hálózat alapú osztályozó. Az osztályozók az elkészített Vektortér mátrix adatain végzik el az osztályozást.

Beépített osztályozók használata
A szövegosztályozás elméletének és módszereinek részletes leírása megtalálható: (Tikk, 2007; Subecz, 2011a)

Döntési fa alapú osztályozó
A döntési fa tanulása két lépésből áll:

a. annak ellenőrzése, hogy a csomóponthoz tartozó minden tanítódokumentumnak ugyanaz-e a címkéje (c_1 vagy c_2)
b. ha nem, akkor azon szó kiválasztása, amely alapján elvégezzük a partícionálást úgy, hogy az egyes partíciókban a szóhoz tartozó értékek megegyezőek legyenek.

A módszer rekurzív módon addig folytatódik, amíg az egyes levelekben csak azonos kategóriájú tanítóadatok lesznek. Ezzel a kategóriával címkézzük a leveleket.

A programmal az osztályozások eredményességéről sok fajta statisztikai adatot lehet megjeleníteni. Itt csak a pontosság értékeit adtam meg.

$$Pontosság = \frac{Helyesen\ osztályozott\ dokumentumok}{Összes\ dokumentum}$$

Az osztályozó eredménye:

- nem vágott fa: 95.1 % futási idő: 8 perc
- vágott fa: 95,15%

Az osztályozó által elkészített döntési fát karakteresen és grafikusan is meg lehet jeleníteni, ami segít az osztályozási döntések szemléltetésére.

Valószínűség alapú osztályozás: a naiv Bayes módszer
Valószínűségelméleti megközelítésben az osztályozó elkészítésének feladatát a $P(c_j|d_i)$ feltételes valószínűségi értékekre vonatkozó becslésként fogalmazhatjuk meg. Ez az érték megadja, hogy milyen valószínűséggel tartozik a d_i dokumentum a c_j kategóriába. A becslés a Bayes-tétel alapján történik, amely az alábbi összefüggést mondja ki feltételes valószínűségekre:

$$P(c_j|d_i) = \frac{P(d_i)P(d_i|c_j)}{P(d_i)}$$

A d_i dokumentumot ahhoz a c kategóriához rendeljük, amelyikre a $P(c_j|d_i)$ értéke maximális.

Ennek az osztályozónak az eredménye:
91.1 % futási idő: 3 perc
Legközelebbi szomszédokon alapuló osztályozó (k-NN)
A lineáris osztályozókkal ellentétben lokálisan, az adott tesztdokumentumhoz hasonló tanítódokumentum címkéje alapján osztályoz. Az adott dokumentum címkéjét k legközelebbi szomszéd esetén többségi döntés alapján határozzák meg.

Az osztályozást több k esetén is elvégeztem.
Az eredmények:
1NN.: 51,45%; 5NN: 54,25%; 10NN: 58,35%;
20NN: 58,15% 50NN 53,05%
futási idő: 1-1,5 perc

Szupportvektor-gépek (SVM)
A számos más alkalmazási területen is jó eredményeket adó szupportvektor-gépekkel (Support Vector Machine) történő osztályozás szöveges dokumentumok esetén is egyike a leghatékonyabb módszereknek. Nem csak egy olyan hipersíkot keres, ami elválasztja a pozitív és negatív tanító mintákat, hanem ezek közül a legjobbat is kiválasztja, vagyis azt, amelyik a két osztály mintái között éppen „középen” fekszik.

Ennek az osztályozónak az eredménye:
98,1% futási idő: 0,5 perc

Saját osztályozók készítése
Az adatállományok Weka-ba való beolvasása és a Vektortér modell elkészítése után saját osztályozót is készíthetünk.

Rocchio osztályozó
Mivel a Rocchio osztályozót gyakran használják és a Weka programcsomagban nincs megvalósítva, ezért elkészítettem saját programmal.

a. Meghatározzuk a c_j osztályokat reprezentáló centroidokat, amelyek az osztályba tartozó dokumentumvektorok átlagai.
b. A dokumentum és az osztály hasonlósága ekkor meghatározható a dokumentumvektor és az osztály centroidjának koszinusz-távolságaként.
c. Minden dokumentumot ahhoz az osztályhoz rendeljük, amelyikre a koszinusz-távolság nagyobb lesz.

Ennek az osztályozónak az eredménye:
97,9 % futási idő: 40 mp

A Rocchio osztályozó eredményei a Vektortér modell más reprezentációja esetén
bináris súlyozás: 97,95%; előfordulás alapú súlyozás: 95,3%
gyakoriság alapú súlyozás: 95,5%

A Vektortér mátrix méretének csökkentése: csak a legfontosabb oszlopok megtartásával
A Weka programban lehetőség van megadni, hogy a Vektortér mátrixot hány oszloppal készítse el. Ilyenkor azokat az oszlopokat tartja meg, amelyek a legfontosabb szavakhoz tartoznak. Ezzel a vektortér mátrix mérete és így a memóriaigény és a futási idő jelentősen csökkenthető.

Ebben az esetben az osztályozó eredménye:
1000 szó megtartásával 97,45% futási idő: 4 mp
2000 szó megtartásával 97,65% futási idő: 7 mp
5000 szó megtartásával 97,9% futási idő: 10 mp
10000 szó megtartásával 97,9% futási idő: 20 mp

A Vektortér mátrix méretének csökkentése: Stopszó szűréssel és lemmatizálással
Stopszó szűrés és lemmatizálás leírása: (Tikk, 2007; Subecz 2011 3)
Ezen esetekben az osztályozó eredménye:
Stopszó szűrés hatása 98,15% futási idő: 34 mp
Lemmatizálás hatása 97,75% futási idő: 29 mp
Stopszó szűrés és Lemmatizálás együttes hatása: 97,9% futási idő: 26 mp

Neurális hálózat alapú módszer
Bár a Weka programcsomagban vannak neurális hálózat alapú osztályozási lehetőségek, de nagyon lassan futottak le. Ezért készítettem egy saját programot hozzá. A legegyszerűbb bináris osztályozást végző neurális hálózat a **perceptron.** Minden bemenethez (20 969 db) egy súly tartozik. A tanításkor a súlyok értékét változtatjuk. A bemeneti vektor és a súlyvektor skaláris szorzata adja a hálózat kimeneti függvényét. Ennek értéke dönti el, hogy az adott dokumentumot melyik osztályba soroljuk.

A neurális alapú osztályozásnál készítettem egy optimalizálás nélküli megvalósítást és egy optimalizálási megoldást is.

Eredmények optimalizálás nélkül:
A következő beállításokkal futtattam az osztályozót: KezdoErtek = 2 (súlyok kezdeti értéke), EpochMax = 10 (tanulási lépések száma), $\gamma = 1$ (tanulási ráta)
Eredmény: 98,5% Futási idő = 7 mp
Eredmények optimalizálással:
Az osztályozást a következő egymásba ágyazott ciklusokkal futtattam le:
ciklus KezdoErtek = 0-tól 10 ig
    ciklus EpochMax = 1-től 12-ig
        ciklus Gamma = 0,1-től 10-ig 0,1-esével

Lépések száma = 11*12*100=13 200. Vagyis a program 13 200 osztályozót tanított és tesztelt. Együttes futási idő: 10 óra

1. táblázat: Az optimalizálás legjobb eredményei

Pontosság	dbrossz	dbjo	KezdőÉrték	EpochMax	γ
0,9915	17	1983	8	9	6,1
0,991	18	1982	9	7	9,8
0,9905	19	1981	7	11	9,6
0,99	20	1980	1	9	0,7
0,99	20	1980	2	9	1,4

Forrás: saját szerkesztés

Az optimalizálásnál a legjobb eredményt a következő beállítás adta:
Kezdőérték = 8; Epochmax = 9; γ = 6,1; Eredmény = 99,15 %
Az optimalizálás természetesen túltanulást jelent. Más tanító és teszt adatsornál bizonyára nem ez lenne a legjobb megoldás. Lehet, hogy ez csak egy átlagos eredményt adna.

Osztályozók eredményének összehasonlítása

2. táblázat Az osztályozók összehasonlítása:

Osztályozó	**Pontosság** (%)	**Futási idő**
Döntési fa alapú, nem vágott fa	95,1	8 perc
Döntési fa alapú, vágott fa	*95,15*	*8 perc*
naiv Bayes módszer	91,1	3 perc
(1-NN); (5-NN)	51,45; 54,25	1 perc
(10-NN); (20-NN); (50-NN)	58,35; 58,15; 53,05	1,5 perc
Szupportvektor-gépek (SVM)	*98,1*	*30 mp*
Rocchio, tf-idf súlyozás	*97,9*	*40 mp*
Rocchio, bináris súlyozás	*97,95*	*40 mp*
Rocchio, előfordulás alapú súlyozás	95,3	40 mp
Rocchio, gyakoriság alapú súlyozás	95,5	40 mp
Rocchio, 1000 szó megtartásával	97,45	4 mp
Rocchio, 2000 szó megtartásával	97,65	7 mp
Rocchio, 5000 szó megtartásával	*97,9*	*10 mp*
Rocchio, 10000 szó megtartásával	97,9	20 mp
Rocchio, Stopszó szűréssel	98,15	34 mp
Rocchio, Lemmatizálással	97,75	29 mp
Rocchio, Stopszó szűréssel és Lemmatizálással	97,9	26 mp
Neurális hálózat, optimalizálás nélkül	*98,5*	*7 mp*
Neurális hálózat, optimalizálással	*99,15*	*10 óra*

Forrás: saját szerkesztés

Kiemelt eredményeket adott osztályozók:
Neurális hálózat alapú módszer (98,5%; 7mp),
Szupportvektor-gépek (98,1%; 30 mp),
Rocchio osztályozó (98,15%; 34 mp)
Döntési fa alapú osztályozó (95,15%; 8 perc)

Összefoglalás

Dolgozatomban bemutattam a Weka adatbányászati szoftver használatát és a szövegosztályozás alapelveit. Ezután több szövegosztályozási módszert megvizsgáltam. Ennek a széleskörű vizsgálatnak az eredményei jól jellemezték, hogy milyen osztályozási módszereket érdemes alkalmazni a bemutatott típusú szövegeken.

Hivatkozott források

Jurafsky, M. D.,Martin, J. H.(2009): Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition. Stanford, CA : Pearson Prentice Hall.

Tikk D.(2007): Szövegbányászat. Typotex, Budapest

Subecz Z.(2011a): Stopszó szűrés, lemmatizálás hatása és osztályozás a Vektortér modellel, Tudomány napi Konferencia Szolnok

Subecz Z.(2011b): Szöveg feldolgozási lépések a Vektortér modellig, Economica − A Szolnoki Főiskola Tudományos közleményei, 11. szám, 2011. 57-68.o

Szerző:

Subecz Zoltán
főiskolai tanársegéd
Szolnoki Főiskola, Gazdaságelemzési és Módszertani Tanszék
subecz@szolf.hu