Gradus Vol 4, No 2 (2017) 444-452
ISSN 2064-8014

ADATBÁNYÁSZATI SZOFTVER HASZNÁLATA SZÖVEGOSZTÁLYOZÁSHOZ

DATA MINING SOFTWARE FOR TEXT CLASSIFICATION

Subecz Zoltán[1], Nagyné Dr. Csák Éva[2]*

[1]Informatika Tanszék, GAMF Műszaki és Informatikai Kar, Neumann János Egyetem, Magyarország
[2]Idegennyelvi és Továbbképzési Tanszék, Pedagógusképző Kar, Neumann János Egyetem, Magyarország

Kulcsszavak:
szövegosztályozás
információkinyerés
adatbányászat
szövegbányászat
mesterséges intelligencia

Keywords:
text classification
information extraction
data mining
text mining
artificial intelligence

Cikktörténet:
Beérkezett: 2017. szeptember 16
Átdolgozva: 2017. szeptember 25
Elfogadva: 2017. október 15.

Összefoglalás

Dolgozatunkban a Weka adatbányászati szoftver használatát és a szövegosztályozás alapelveit mutatjuk be. Egy gyakorlati példán keresztül, amiben Internetről letöltött 4000 db ingatlanhirdetési szöveget dolgoztunk fel, több szöveg-osztályozási módszert megvizsgáltunk. Voltak olyan módszerek, amelyekhez a Weka beépített algoritmusát használtuk fel, és előfordultak olyanok is, amelyekhez saját programot készítettünk. Több módszert is részletesen elemeztünk a paraméterek beállításának változtatásával. Az egyes módszerek eredményeit összehasonlítottuk az osztályozási pontosság és a futási idő szerint. A feladatokhoz a programokat Java nyelven írtuk meg.

Abstract

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

1. 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. [1] 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

* Kapcsolattartó szerző. Tel.: +36 56 510 300
E-mail cím: subecz.zoltan@gamf.uni-neumann.hu

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.

2. 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. Mi is az utóbbi módszert használtuk a szövegosztályozáshoz.

3. A felhasznált dokumentumhalmaz

Az Interneten megtalálható ingatlanközvetítői hirdetések szövegét dolgoztuk fel. Az Internetről letöltöttünk 4000 ingatlan-hirdetési oldalt. Ezek egyik fele volt 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 tettük meg.

- Az Internetes HTML oldalról kigyűjtöttük a csak szöveges hirdetési részt szöveges fájlokba (4000 db text fájl)
- Ezekből készítettünk újabb 4000 db szöveges fájlt, amelyek már csak az adott dokumentum hasznos szavait tartalmazta. Az írásjeleket és a felesleges karaktereket eltávolítottuk. A szavakra bontást a következő karaktereknél végeztük el: szóköz, tabulátor, ()[]{}?!/*=<>#&;- A következő karaktereknél is daraboltunk, kivéve, ha előtte és utána is számjegy áll: pont(.) vessző (,) kettőspont (:). A dokumentum így már csak a szavakat tartalmazza pontosvesszővel (;) elválasztva. Így 2 db dokumentumunk lesz: a tanító és a kiértékelő fájlok. Mindkét fájl 2000-2000 ingatlanhirdetés szövegét tartalmazza.

4. A Vektortér modell és súlyozási változatai

A dokumentumok tárolásához és az osztályozáshoz a szózsák (bag of words) modellt használtuk. 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 [2].

A *vektortér modellben* a $D=\{d_1,\ldots,d_N\}$ dokumentumgyűjteményt a *szó-dokumentum mátrixszal* (term-document mátrix) reprezentáljuk ($D \in R^{MxN}$), ahol a mátrix d_{ki} eleme a k-adik szó (t_k) relevanciáját reprezentálja az i-edik dokumentumban, d_i-ben. A d_i dokumentumot reprezentáló dokumentumvektort $d_i=<d_{1i},\ldots,d_{Mi}>$- vel jelöljük. A D mátrixban a sorok száma, M megegyezik az egyedi szavak számával. N pedig a dokumentumok száma.

A dokumentumvektorok tehát a mátrix oszlopai lesznek. A mátrix egy sora azon pozíciókban tartalmaz nullától különböző értéket, amelyekhez tartozó dokumentumokban a szó nem nulla relevanciájú. Ez általában ekvivalens azzal, hogy a szó nem szerepel a dokumentumban, de ettől eltérő esetekben is lehet a relevancia nulla.

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: bináris mátrix, szógyakorisági TD mátrix, súlyozott TD mátrix. [3]. Ezek bemutatását a következő alfejezetekben ismertetjük.

Az előkészített fájlokat beolvastuk a Weka programba. A Weka elő-feldolgozási modulja automatikusan el tudja készíteni a bináris és a szógyakorisági mátrixot. A súlyozott TD mátrix elkészítéséhez egy átalakító metódust kellett készíteni.

4.1. Bináris mátrix

A lehetséges mátrixok közül ez a legegyszerűbb. Ebben az esetben azt tároljuk a mátrixban, hogy az adott dokumentumban előfordult-e az adott szó vagy nem. A bináris reprezentációnál a szó-dokumentum mátrixban a d_{ki} értékeket a következő képen határozzuk meg:

$$d_{ki} = \begin{cases} 1, ha\ n_{ki} > 0 \\ 1, ha\ n_{ki} = 0 \end{cases} \quad (1)$$

ahol n_{ki} a t_k szó előfordulásainak száma (másképpen támogatottsága) a d_i dokumentumban. Vagyis, ha egy adott szó előfordul legalább egyszer a dokumentumban, akkor ott a mátrixban az érték 1, különben 0.

4.2. A Szógyakorisági TD mátrix

Ebben az esetben $d_{ki}= n_{ki}$. Vagyis a d_{ki} itt azt adja meg, hogy az adott szó hányszor fordul elő az i. dokumentumban. Egy szónak annál nagyobb a súlya egy dokumentumban, minél többször fordul elő.

4.3. A súlyozott TD mátrix

Gyakorlati szempontból ennek a mátrixnak van a *legnagyobb jelentősége* az előző kettővel összehasonlítva. Egy-egy cellájában arra ad értéket, hogy egy adott szónak mekkora a jelentősége egy adott dokumentumban. A szógyakorisági TD mátrix egy cellája azt tartalmazta, hogy egy adott szó hányszor szerepel egy adott dokumentumban. A súlyozott TD mátrixban ezt az értéket két lényeges szempont szerint módosítjuk.

Az eddig ismertetett súlyozási sémák figyelmen kívül hagyták a dokumentumok hosszát, noha egy 100 szavas dokumentumban egy szó tízszeri előfordulása nyilván sokkal jelentősebb, mint egy 10 000 szavasban. Ezt a szempontot a dokumentumok hossz szerinti normálásával vehetjük számításba. A súlyozott TD mátrixban tehát nem azt vizsgáljuk, hogy egy szó hányszor szerepelt egy dokumentumban, hanem ezt a számot viszonyítjuk az adott dokumentum szavainak a számához.

A dokumentum szavainak számát jelöljük:

$$|d_i| = \sum_{k=1}^{M} n_{ki} \quad (2)$$

akkor a gyakoriság alapú súlyozás: $f_{ki} = n_{ki} / |d_i|$

Az így definiált f_{ki} a t_k szó d_i dokumentumbeli gyakorisága, vagy frekvenciája, amit az angol elnevezés rövidítése alapján *TF-súlyozásnak (term frequency)* is hívnak. A súlyozott TD mátrixnál e mellett még egy szempontot figyelembe veszünk. Mégpedig azt, hogy az adott szó hány

dokumentumban szerepel. Minél több dokumentumban szerepel egy szó, annál kisebb a jelentősége (stopszavak pl. névelők). Mindeddig a dokumentumokban (szótárban) előforduló összes szót egyenrangúnak tekintettük, holott a dokumentumok tartalmi jellemzését illetően a szavak jelentősége eltérő. Például egy dokumentumon belül nem sok tartalmi jelentősége van a névelőknek, mert azok sok dokumentumban előfordulnak. Ha van két szó, amelyik a 100 dokumentumban ugyanannyiszor fordul elő, akkor a két szó közül az a fontosabb, amelyik koncentráltan, kevés dokumentumban, de azokon belül nagy gyakorisággal fordul elő, semmint az, amelyik sok dokumentumban alacsony gyakorisággal.

Jelöljük n_k-val azon dokumentumok számát, amelyben a t_k szó előfordul. Ekkor az n_k/N hányados, amit *dokumentum gyakoriságnak (document frequency, df)* neveznek, jól jellemzi a szó ritkaságát a korpuszban. Ez az érték megadja, hogy mekkora megkülönböztető ereje van, avagy mennyire tekinthető indikátornak a szó jelenléte (és előfordulásainak a száma) a dokumentum tartalmára vonatkozóan.

A súlyozási sémákban inkább a dokumentumgyakoriság inverzével számolnak (idf ,inverse document frequency): $idf(t_k) = \log(N/n_k)$

Így kapjuk a leggyakrabban használt *td-idf súlyozást. (term frequency & inverse document frequency)*: $d_{ki} = f_{ki} * idf(t_k)$

A *tf -idf súlyozás* értéke tehát:

1. Magas lesz azon szavak esetében, amelyek az adott d_i dokumentumban gyakran fordulnak elő, míg a teljes korpuszban ritkán (nagy a megkülönböztető képességük).
2. Alacsonyabb lesz azon szavak esetén, amelyek a d_i dokumentumban ritkábban, vagy a korpuszban gyakrabban fordulnak elő.
3. és kicsi lesz azon szavakra, amelyek szinte a korpusz összes dokumentumában előfordulnak.

A súlyozott TD mátrix egy-egy cellájában arra ad értéket, hogy egy adott szónak mekkora a jelentősége egy adott dokumentumban.

5. 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 próbáltuk ki: *Döntési fa alapú, Valószínűség alapú, Legközelebbi szomszédokon alapuló, Szupportvektor-gépek*. Ezeken kívül saját programot készítettünk a következő módszerekhez: *Rocchio osztályozó, Neurális hálózat alapú osztályozó.* Az osztályozók az elkészített Vektortér mátrix adatain végzik el az osztályozást.

A megadott futási idők a tanítási időt jelentik.

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

$$Pontosság = \frac{Helyesen\ osztályozott\ dokumentumok}{Összes\ dokumentum} \tag{1}$$

5.1. Beépített osztályozók használata

5.1.1. Döntési fa alapú osztályozó

A döntési fa a különböző döntési lehetőségeket ábrázolja, az esetleges következményeket, esélyeket, hasznosságot és erőforrásokat figyelembe véve, attól függően, hogy mire használják. A döntési fa egy olyan faszerkezet, amelyben minden belső csúcs egy értékre vonatkozó ellenőrzést jelöl, a csúcsból kivezető minden él pedig az ellenőrzés egy-egy kimenetének feleltethető meg.

A döntési fán alapuló szövegosztályozó egy olyan fa, amelyben a közbenső csomópontok szavakat reprezentálnak, a csomópontokból kiinduló ágak ellenőrzési feltételeket írnak elő az adott szóra vonatkozóan a tesztdokumentumra, a levelek kategóriákkal vannak címkézve.

A d tesztdokumentum osztályozása a döntési fa csomópontjaihoz tartozó szavak d-beli súlyának vizsgálata alapján, rekurzív módon történik, a dokumentumhoz a levél kategóriacímkéjét rendeljük hozzá.

A döntési fa tanulása két lépésből áll:

a) annak ellenőrzése, hogy a csomóponthoz tartozó minden tanító dokumentumnak ugyanaz-e a címkéje (c1 vagy c2)
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.

Az osztályozás eredménye:
95.1 % futási idő: 8 perc

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

5.1.2. 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:

$$P(c_j|d_i) = \frac{P(c_j)P(d_i|c_j)}{P(d_i)} \quad (2)$$

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

5.1.3. Legközelebbi szomszédokon alapuló osztályozó (k-NN: k-NearestNeighbor)

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. Bináris feladatnál ez azt jelenti, hogy megvizsgáljuk d_i k szomszédjából hány tartozik a c_j kategóriába, és ha ezek aránya elég nagy , akkor c_j-hez rendeljük, egyébként a komplementeréhez. Egycímkés esetben azt a kategóriát választjuk, amelyiknek a legtöbb példánya szerepel a k szomszéd között. Ha több ilyen kategória is van, akkor véletlenszerűen döntünk az egyik mellett. A hasonlóság mérésére a koszinusz-távolságot szokták használni.

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

5.1.4. 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 (jó eredmény!)

5.2. Saját osztályozók készítése

Az adatállományoknak az adatbányászati szoftverbe való beolvasása és a Vektortér modell elkészítése után saját osztályozót is készíthetünk.

5.2.1. Rocchio osztályozó

Mivel a Rocchio osztályozót gyakran használják és a Weka programcsomagban nincs megvalósítva, ezért elkészítettük 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 (jó eredmény!)

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, 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: gyakori, az osztályozást nem befolyásoló szavak elhagyása. Például: és, névelők

Lemmatizálás: A szó szótári alakjának tárolása

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

5.2.2. Neurális hálózat alapú módszer

Bár a Weka programcsomagban vannak neurális hálózat alapú osztályozási lehetőségek, de nagyon lassan futottak le. Ezért készítettünk 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 készítettünk egy optimalizálás nélküli megvalósítást és egy megoldást optimalizálással is.

Eredmények optimalizálás nélkül:
A következő beállításokkal futtattuk az osztályozót:
KezdoErtek = 2 (súlyok kezdeti értéke),
EpochMax = 10 (tanulási lépések száma),
γ = 1 (tanulási ráta)
Eredmény:
98,5% Futási idő = 7 mp (jó eredmény!)

Eredmények optimalizálással:
Az osztályozást a következő egymásba ágyazott ciklusokkal futtattam le:
ciklusKezdoErtek = 0-tól 10 ig
  ciklusEpochMax = 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. A legjobb eredményei az 1. Táblázatban láthatóak.

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

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

5.3. Osztályozók eredményének összehasonlítása

A bemutatott osztályozók összesítését a 2. táblázatban láthatjuk.

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

5.4. 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

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

Köszönetnyilvánítás

Köszönettel tartozunk a kutatás támogatásáért, amely az **EFOP-3.6.1-16-2016-00006 „A kutatási potenciál fejlesztése és bővítése a Neumann János Egyetemen” pályázat keretében valósult meg.** A projekt a Magyar Állam és az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával, a Széchenyi 2020 program keretében valósul meg.

Irodalomjegyzék

[1] Martin Daniel Jurafsky, James H. Martin: Speech and LanguageProcessing: An introductiontonaturallanguageprocessing, computationallinguistics, and speechrecognition. Stanford, CA: PearsonPrentice Hall, 2009.
[2] Subecz Zoltán: A Vektortér-modell használata a Szövegbányászatban. A Pallasz Athéné Egyetem 17. Tudományos Konferenciája a Magyar Tudomány Ünnepe Alkalmából. GRADUS elektronikus folyóirat, VOL 3, NO 2 (2016): AUTUMN (NOVEMBER), pp. 73-79.
[3] Tikk Domonkos: Szövegbányászat. Typotex kiadó, Budapest, 2007.