Budapesti Műszaki és Gazdaságtudományi Egyetem
Matematika Intézet
Sztochasztika tanszék

Toxikus szövegek detektálása

BSc szakdolgozat

Homolya Panni

Témavezető:

Dr. Recski Gábor
TU Wien

Konzulens:

Dr. Kornai András
Egyetemi tanár
Algebra Tanszék, Budapesti Műszaki és Gazdaságtudományi Egyetem

2021

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Recski Gábornak, aki szakértelmével, hasznos magyarázataival és a konzultációk során elhangzott elengedhetetlen tanácsaival, meglátásaival hatalmas segítséget nyújtott szakdolgozatom elkészülésében.

Hálával tartozom továbbá szüleimnek és testvéremnek, akik nélkül ez a szakdolgozat nem jöhetett volna létre. Köszönöm nekik, hogy tanulmányaim folyamán segítettek, türelemmel és megértéssel támogattak, és minden helyzetben mellettem álltak.

Külön köszönöm a páromnak, hogy egyetemi éveim során végtelen kitartással és szeretettel támogatott, és nélkülözhetetlen tanácsaival valamint segítségével hozzájárult, hogy ez a szakdolgozat megszülethessen.

Tartalomjegyzék

Bevezetés

A technológia fejlődésnek köszönhetően, az internetes világ része lett az életünknek. Manapság sokan használnak *Facebook*-ot, *Twitter*-t vagy *Instagram*-ot, hogy kifejezzék gondolataikat és véleményüket. Viszont ahogy szóban, úgy írásban is meg kell válogatnunk a kifejezéseinket. Az interneten, különösen a közösségi médiában jelen van a gyűlöletbeszéd és egyéb toxikus tartalom, aminek felismerésével, osztályozásával és kiszűrésével a kutatók is elkezdtek foglalkozni. A szakdolgozat folyamán a toxikus szövegek detektálás témakörével foglalkozunk és gépi tanulás ismereteinket felhasználva internetes bejegyzéseket osztályozunk.

Az első fejezetben számos toxikus szöveget vagy gyűlölködőbeszédet tartalmazó adathalmazt, köztük a dolgozat során használt adathalmazt is bemutatjuk. Ezenkívül a témában született különböző tanulmányokat, módszereket és eredményeket ismertetjük, továbbá részletes feladatleírást adunk a dolgozat témájáról. A második fejezetben egyrészt az adattudomány elméleti hátterét, másrészt a modell felépítés folyamatát mutatjuk be. Szó esik fontos fogalmakról, mint például a felügyelt tanulásról, tanító, validációs és tesztalmazról, illetve magáról a tanítás és tesztelés jelentéséről is. A fejezetben ismertetjük a logisztikus regressziós modell működését és a modellben használt gradiens módszert, továbbá bemutatjuk a modell kiértékelésénél használt mérőszámokat. Továbbá kifejtjük a szöveges adat előfeldolgozási folyamatait, mint például a tokenizációt, a szótövesítést, illetve az attribútumgenerálást. Ezenfelül bemutatjuk a kód tényleges működését és ismertetjük a különböző paramétereket. Az utolsó fejezetben a hibaelemzés folyamatát, illetve a kvantitatív eredményeinket ismertetjük.

1. fejezet

A feladat hátterének bemutatása

1.1. Kapcsolódó tanulmányok

A toxikus, gyűlölködő szövegekhez számos adathalmaz áll rendelkezésre, amiből néhányat a következőkben ismertetünk.

A munkásságunk folyamán a *Jigsaw Unintended Bias in Toxicity Classification*[1] verseny toxikus adatával dolgoztunk. Az adathoz szorosan kapcsolódik a *Toxic Comment Classification Challenge*[2] verseny adata is, mivel mindkét versenyt a *Jigsaw* hirdetette meg. A könnyebb hivatkozás kedvéért, a *Toxic Comment Classification Challenge*-et nevezzük el *Jigsaw1* adatnak, a *Jigsaw Unintended Bias in Toxicity Classification* pedig *Jigsaw2*-nek. Az említett adatok részletesebb bemutatását a 1.2 részben tárgyaljuk.

Egy másik adathalmaz a *Banchmark Dataset*[3], amit gyűlöletbeszéd vizsgálatához alkalmaznak. Az adatot *Gab*-ről[4] és *Reddit*-ről[5] gyűjtötték, amiket manuálisan címkéztek fel a *Mechanical Turk* dolgozók, gyűlöletbeszéd, illetve nem gyűlöletbeszéd címkéket adva.

Szintén gyűlölködő szövegeket tartalmaz a *Crowdflower* adathalmaza.[6] Az adat *Twitter* bejegyzésekből áll, amikről 3-3 személy döntött adott rekordnál, hogy melyik kategóriába sorolják őket: gyűlölködő szöveg, offenzív szöveg vagy egyik sem.

Az előző adatokat felhasználva számos cikk született, amik különböző

[1] `https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification`

[2] `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`

[3] `https://github.com/jing-qian/A-Benchmark-Dataset-for-Learning-to-Intervene-in-Online-Hate-Speech`

[4] `https://gab.ai`

[5] `https://www.reddit.com`

[6] `https://data.world/crowdflower/hate-speech-identification`

módszerekkel különböző problémákat, feladatokat vizsgálnak. A következőkben néhány ilyen cikket mutatunk be.

A szakdolgozatban használt adatot a legtöbb cikk, mint például a [1], a modellbeli véletlen torzítás minimalizálására alkalmazza. A torzítások minimalizálásával, minél pontosabb eredményeket szeretnének kapni a kiértékelés folyamán. A cikkben az *Area Under the Receiver Operating Characteristic Curve* (AUC) mérőszámot vizsgálják,[7] hogy a legkisebb eltérésekre mennyire érzékeny és ezáltal mennyire jól lehet vele a torzításokat meghatározni.

Szintén a *Jigsaw2* adathoz tartozó versenyt próbálja implementálni a [2] cikk. A cikkben *Logisztikus regressziós*[8], *Naive Bayes* [3] és neurális hálózatos modelleket is teszteltek. Az előző cikkhez hasonlóan, szintén AUC értékeket vizsgáltak. A legjobb modelljüknek az eredménye $0,7785$ értéket adta.

A [4] cikk az *Jigsaw1* adattal dolgozik, amiben egy többosztályos feladatot, tehát maga a verseny feladatot próbálják a lehető legjobban megoldani. A kommenteket a lehető legjobb osztályokba, alosztályokba szeretnék sorolni felhasználva *Logisztikus regressziós* modellt és *Convolutional Neural Network* (CNN)[5] és *Long-Short Term Memory* (LSTM) [6] neurális hálón alapuló modelleket. A legjobb *accuracy*-val[7] rendelkező modell 2 LSTM réteget és 4 konvolúciós réteget tartalmaz. Ezzel a modellel $0,9645$-ös értéket kaptak, elég közeli értéket a nyertes *kaggle*-s csapathoz, akik $0,985$ pontossággal jósolták meg a kommentekhez tartozó címkéket.

Továbbá a [7] cikk számos neurális hálózatokból álló modellt hasonlít össze szintén a *Jigsaw1* adat osztályozása folyamán. Olvashatunk CNN, LSTM, *Feed Forward Neural Network* (FFNN) [8] és *Gated Recurrent Unit* (GRU) [9] osztályozások eredményeiről.[9] Továbbá részletesen beszámolnak az adat előkészítés folyamatáról, illetve az attribútumok generálásáról, mint például *GloVe* [10] vagy *fastText* [11] modellekről.

A [12] cikk az előzőktől eltérően nem a *Jigsaw* adatokat, hanem a fent említett *Crowdflower* adatát használja fel. A cikkben többosztályozós modelleket építenek fel és hasonlítanak össze, mint például a *Logisztikus regressziót*, *Naive Bayest* vagy *Support Vector Machine*-t (SVM) [13]. Továbbá nem csak modelljeik változatosak, hanem a szövegek feldolgozása is. Az adat előfeldolgozása során *N-gram* modellt is alkalmaznak, amiről a 2.4 részben későbbiek folyamán részletesen ismertetünk. A legjobb eredményt a logisztikus regressziós modell használatával érték el.

[7] A 2.3 részben részletesen kifejtjük az AUC és accuracy mérőszámok fogalmát

[8] A 2.2 részben részletesen beszámolunk a modellről.

[9] Mivel sok modellt hasonlít össze, különböző paraméterek beállításával és mindegyik modell más-más beállításnál ad kiváló értékeket, így ezt nem részletezzük, viszont a cikkben a részletes eredmények megtalálhatóak.

1.2. A feladat

A szakdolgozat során a gépi tanulás ismeretek felhasználásával egy programot készítünk, amely képes sértő, bántó tartalmú szövegeket detektálni. Egy hasonló korábbi kezdeményezés 2015-ben kezdődött, amikor *Aja Bogdanoff* és *Christa Mrgan* az udvariasság jelenlétét kezdték vizsgálni az online platformokon, különös tekintettel a híroldalakra. Ennek eredményeképp létrehozták a *Civil Comments*[10] nevű szoftvert, amely lehetővé tette a kommentet beküldő felhasználók számára, hogy egymás kommentjeit értékeljék udvariasság szempontjából. Ez a szoftver nagyon hatékonynak bizonyult, a kommentek egyre igényesebbek, átgondoltabbak lettek, egyre kevesebb vulgaritást tartalmazva. Azonban befektetők hiányában a kezdeményezésnek vége szakadt, viszont a projekt során begyűjtött adatokat (a kommenteket és azon értékeléseit) nyilvánosságra hozták, további kutatási lehetőséget biztosítva. Ezt felhasználva a *Jigsaw Unintended Bias in Toxicity Classification* létrehozott egy kibővített adathalmazt, amelyet eredetileg egy, *kaggle*-n meghirdetett verseny keretein belül használtak. A verseny célja a toxikus kommentek felismerése és a modellbeli véletlen torzítás minimalizálása volt. A munkánk során mi is ezzel az adattal fogunk dolgozni.

A *Jigsaw* egy hasonló versenyt is meghirdetett azelőtt, a *Toxic Comment Classification Challenge*-et, ahol szintén toxikus kommentekkel dolgoztak a versenyzők, viszont a fő feladat a kommentek különféle csoportokba, alcsoportokba való helyes osztályozása volt. A dolgozat az utóbbi feladathoz közelebb áll, viszont többosztályozós feladat helyett, bináris osztályozással foglalkozunk.

A *Jigsaw2* adatban számos online komment olvasható, amik egy része vulgáris, offenzív szavakat, kifejezéseket tartalmaz.

Az adat több fájlból is áll, melyek egy része a verseny után lett hozzáadva további kutatási lehetőséget nyújtva. Ezek közül a munkánk során a *train.csv* és a *test_private_expanded.csv* fájlokat használjuk. A *train.csv*-t a modell tanítására és validációjára, míg a *test_private_expanded.csv*-t a tesztelésre alkalmaztuk.

A *train.csv* 1804874 sorból és 45 oszlopból áll. Az adat egy részét a 1.1 ábra szemlélteti.

[10] `https://medium.com/@aja_15265/saying-goodbye-to-civil-comments-41859d3a2b1d`

id	target	comment_text	severe_toxicity	obscene	identity_attack	insult	threat
59856	0.8936170212765957	haha you guys are a bunch of losers.	0.021276595744680847	0.0	0.021276595744680847	0.8723404255319149	0.0
59859	0.6666666666666666	ur a sh*tty comment.	0.047619047619047616	0.6380952380952379	0.0	0.3333333333333333	0.0
59861	0.4576271186440678	hahahahahahahahhha suck it.	0.05084745762711865	0.3050847457627119	0.0	0.2542372881355932	0.0
59863	0.0	FFFFUUUUUUUUUUUUUU	0.0	0.0	0.0	0.0	0.0
239575	0.0	The ranchers seem motivated by mostly by greed; no one should have the right to allow their animals ...	0.0	0.0	0.0	0.0	0.0
239576	0.0	It was a great show. Not a combo I'd of expected to be good together but it was.	0.0	0.0	0.0	0.0	0.0

1.1. ábra. train.csv adat pár soros részlete

Az *id* a nevéből adódóan egy egyedi azonosítót biztosít a rekordoknak, aminek köszönhetően egy-egy rekord könnyen visszakereshető. A kommentek tényleges szövegét a *comment_ text* tartalmazza. A *target* folytonos változókat tartalmazó oszlop, a $[0, 1]$ intervallumból, amely a toxicitás mértékét mutatja: minél közelebb van 1-hez annál negatívabb hangvételű, és minél közelebb van 0-hoz, annál kevesebb negatív elemet tartalmaz a szöveg. Ez az érték azon felhasználók arányát mutatják, amelyek toxikusnak ítélték az adott kommentet. A célunk a target változó prediktálása, a teszthalmazon. A további attribútumok kevésbé lényeges szerepet töltenek be a dolgozatunk szempontjából. Ezek a változók specifikusabb (például bőrszínre, nemi identitásra, vallásra vonatkozó) kutatásokra adnak lehetőséget, a dolgozat célja azonban általánosan detektálni a toxicitást. Ennek kivitelezésére a comment_text változó elegendőnek bizonyult. Mindemellett röviden bemutatjuk a további változókat is.

Ezen változók, mint a korábban bemutatott target változó, szintén folytonos változók a $[0, 1]$ intervallumon. A változókat következőképpen csoportosíthatjuk:

- toxicitás alcsoportjait kifejező attribútumok: severe_toxicity, obscene, threat, insult, identity_attack, sexual_explicit
- nemiségre vonatkozó attribútumok: male, female, other_gender, transgender
- szexuális beállítottságra vonatkozó attribútumok: heterosexual, homosexual_gay_or_lesbian, bisexual, other_sexual_orientation

- vallási felekezetre vonatkozó attribútumok: christian, jewish, muslim, hindu, buddhist, atheist, other_religion
- bőrszínre vonatkozó attribútumok: other_race_or_ethnicity, black, white, asian, latino
- fizikális és mentális egészségre vonatkozó attribútumok: other_disability, physical_disability, intellectual_or_learning_disability, psychiatric_or mental_illness

A verseny folyamán regressziós feladatot oldottak meg a versenyzők, viszont a dolgozatban bináris osztályozással foglalkozunk, amihez a folytonos target változót binárissá kell alakítanunk. A változót a következőképpen alakítottuk binárissá: a 0.3-nál kisebb értékeknek 0-ás címkét, a 0.7-nél nagyobb értékeknek pedig 1-es címkét adtuk. A köztes részt, tehát a 0.3-nál nagyobb de 0.7-nél kisebb értékeket elhagytuk.

A célváltozó értékének binárissá alakításával mesterségesen egyszerűsítettük az adatot, így a feladatot is. Az adat egyszerűsítésnek az okát a 1.1 táblázatban található példa rekordokon keresztül mutatjuk be.

Rekord	**Target**
This bitch is nuts. Who would read a book by a woman.	0.83
haha you guys are a bunch of losers.	0.89

1.1. táblázat. Példa rekordok a train.csv adatból

Elolvasva a példamondatokat érezhető a toxicitás tartalom, viszont a célváltozó értékét nem feltétlenül tudjuk értelmezni, hogy az egyik mondat 0.06-tal miért is lett negatívabb, mint a másik. A szövegeket emberek pontozták $[0, 1]$ intervallumon toxikusságukat tekintve, majd a pontszámokat átlagolták, így kapva meg a target értéket. Ahogy a táblázatban is látható, kis eltérések nem változtatnak azon, hogy egy mondat toxikus-e vagy sem, emiatt a fent részletezett binárissá alakítást elvégezhetjük. Továbbá számunkra elég a biztosan nem toxikus és a biztosan toxikus tartalmú szövegeket vizsgálni, mivel a köztes részen apró eltérésekből adódik, hogy hova sorolták a rekordokat, így könnyebb ezt a részt elhagyni.

A *test_private_expanded.csv* adatot a *kaggle* verseny során is a végső teszteléshez használták a verseny kiírók. Ezen az adaton való végső kiértékelés határozta meg a verseny végső ranglistáját, így mi is ezen végeztük a végső futtatásokat. Az adat a *train.csv*-hez hasonlóan ugyanazokkal az attribútummal rendelkezik. A számunkra fontos attribútumok a *id*, a *comment_text* és a *toxicity*. A *train.csv* adathoz hasonlóan az *id* az egyedi

azonosító, és a *comment_ text* tartalmazza a szövegeket. Az *toxicity* ugyanúgy, mint a *target* változó $[0, 1]$ intervallumon veszi fel az értékeit, és minél közelebb van a változó értéke 1-hez, annál toxikusabb az adott szöveg. Mivel bináris osztályozással dolgozunk, a fentiekhez hasonló módon az adat alsó és felső részével dolgozunk, a középső részt elhagyjuk. Ennek és a többi futtatásnak az eredménye a 3. fejezetben olvasható.

2. fejezet

Módszerek áttekintése

2.1. Felügyelt tanulás

A gépi tanulás egyik alcsoportja, az úgynevezett *felügyelt tanulás*. A felügyelt tanulás során az adatunkat értelmezhetjük úgy, mint egy táblázatot. A táblázat sorait rekordoknak vagy példányoknak nevezzük. A táblázat oszlopait attribútumoknak hívjuk, amik a rekordok jellemzőit tartalmazzák. Az attribútumok között van egy úgynevezett *kitüntetett attribútum*. A kitüntetett attribútumot célváltozónak, címkeváltozónak, osztályváltozónak vagy magyarázott változónak nevezzük. Ennek alapján a többi attribútumot magyarázó változónak is szokták nevezni. Ha címkeváltozó értékkészlete diszkrét, osztályozási feladatról beszélünk, ha a címkeváltozó értéke folytonos, akkor regresszióról. Hogyan is zajlik egy felügyelt tanítás?

A tanítás megkezdése előtt fontos, hogy három adathalmaz a rendelkezésünkre álljon: tanító, validációs és teszthalmaz. A tanítás folyamán a *tanító halmazon* építjük fel a modellünket. A tanító halmazunk attribútumokból és címkeváltozókból áll. A modellépítés során regressziós vagy osztályozó eljárásokat alkalmazva, a magyarázó változókat felhasználva szeretnék megjósolni a magyarázott változót. A betanított modellt a *validációs halmazon* teszteljük le. Tesztelés során a betanított modellnek megadjuk a magyarázó változókat, amikhez a modell jósol egy címkét. A validációs halmaz a tanító halmaz részhalmaza, így ismertek a magyarázó és magyarázott változói. Ennek köszönhetően a validációs halmazon külön választhatjuk az attribútumokat a címkéktől, és a modellünket tesztelhetjük ezeken az attribútumokon. Mivel a címkeváltozók ismertek, így a modell által jósolt eredményeket össze tudjuk hasonlítani az eredeti értékekkel, ezáltal egy visszajelzést kapunk a modell osztályozóképességéről. Ha nem vagyunk megelégedve az osztályozást illetően, akkor a modellen módosíthatunk, azáltal, hogy más paraméterbeál-

lításokat használunk, esetleg új magyarázó változókat vezetünk be. Majd a módosított modellt a tanító és validációs halmazon, az előzőkben leírtak szerint újra futtatjuk. Ezeket a folyamatokat addig ismételjük, amíg elégedettek nem leszünk a kapott eredménnyel. Ha már nem akarunk többet változtatni a modellünkön és véglegesnek gondoljuk azt, csakis abban az esetben futtatjuk a *teszthalmazon*. A teszthalmazon való futtatás adja meg, hogy a modell egy ismeretlen, még nem látott adaton, milyen jó predikcióval szolgál.

2.2. Logisztikus regresszió

A logisztikus regresszió, amely egy osztályozási algoritmus, a nevét a logisztikus függvényről, azaz a szigmoid függvényről kapta. Vezessük be a következő matematikai jelöléseket:

Legyen X a tanítóhalmazunk, ami n darab dokumentumot tartalmaz. A dokumentumaink szöveg-címke párokból állnak, jelöljük a tanítópontokat az alábbi módon: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Az x_i szöveghez k darab attribútum tartozik, amit a következőképpen írhatunk fel: $x_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$. Az $y_i \in \{2, 3, \ldots, M\}, M \leq n$ halmaznak, mivel a rekordjainak legalább két de legfeljebb n osztályba tartozhatnak. Az attribútumainkhoz súlyokat társítunk. Jelöljük ezeket a súlyokat w-vel, ahol $w = (w_0, w_1, \ldots, w_k)$. A súly-attribútum kapcsolatot lineárisan írjuk fel, amihez bevezetjük, hogy $\forall i$-re, $x_{i0} = 1$. Így $w^T x_i$-re a következő egyenletet kapjuk:

$$w^T x_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \ldots + w_k x_{ik} \tag{2.1}$$

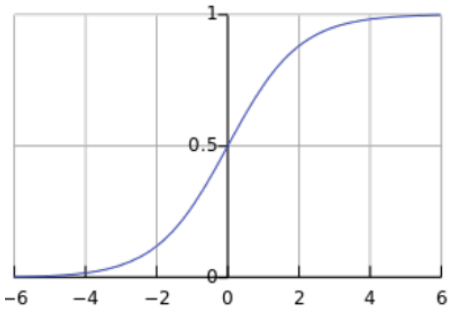
Ekkor a hipotézisünket az alábbi módon írhatjuk fel:

$$\hat{y} = h_w(x_i) = \sigma(w^T x_i) \tag{2.2}$$

ahol $\sigma(z) = \frac{1}{1+e^{-z}}$ a szigmoid függvény.

A 2.2 egyenletet szavakkal a következőképpen fogalmazhatjuk meg: a hipotézis megadja, hogy milyen valószínűséggel lesz 1-es a címkeváltozó az x_i bemenettel.

A 2.1 kifejezés értékkészlete $[-\infty, \infty]$, az osztályozáshoz azonban valószínűségi értékekre van szükség, amelyek $[0, 1]$-beliek.



2.1. ábra. Szigmoid függvény [1]

A 2.1 ábrán a szigmoid függvény látható. A szigmoid függvény segítségével a $[-\infty, \infty]$ intervallumról a $[0, 1]$-re tudjuk képezni az értékeinket. A függvényt a következő tulajdonságai miatt használják előszeretettel:

1. valós értékű
2. nemlineáris
3. monoton
4. $[0, 1]$ intervallumra képez

Továbbá a 2.1 ábráról leolvasható, hogy a pozitív x értékek esetén 1-hez közeli, a negatív x esetén 0-hoz közeli számokat ad vissza. Ha $x = 0$, akkor pontosan 0.5 lesz a függvény értéke. Tehát tényleg a valószínűségszámításban megszokott értelmezési tartományt kapjuk vissza.

A teljesség kedvéért megjegyezzük, hogy a tényleges modellezési fázisban a hipotézis értékét átváltjuk az osztálycímkék értékére a *döntési határ* segítségével. A döntési határt adott x értékre a 2.3 egyenlet mutatja be:

$$\hat{y} = \begin{cases} 0 & \text{ha } h_w(x) < 0.5 \\ 1 & \text{ha } h_w(x) \geq 0.5 \end{cases} \tag{2.3}$$

Ha a hipotézis értéke 0.5-nél nagyobb, akkor 1-es osztályba sorolja a rekordot, különben 0-ásba. Általában a döntési határt 0.5-nek választják, és a modellezés folyamán ezen értéken mi sem változtatunk. A továbbiakban a hipotézis vizsgálatánál maradunk, és egy-egy jóslás értéket valószínűségi tekintetében értelmezzük. Ez azt jelenti, hogy a jóslás értéke nem bináris értéket vesz

[1] A kép forrása: `https://laptrinhx.com/activation-functions-explained-1722166459/`

fel, hanem a $[0, 1]$ intervallumról egy számot, ami megmutatja, hogy milyen valószínűséggel ad a modellünk 1-es címkét a rekordnak.

A következőkben a *költségfüggvény* fogalmát vezetjük be. A költségfüggvény kapcsolatot teremt a jósolt és az eredeti címke értéke között. A modellünk minél jobb predikcióval szolgál az eredeti címkére, annál kisebb értéket vesz fel a költségfüggvényünk. Ha a jóslat eltér az adott címkétől, akkor a költségfüggvény értéke nagy lesz. A költségfüggvényünk akkor teljesít jól, ha a függvény értéke a lehető legkisebb, tehát a célunk a függvény minimalizálása. Képezzük a minimalizálandó célfüggvényt, azaz a költségfüggvényt egy konkrét rekordra, amit a következőképpen írhatunk fel:

$$cost(h_w(x), y) = \begin{cases} -\log(h_w(x)) & \text{ha y=1} \\ -\log(1 - h_w(x)) & \text{ha y=0} \end{cases} \tag{2.4}$$

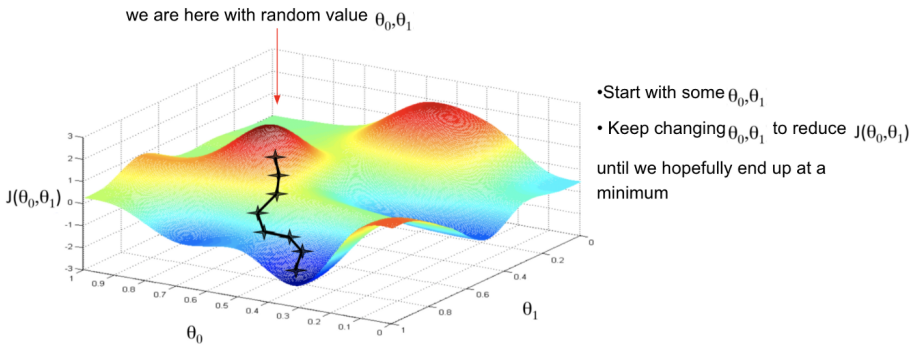
Alakítsuk át egy kompaktabb formává:

$$cost(h_w(x), y) = -y \log(h_w(x)) - (1 - y) \log(1 - h_w(x)) \tag{2.5}$$

Az előző alakból már könnyedén felírható a teljes költségfüggvény:

$$C(w) = -\frac{1}{n} \sum_{i=1}^{n} (y_i \log(h_w(x_i)) - (1 - y_i) \log(1 - h_w(x_i))) \tag{2.6}$$

Miután megkaptuk a teljes költségfüggvényt, már csak a minimalizálás szükséges. A minimalizálás történhet analitikus úton, gradiens vagy sztochasztikus gradiens módszerrel egyaránt. Általában a minimalizálást a logisztikus regressziós modelleknél gradiens módszerrel végzik, így ezt a minimalizációs folyamatot fejtjük ki részletekbe menően.



2.2. ábra. Gradiens módszer szemléltetése[2]

A gradiens módszer egy f függvény minimumhelyének megtalálására szolgál (analóg módon maximumhely keresésére is alkalmazható). Vizsgáljuk meg a 2.2 ábrát, ami szemlélteti a módszer működését. Az ábrán a θ_0 a w_0-val, a θ_1 a w_1-gyel és a $J(\theta_0, \theta_1)$ a $C(w)$, $i = 1$ jelöléssel analóg. Kezdésként kiválasztunk egy véletlen kezdőpontot. Minden irányba megvizsgáljuk a csökkenés mértékét, és ahol a legnagyobb, abba az irányba lépünk. Ezt adja meg a $-gradf$ kifejezés. Az előző műveletet addig ismételjük, amíg a minimalizálandó függvényünk együtthatói minimális változást eredményeznek az iterációk során. Az ábrán egy szerencsés kiindulási ponttal egy olyan útvonal látható, amivel a globális minimumba érkezünk. Ha jobban megvizsgáljuk az ábrát, akkor a $\theta_1 \approx 0.4$ körül található egy lokális minimum, és akár más kiindulási ponttal oda is konvergálhatott volna a függvényünk. Mivel a globális minimumot keressük, ezért a lokális minimumba való beragadás elkerülésére törekszünk. Ha egy lokális minimumba konvergál a függvényünk, akkor hiába szeretnénk kimozdítani onnan, nincs rá lehetőség, mivel nála kisebb irányba lépés nem lehetséges. Ezért a lokális minimumba érkezést és az ott ragadást megelőzve, több kiindulási pontból is érdemes letesztelni az algoritmust, hogy tényleg a megfelelő minimumhelyet adja-e vissza. Ha minden véletlen választással ugyanazokat az együtthatókat kapjuk, akkor ezen értékeket kiválasztva, megkapjuk a függvényünk minimális helyét.

A gradiens módszer egyik fontos paramétere az úgynevezett *tanulóráta* (*learning rate*). Ezt a paramétert, a következőképpen fogalmaztuk meg az előzőkben: „ *a legnagyobb csökkenés irányába lépünk* ". Ezt a bizonyos „ *lépést* " nevezzük szakszóval tanulórátának. Ez a paraméter befolyásolja, hogy milyen gyorsan haladunk a minimumhely felé. Ha a tanulóráta értékét túl nagyra állítjuk, akkor fennáll a veszélye, hogy átugorjuk a minimumhelyet. Ha túl kicsinek adjuk meg az értéket, akkor sok lépést szükséges megtenni, amíg elérjük a kívánt helyet.

A sztochasztikus gradiens módszer az előzőkben részletezett gradiens módszerhez hasonlóan működik, viszont nem a teljes adathalmazra vonatkozóan, hanem egy adott rekordra számítja ki a gradienst az iterációban. Matematikailag a következőképpen fogalmazhatjuk meg a sztochasztikus gradiens iterációnkénti változását:

$$w^{(i+1)} = w^{(i)} - \lambda \cdot \left. \frac{\partial cost(h_w(x), y)}{\partial w} \right|_{w^{(i)}} \tag{2.7}$$

ahol λ a tanulóráta, a $cost(h_w(x), y)$ adott x rekordra vonatkozik.

Térjünk vissza a 2.6 egyenlettel megadott költségfüggvényünkhöz és írjuk fel a gradiensét a következőképpen:

[2]A kép forrása: `https://www.kdnuggets.com/2020/05/5-concepts-gradient-descent-cost-function.html`

$$\frac{\partial C(w)}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}(h_w(x_i) - y_i)x_i \tag{2.8}$$

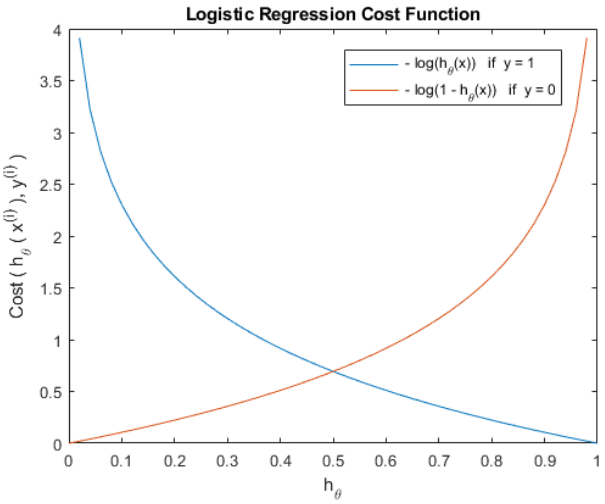
Ezután a megadott 2.8 egyenlettel, minden iterációban frissítjük a súlyokat, hogy megkapjuk a végső értékeket, amik a minimumhelyet jelölik. A súlyok iterációban történő változását az alábbi módon írhatjuk fel:

$$w^{(i+1)} = w^{(i)} - \lambda \cdot \left.\frac{\partial C(w)}{\partial w}\right|_{w^{(i)}} \tag{2.9}$$

ahol a $w^{(i)} = (w_0^{(i)}, w_1^{(i)}, \ldots, w_k^{(i)})$ az i-edik iterációban szereplő súlyokat, a $w^{(i+1)}$ az új súlyokat, a λ pedig a tanulórátát jelöli.

A 2.6 egyenlettel megadtunk egy költségfüggvényt, viszont eddig nem vizsgáltuk meg, hogy tényleg megfelelően működik a konstruált függvényünk. A függvény akkor teljesít megfelelően, ha félreosztályzás esetén nagy, jól osztályzás esetén kis értéket ad vissza. Vizsgáljuk meg a következő felmerülő eseteket:

- Ha a jósolt érték, tehát a $h_w(x)$ értéke közel van 1-hez és $y = 1$, akkor a különbség jobb oldala 0 lesz, és a logaritmus kis értéket ad vissza, így a költség kicsi marad.
- Ha a $h_w(x)$ értéke közel van 0-hoz és $y = 1$, akkor a különbség jobb oldala 0 lesz, és a logaritmus nagy értéket ad vissza, így a költség nagy lesz, tehát bünteti a rossz jóslást.
- Ha a $h_w(x)$ értéke közel van 0-hoz és $y = 0$, akkor a különbség bal oldala 0 lesz, és a logaritmus kis értéket ad vissza, így a költség kicsi lesz.
- Ha a $h_w(x)$ értéke közel van 1-hez és $y = 0$, akkor a különbség bal oldala 0 lesz, és a logaritmus nagy értéket ad vissza, így a költség nagy lesz, tehát bünteti a rossz jóslást.



2.3. ábra. Költségfüggvény változása [3]

A költségfüggvény változását a 2.3 ábrán nyomon követhető. Az ábrán feltüntetett jelölések megegyeznek a következő általunk használt jelölésekkel: $h_\theta(x)$ a $h_w(x)$-szel, $cost(h_\theta(x^{(i)}), y^{(i)})$ a $cost(h_w(x), y)$-nal analóg. A kép jobb felső sarkában látható, hogy adott y értékhez, melyik görbe tartozik és a $h_\theta(x)$ érték függvényében mekkora változást eredményez.

A logisztikus regressziót szokták *logit modellnek* is nevezni. Az elnevezést onnan kapta, hogy a szigmoid függvény inverzét logit függvénynek nevezik. Használjuk a következő jelölést a valószínűség értelmezés miatt:

$$p = h_w(x_i) = \sigma(w^T x_i) = \frac{1}{1 + e^{-w^T x_i}} \Rightarrow p = \frac{1}{1 + e^{-w^T x_i}} \tag{2.10}$$

Képezzük a kifejezés inverzét:

$$\begin{aligned} \frac{1}{1 + e^{-w^T x_i}} &= \frac{1}{p} \\ e^{-w^T x_i} &= \frac{1}{p} - 1 = \frac{1 - p}{p} \\ -w^T x_i &= \log\left(\frac{1 - p}{p}\right) \\ w^T x_i &= \log\left(\frac{p}{1 - p}\right) \end{aligned} \tag{2.11}$$

[3] A kép forrása: `https://blogs.cuit.columbia.edu/zp2130/files/2018/09/Matlab_Plot_Logistic_Regression_Cost_Function.png`

Tehát a logit függvény a következőképpen írható fel:

$$logit\,(p) = \log\left(\frac{p}{1-p}\right) = w_0 + w_1 x_{i1} + \ldots + w_k x_{ik} \tag{2.12}$$

A $w = (w_0, w_1, \ldots, w_k)$ súlyok fontos szerepet szolgálnak a modell értelmezésében, amit a 3. fejezetben részletezünk.

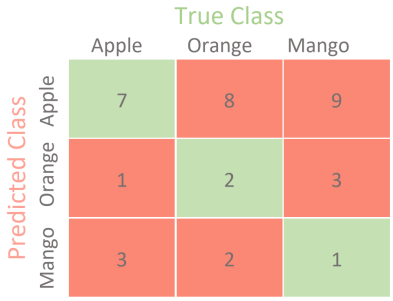
2.3. Mérőszámok

Az egyik legfontosabb mérőszám az osztályozó *pontossága* (*angolul accuracy*), ami a jól osztályozott példányok arányát veszi az összes példány számához viszonyítva. A pontossághoz hasonló mérőszám a *hibaarány* (*angolul misclassification error, error rate*), ami a félreosztályzott példányok arányát adja meg. Ezt a következőképpen is megkaphatjuk:

$$\text{hibaarány} = 1 - \text{pontosság} \tag{2.13}$$

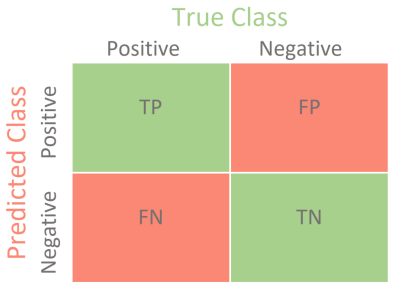
Viszont a pontosság és a hibaarány nem minden esetben ad elegendő információt a modellezés folyamán. A következőkben egy példát mutatunk arra, amikor a pontosság félrevezető mérőszámnak bizonyul. Vegyük a következő kiegyensúlyozatlan osztályozási feladatot: 1000 emberből, az emberek 1%-ka egy ritka betegségben szenved és szeretnénk meghatározni, kiket érint a betegség. Két osztályba soroljuk az egyedeket: a betegségben szenvedő, illetve nem szenvedő csoportokra. Ha a modellünk az összes betegségben szenvedő egyént nem betegnek jósolja, akkor is a modellünk 99%-os pontossággal teljesít. Az eredmény félrevezető, hiszen a modell majdnem 100%-osan teljesít, mégis a kérdéses legfontosabb 1%-ot nem tudja eltalálni.

Az előző példa ismeretében belátható, hogy szükségünk van további mérőszámok bevezetésére. Ezek megadásához vezessük be az úgynevezett *keveredési* vagy *tévesztési mátrixot* (*angolul onfusion matrix*), ami annyi sorból és oszlopból áll, ahány osztálya van. A mátrix (i, j) eleme azt mutatja meg, hogy az i-edik osztályozó hányszor sorolja a j-edik osztályba. A mátrix diagonális elemei lesznek a helyesen osztályozott rekordok száma.



2.4. ábra. Keveredési mátrix [4]

A 2.4 ábrán látható egy példa keveredési mátrixra. Három különböző osztálya van a feladatnak, így egy 3×3-as mátrixot kapunk. Vizsgáljuk meg néhány elemét a mátrixnak, hogy jobban megértsük a tévesztési mátrixot. Az első oszlop megmutatja, hogy 11 rekord eredetileg alma osztályba tartozik. A mátrix $(1, 1)$ eleme megadja, hogy a 11 rekordból mindössze 7 rekordot mondott ténylegesen almának a modell, a többit félreosztályozta. A félreosztályzott elemeket a $(2, 1)$ és $(3, 1)$ cellák tartalmazzák. A modell 1 almát narancsnak, 3 almát pedig mangónak osztályzott. A mátrix többi eleme hasonló gondolatmenettel értelmezhető.



2.5. ábra. Bináris keveredési mátrix [4]

További fogalmakat vezethetünk be, ha bináris osztályozásról beszélünk. A jól osztályozott példányokat *True Positive*-nak (TP) és *True Negative*-nak (TN), illetve a hibásan osztályozott példányokat *False Positive*-nak (FP) és *False Negative*-nak (FN) nevezzük. Mivel bináris osztályozásról beszélünk, így 2×2 tévesztési mátrixot határozunk meg, ahogy a 2.5 ábra is mutatja. Az ábrán továbbá látható, hogy a bevezetett jelöléseink melyik cellákra vonat-

[4] A képek forrása: `https://laptrinhx.com/confusion-matrix-for-your-multi-class-machine-learning-model-3282802856/`

koznak. A bináris keveredési mátrix elemeiből a következőkben bevezetett mérőszámokat kaphatjuk.

A *recall*[5] a következő hányados adja:

$$R = \frac{TP}{TP + FN} \tag{2.14}$$

Tehát azt adja meg, hogy milyen arányban találjuk meg a pozitívakat a valóban pozitívak közül.

A *precision*-t[5] a következőképpen számoljuk:

$$P = \frac{TP}{TP + FP} \tag{2.15}$$

Ez megadja, hogy milyen arányban valóban pozitív a pozitívaknak mondottak közül.

Az előző két mérőszámnak a harmonikus közepét *F-mértéknek* (F-measure) nevezzük, és az alábbi módon kapjuk:

$$F = \frac{2PR}{P + R} \tag{2.16}$$

A már korábban bevezetett pontosságot(accuracy) is felírhatjuk ezen új jelölésekkel:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.17}$$

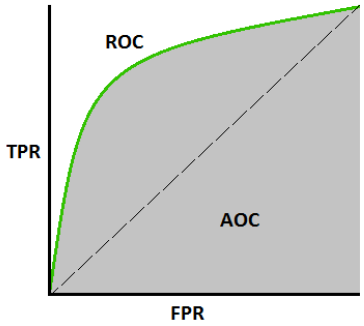
Az előbb bevezetett fogalmak többosztályos osztályozási feladat esetén is alkalmazhatóak, ha az egyik osztályt kinevezzük pozitívnak és a többit egybe negatívnak vesszük.

Az előbb bevezetett fogalmak további mérőszámok bevezetésére alkalmazhatóak. A *True Positive Rate* (TPR) más néven recall, a 2.14 egyenlet szerint számoljuk ki. A *False Positive Rate* (FPR) az FP és a valóban negatív osztály hányadosa adja, és a következőképpen írható fel:

$$FPR = \frac{FP}{TN + FP} \tag{2.18}$$

A TPR-t ábrázolva az FPR függvényében az úgynevezett *ROC-görbét* (*angolul Receiver Operating Characteristics Curve*) kapjuk. A görbét $[0, 1] \times [0, 1]$ részen értelmezzük. A 2.6 ábrán látható egy modell ROC-görbéje.

[5] A precision és recall mérőszámokat magyarul *pontosságnak* és *megbízhatóságnak* szokták fordítani, viszont dolgozatunkban a félreértelmezés elkerülése érdekében maradunk az angol elnevezéseknél.



2.6. ábra. ROC-görbe [6]

A ROC-görbe esetén, pontértékek alapján sorba rendezzük a tesztadatunkat és egy bizonyos küszöbérték felett 1-esnek, alatta 0-ásnak osztályozzuk a rekordokat. A ROC-görbe néhány pontjának értelmezése ismert, ezeket a következő pontokban foglaljuk össze:

- $(TPR = 0, FPR = 0)$: A modell minden rekordját negatív osztályúnak jósoljuk
- $(TPR = 1, FPR = 1)$: A modell minden rekordját pozitív osztályúnak jósoljuk
- $(TPR = 1, FPR = 0)$: A tökéletesen prediktáló modell

A ROC-görbe alatti területet AUC-nak (*angolul Area Under the Curve*) nevezzük, amit a 2.6 ábra szürke színnel jelöl. A görbe alatti terület minél közelebbi értéket vesz fel az 1-hez, annál pontosabb predikciót ad. Ha az AUC értéke 0.5-tel egyenlő, tehát a 2.6 ábrán a szaggatott vonal alatti terület nagyságát adja meg a mérőszám, akkor a modellünk ugyanolyan arányba találja el a 0-ás osztályt, mint az 1-esét. Az AUC mérőszámot modellek összehasonlítására is alkalmazhatjuk, ha nem tudunk két modell közül dönteni. Az összehasonlítás során azt a modellt választjuk, amelyiknek nagyobb az AUC értéke. Fontos megjegyezni, hogy a görbe alatti területet nem minden esetben megbízható mérőszám. Vegyük a fentiekben kifejtett kiegyensúlyozatlan osztályozási feladat példáját. A példában ismét minden betegségben szenvedő példányt osztályozzunk nem beteg egyednek. Ekkor a TPR és FPR értékeket kiszámolva, majd a ROC-görbét ábrázolva, azt tapasztalnánk, hogy a görbe a $(0, 1)$ ponthoz közeli ívet írna le. A görbe alatti területet kiszámolva egy 1-hez közeli számot kapnánk, amiből arra következtethetnénk, hogy egy jól osztályozó modellt készítettünk. Viszont ugyanúgy,

[6] A kép forrása: `https://www.mygreatlearning.com/blog/roc-curve/`

mint az accuracy-nál a feladatot nem teljesítettük, hiszen egyetlen betegségben szenvedő egyedet sem talált meg a modellünk. Ilyen esetekben jobb más mérőszámokkal vizsgálnunk a modellünket, például F-mértékkel.

2.4. Előfeldolgozás, attribútumgenerálás

Amikor szöveges adattal dolgozunk, számos adat előfeldolgozási folyamatot kell elvégeznünk, mint például adat tisztítása, formázása vagy átalakítása. Ezen folyamatokra azért van szükségünk, mert a gépi tanulási modellek nem tudnak nyers szöveges adattal dolgozni, ennek következtében a szövegeket vektorokká kell alakítani. A továbbiakban az atribútumokból létrehozott változókat *feature*-öknek (magyarul tulajdonságnak) nevezzük. Az egyik szöveg konvertálási módszer az úgynevezett *Bag of Words modell*. A modell működését a 2.7 ábrán szereplő példa segítségével mutatjuk be.

2.7. ábra. Példa a Bag of words modellre [7]

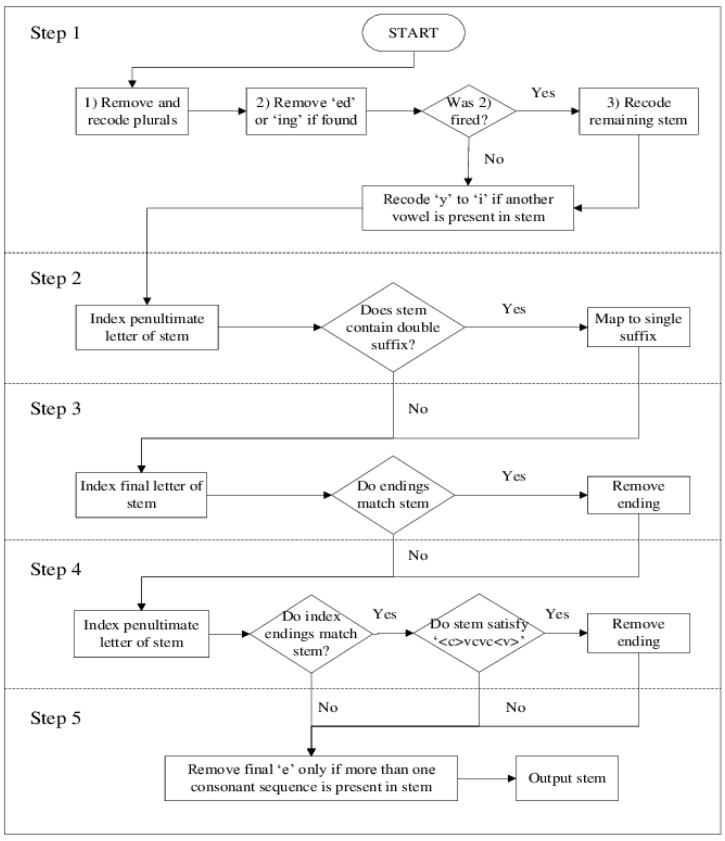
A 2.7 ábra négy dokumentumot tartalmaz, amelyeknek a vektoros alakjuk a táblázat megfelelő sorából kiolvasható. A példamondatokhoz tartozó sorokat megvizsgálva, látható, hogy a táblázat azon oszlopaiban szerepelnek 1-esek, amely szavak a példamondatokban megtalálhatóak. A táblázatban szereplő értékeket a modell a következőkben leírtak szerint hozza létre. Első lépésként a szövegekben előforduló különböző szavakból létrehozza a feature-öket, esetünkben a táblázat első sorát. Következő lépésben a modell rekordonként meghatározza, hogy melyik feature-ök szerepelnek a mondatban és azokhoz a feature-ökhöz 1-et rendel értékül, a többihez 0-át. A modell befejeztével egy mátrixot kapunk, aminek annyi sora van ahány dokumentum található a szövegben és annyi oszlopa, ahány különböző szó található az adatban.

A szakdolgozat folyamán Bag of Words típusú modelleket fogunk tanítani, viszont az adatban előforduló hangulatjelek, hastag-ek vagy linkek, a modell

[7] A kép forrása: `https://www.ronaldjamesgroup.com/blog/grab-your-wine-its-time-to-demystify-ml-and-nlp`

tanítása során nem hasznos feature-öket eredményezhetnek, emiatt ezeket az attribútumgenerálás megkezdése előtt kiszűrjük.

Az adat tisztításához a *tweet-preprocessor* és a tokenizációhoz, szótövesítéshez, illetve a stopword kiszűréséhez(ezeket a fogalmakat a következőkben részletesen ismertetjük) a *Natural Language Toolkit* Python könyvtárakat használtuk. Az adat tisztás folyamán kiszűrtük a szövegekből a hastag-eket, hangulatjeleket és a linkeket. Ezek után tokenizáltuk a rekordokat, azaz a mondatokat, szavak, illetve írásjelek szintjére bontottuk le. Következő lépésként, a kapott szavakat tartalmazó listát szótövesítettük. Erre a lépésre azért volt szükségünk, hogy ugyanazok a szavak a toldalékuk miatt, ne forduljanak később többször elő a feature-ök között. A modellünk például az *idiot* és *idiots* szavakat külön feature-nek venné, viszont ezek a szavak ugyanazt jelentik, csak az egyik egyes- a másik többesszámú alakban fordul elő. A szótövesítésnek a segítségével ezt a problémát kiküszöbölhetjük. Az nltk csomag különböző szótövesítő algoritmusokat tartalmaz, amiből mi a *Porter Stemmer* algoritmust használtuk.



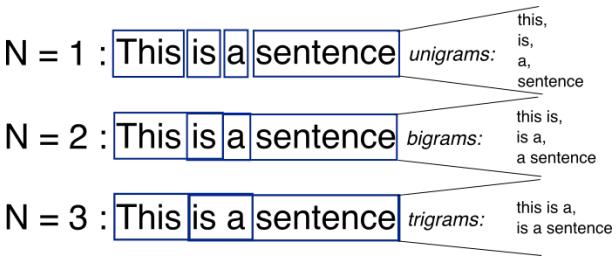
2.8. ábra. Porter Stemmer működése [8]

Az algoritmus öt lépésben távolítja el a szavak végéről a toldalékokat, amit a 2.8 ábrán részletesen nyomon lehet követni. Az algoritmus egyik hátránya, hogy nem létező szavakat hozhat létre ha az adott szónak a szóvégét tévesen ítéli egy adott toldaléknak.

A szótövesítést követően kiszűrjük az úgynevezett *stopword*-eket is. A *stopword*-ök közé tartoznak az olyan szavak, amik sokszor előfordulnak, viszont egyedül nem állnak a szövegben, ilyen például a prepozíciók, a névelők vagy a kötőszavak. Ezek a szavak nem hordoznak magukban információt, hiszen minden mondatban megtalálhatóak, ezért nem teszik egyedivé a szöveget, így elhagyhatóak. Az nltk csomag tartalmaz angol nyelvű szólistát, ami az előforduló stopword-öket tartalmazza. A szólistát meghívva a programunkba, a mondatokból csak azokat a szavakat tartjuk meg, amelyek ebben a stopword szólistában nem szerepnek, ezzel megszűrve a szövegeinket. A szólista megtalálható és letölthető a lábjegyzetben hivatkozott oldalon.[9]

A következőkben arról számolunk be részletesen, hogy milyen feature-öket hoztunk létre a modelljeink tanításához.

Az első kísérletek során a modellek feature-jeit a *Bag of N-grams modell* segítségével állítottunk elő. A Bag of N-grams modell a Bag of Words modell kiterjesztett változata, ahol szó feature-ök helyett n-gram feature-öket állítuk elő. Az N-gram egy N hosszú szavakból, írásjelekből álló listát határoz meg. A kiterjesztésnek köszönhetően a szövegben található szavakról több információhoz jutunk, például, hogy mely szavak fordulnak elő gyakran egymás után és melyek ritkábban. A munkásságunk során a feature-ök generálásához *unigrams*-ot használtunk, aminek a működését a 2.9. ábra szemlélteti, összehasonlítva a *bigrams*-szal és *trigrams*-szal.



2.9. ábra. Unigrams, bigrams, trigrams [10]

Az ábra a *This is a sentence* példamondaton mutatja be az *N-gram* működését, $N = 1, 2, 3$ esetekre. A példamondatra használva az algoritmust

[8] A kép forrása: `https://www.researchgate.net/figure/Porter-Stemming-Algorithm-4_fig1_260385215`

[9] `https://www.nltk.org/nltk_data/`

[10] A kép forrása: `https://goo.gl/images/mQMt5x`

$N = 1$ esetben, az első sor jobb oldali kimenetét kapjuk, tehát a végeredmény különálló szavak sorozata. Hasonlóan $N = 2$, $N = 3$ esetre, ahol különálló szavak helyett szópárokat és szóhármasokat kapunk. A folyamat hasonlóan zajlik N növelése mellett, az algoritmus kimenetként szó N-eseket ad vissza.

Tehát a szövegekből képzett eltérő unigrams-ok lesznek a feature-jeink és egy rekord az adott feature-nél 1-et vesz fel értékül, ha megtalálható benne, különben 0-át.

A Bag of N-gram feature-ökkel rendelkező modelleken elért eredményeken javítani szándékoztunk, így új feature-öket vezettünk be különböző megfigyelések alapján. Nevezzük alapmodellnek azt a modellt, amit az unigrams feature-ökkel rendelkező adaton tanítottunk.

Az alapmodell kvantitatív kiértékelését követően, ami a 3.1 részben olvasható, a félreosztályzott rekordokat vizsgáltuk meg. A részletes hibaelemzést a félreosztályzott rekordokról a 3.2 részben olvasható. A félreosztályzott toxikus szövegek közel fele rövid mondatokból állt a hibaelemzések során és ezen megfigyelés miatt vezettük be a *number of words* attribútumot. A number of words változó megszámolja, hogy milyen hosszú a tokenizált szöveg és a változó egy adott rekordnál, a rekord tokenizált hosszát adja vissza.

A következő feature-t a toxikus szövegek szófaj vizsgálatát követően vezettük be. A hibaelemzést követően a félreosztályzott toxikus szövegekben kevés ige fordult elő, emiatt létrehoztuk a *POS tags* feature-t. A POS tag-ok(*Part of Speech magyarul szófajok*), változókból négy feature-t különböztetünk meg egymástól: ige, főnév, melléknév és határozószó változókat. A változók lényege, hogy a tokenizált szöveget, szó-szófaj párokká alakítja, majd leszámolja, hogy az adott szövegben mennyi található abból a szófajból. Egyszerűség kedvéért a szófajoknak az alcsoportjait nem számoljuk külön, hanem egybe vesszük az összes fajtáját és az adja meg a szófaj számosságát. Tehát egy rekord ezen feature-nél, a szövegben található adott szófajú szavak számát tartalmazza.

```
@staticmethod
def POS_tag_verb(text):
    tags = nltk.pos_tag(word_tokenize(text))
    verb = sum(1 for w, tag in tags if tag == 'VB' or tag == 'VBD' or tag == 'VBG' or tag == 'VBN')
    yield ('VERB',verb)
```

2.10. ábra. Kódrészlet: POS_tag változók közül az igéket leszámláló változó

A 2.10 ábra egy kódrészletet tartalmaz a modellünkből az igék leszámlálásáról. A függvénybe bekerül a vizsgálandó szöveg, amit tokenizálunk. A tokenizációt követően elkészítjük a szó-szófaj párokat, majd csak azokat a szavakat számoljuk meg, amelyek igék. A többi szófaj leszámláló kód ha-

sonlóan működik, csak a megengedett leszámolható alszófajokban térnek el egymástól.

A következő változót, az úgynevezett *gazetteer*-t, a logisztikus regressziós modellből lekérdezett súlyok és a hozzájuk tartozó feature-ök kapcsolatából készítettük el. A gazetteer változó *Natural Language Processing* (*magyarul Természetes Nyelv Feldolgozás*) (NLP) feladatokban gyakran használt technikai eszköz. A változót a *Named Entity Recognition* (*magyarul Entitás Felismerés*) (NER) feladathoz használták elsősorban, ami egy olyan nyelvi elemző eljárás, amelynek a segítségével azonosíthatók egy szövegben a tulajdonnevek és a tulajdonnevekhez hasonlóan viselkedő elemek. Az azonosítás mellett szükséges a névelem típusának (például helyszín, személynév, földrajzi név) azonosítása is. A NER feladatról és megoldásához használt különböző gazetteer változókról a [14, 15] cikkekben olvasható.

A hibaelemzésben megfigyeltek alapján hoztuk létre a gazetteer változót. A változó, olyan szavakat tartalmaz, amelyek nagy súllyal rendelkeztek a modellben, továbbá olyan szavakat is beletettünk szólistába, amiket fontos toxikus szavaknak tartottunk. A bevezetett gazetteer feature 1-et ad értékül, ha a mondatban megtalálható a listában szerepelő szó, különben 0-át.

A fontosnak tartott szavakat a validációs halmazon félreosztályzott rekordok szavaiból választottuk ki, amelyek nem feltétlenül szerepeltek az eredeti feature-ök között, de mi beletettük a gazetteerbe. Emiatt a modellünk a validációs halmazon nagyon jó eredményeket adott, mivel a modellünk elkezdet rátanulni az egyedi hibákra. További eredményekkel és a gazetteer változó szóválasztás következményeivel a 3. fejezetben foglalkozunk részletesen.

2.5. Tanítás, paraméterbeállítások

A szakdolgozat kódja *Python* programozási nyelvben íródott. A modell négy különböző szkriptből áll, a könnyű kezelhetőség érdekében. A program kódja megtalálható a lábjegyzékben belinkelt *GitHub repository*-ban.[11]

Először *train.csv* adathalmazt beolvassuk az első szkriptbe, ahol a címkéket binárissá alakítjuk, továbbá a szöveges attribútumokból kiszűrjük a hastags-egeket, a linkeket és a hangulatjeleket. Így egy részben tisztított formázott adatot kapunk. Az adatot rögtön *tanító* és *validációs* halmazra bontjuk. Az adat 90%-a képzi a tanító halmazt, a maradék 10%-a, pedig a validációs halmazt. A szkriptünk kimenetét a tanító és validációs halmaz képzi. A következő szkript tartalmazza azokat a kódokat, amikkel a tanító és validációs halmazok attribútumaiból előállítjuk a 2.4 részben kifejtett feature-öket. A bináris feature-jeink az unigrams, gazetteer, a nem bináris

[11] `https://github.com/homolyapanni/thesis`

feature-jeink a POS tags-ek és a number of words változók lesznek. A program során beépítettünk egy úgynevezett *cutoff* paramétert, ami az előállított *unigrams*-ok számának alsó korlátot biztosít. Ez a paraméter megakadályozza, hogy a cutoff paraméternek megadott értéknél kevesebbszer előforduló szavakat a program bevegye a változók közé. A paraméter azért szükséges, hogy gyakori szavak kerüljenek a feature-ök közé. A cutoff paraméternél ügyelni kell a megfelelő alsó korlát kiválasztására. Ha túl kicsi számot választunk, akkor egy feleslegesen létrehozott paramétert kapunk, ellenkező esetben pedig, értékes szavakat hagyhatunk ki a változóink közül. A változók legenerálását követve egy mátrixot kapunk, ami annyi sorból áll ahány rekord található az adatban, és annyi oszlopból ahány attribútumot generáltunk. Továbbá létrehoztunk egy *boolean*[12] változót, ami az új feature-ök bevezetését gátolja. A szkriptet a tanító halmazra lefuttatva a boolean változó segítségével engedélyezzük, hogy a halmaz attribútumaiból előállítson feature-öket. Viszont ugyanezt a kódot futtatva a validációs halmazra már nem engedjük, hogy új változókat adjon a mátrix oszlopaihoz, hiszen tanító halmazon betanított modell nem tudná tesztelni a validációs halmazból képzett mátrixot, mivel olyan változókat is tartalmazna, ami a modellben nincs. A szkript kimenetként a mátrixot, a feature-öket és a rekordokhoz tartozó bináris címkéket téríti vissza. Az előző szkriptben megkapott mátrixot és a hozzájuk tartozó címkéket, a logisztikus regressziós modellen tanítjuk be. A szkript végén a betanított modellt kapjuk vissza, amit a negyedik szkriptben kiértékelünk a validációs halmaz segítségével. A kiértékelés során a 2.3 részben kifejtett mérőszámokat alkalmazzuk.

[12]Olyan változó, ami a program során True (igaz) vagy False (hamis) értékeket vehet fel.

3. fejezet

Eredmények

3.1. Kvantitatív eredmények

A tanítást négy különböző feature beállítással hajtottuk végre, amit a 3.1 táblázat foglal össze. A feature-jeinket cutoff = 5 beállítással generáltuk le.

Modellek	**Features**
Alapmodell	unigrams
Modell_1	unigrams + number of words
Modell_2	unigram + POS tags
Modell_3	unigrams + gazetteer

3.1. táblázat. Modellek bemutatása

A tanítás folyamán logisztikus regressziós modellt használtunk. A logisztikus regresszió a magyarázó és magyarázott változók segítségével felépíti a modellt. Logisztikus regresszió esetében a tanításnak akkor lesz vége, amikor megtalálja az optimális súlyokat a feature-ökhöz, ezt követően a felépített modellt a validációs halmazon értékeljük ki. A kiértékelés során a validációs halmaznak a címkeváltozóit prediktálja meg a modellünk, majd a kijövő eredményeket összehasonlítjuk a tényleges validációs halmaz címkeváltozóival. Az összehasonlítás során előállítjuk a tévesztési mátrixot, aminek a megfelelő elemeiből kiszámoljuk az accuracy, a precision, a recall és az F-mérték mérőszámokat. A 3.2 táblázat a futtatások számszerű eredményeit foglalja össze a különböző modellek esetén.

Mérőszámok	Modellek			
	Alapmodell	**Modell_1**	**Modell_2**	**Modell_3**
True positive	69	62	71	79
False positive	14	10	15	11
True negative	4827	4831	4826	4830
False negativ	90	97	88	80
Accuracy	97.92	97.86	97.94	98.18
Precision	83.13	86.11	82.56	**87.78**
Recall	43.40	38.99	44.65	**49.69**
F-score	57.02	53.68	57.96	**63.45**

3.2. táblázat. Modellek bemutatása különböző mérőszámokkal [1]

A táblázatban a megfelelő mérőszámok soraiban a legnagyobb elemet kiemeltük. A táblázat leglényegesebb sora az utolsó, ahol az F-mérték található. Az F-mérték a precision és a recall harmonikus közepe, tehát ezen változók függvényében változik, emiatt elegendő a precesion és a recall változását az F-mértékben nyomon követni és a modellek F-mértékét összehasonlítani.

Az alapmodellnek az unigrams-ot használó modellt használtuk, ami a táblázat értékeiből kiolvasva kiderül, hogy a véletlen osztályozó modellek eredményeihez hasonlóan teljesít. A véletlen osztályzó modellek alatt azt értjük, hogy a $\frac{1}{2}$-$\frac{1}{2}$ valószínűséggel osztályozza a rekordot toxikusnak vagy nem toxikusnak.

Az alapmodell javítása érdekében kísérleteztünk a bevezetett három modellel. Látható az F-mértékből, hogy a modell_1 az új feature bevezetésének következtében ront az eredeti modell értékén. A modell_2-ben hozzáadott feature minimálisan javít az F-mértéken, viszont a változás értéke nem elég nagy, hogy megtartsuk a feature-t más kísérletek folyamán. Az előző kettővel ellentétben a modell_3 sikeresnek mondható, ugyanis 0.7-dal javított az eredeti modell F-mértékéhez képest.

A modellek tényleges teljesítőképességük vizsgálatához, egy olyan adaton teszteltünk, amit még nem látott a modellünk. A *test_private_expanded.csv* adatot erre a célra használtuk fel. A tesztelés során kizárólag az alapmodellt, illetve a modell_3-at használtuk, mivel a többi modell F-mértéke nem adott olyan jó eredményeket, mint a modell_3-nak. A modellek fent említett beállításain nem változtattunk, kizárólagos különbséget a más adaton való tesztelés jelentett. Az eredményeket a 3.3. táblázat tartalmazza, az előző tesztelésekkel való összehasonlítást, pedig a 3.4. táblázat.

[1] A táblázat accuracy, precision, recall, F-mérték soraiban százalékosan adtuk meg az értékeket.

Mérőszámok	Modellek	
	Alapmodell tesztadaton	**Modell_3 tesztadaton**
True positive	86	98
False positive	10	15
True negative	4796	4791
False negativ	108	96
Accuracy	97.64	97.78
Precision	**89.58**	86.73
Recall	44.33	**50.52**
F-score	59.31	**63.84**

3.3. táblázat. Tesztadat kiértékelése mérőszámokkal[1]

Mérőszámok	Modellek			
	Alapmodell	**Alapmodell tesztadaton**	**Modell_3**	**Modell_3 tesztadaton**
True positive	69	86	79	98
False positive	14	10	11	15
True negative	4827	4796	4830	4791
False negativ	90	108	80	96
Accuracy	97.92	97.64	98.18	97.78
Precision	83.13	**89.58**	87.78	86.73
Recall	43.40	44.33	49.69	**50.52**
F-score	57.02	59.31	63.45	**63.84**

3.4. táblázat. Modellek összehasonlítása[1]

A teszteléshez szintén 5000 soros adatot használtunk, hogy a validációs halmazon elért eredményeket össze tudjuk hasonlítani a kijövő eredményekkel. Az F-mértéket tekintve hasonlóan teljesítenek a modelljeink az új adaton, tehát a modelljeink értelmes predikciót adnak a tesztelések folyamán. Ha megvizsgáljuk az új adaton való tesztelés mérőszámait, akkor láthatjuk, hogy a toxikus szövegeket helyesebben prediktálja, viszont a nem toxikus szövegeknél több a félreosztályozott szöveg. A validációs halmaz eredményein ezt nem tapasztaltuk, mivel az ott félreosztályozott szövegekből állítottuk elő az új magyarázó változónkat. Emiatt a modellünket jobban igazítottuk a validációs halmazra, viszont nem túlzottan, hogy felmerüljön a *túltanulás* hibája. A túltanulás esetében a modellek nagyon jól teljesítenek a validációs halmazon, mivel rátanulnak az ott előforduló hibákra, viszont a teszt halmaz más adatokból áll, nem feltétlenül ugyanolyan hibákat tartalmaz, így ott sokkal rosszabbul fog teljesíteni ugyanaz a modell.

3.2. Kvalitatív eredmények

Ebben a részben a különböző modellek által félreosztályzott rekordokat elemezzük. A hibaelemzés során bemutatjuk, hogy a 2.4 részben bevezetett feature-ök, milyen szövegbeli példákkal támaszthatók alá.

Rekord	**Eredeti címke**	**Jósolt címke**
They are terrorists pure and simple.	toxikus	nem toxikus
Let's make Alaska as dumb as the south.	toxikus	nem toxikus
well, you are entitled to your delusions. and I am not surprised in the least you are proud of your liberal bigotry and shameless staggering hypocrisy.	toxikus	nem toxikus

3.5. táblázat. Példa rekordok feature-ök bevezetésére

Hibaelemzés során az elrontott félreosztályzott rekordok közül kiválasztottunk véletlenszerűen 100 rekordok, amiből 85 rekord eredetileg toxikus címkéjű és 15 nem toxikus címkéjű volt. A célunk, hogy a programunk minél több toxikus rekordot találjon el helyesen, így olyan összefüggéseket kerestünk, amik elsősorban a toxikus rekordok helyes osztályzását segítik elő. Az első észrevételünk, hogy a 85 félreosztályzott toxikus rekord közül 35 rekord a 3.5 táblázatban szereplő első két rekordokhoz hasonlóan egy rövid mondatból állt. Mivel a 85 rekord közül arányaiban sok rövid rekordot tartalmazott, emiatt bevezettük a number of words feature-t.

Miután a number of words feature bevezetése nem javított az alapmodellen, ahogy a 3.2 táblázat második oszlopa mutatja, így új összefüggéseket kerestünk a félreosztályzott rekordok között. Az előbb megvizsgált 35 rekordból, továbbá több mondatból álló rekordból (például a 3.5 táblázat 3. sora) azt a megfigyelést tettük, hogy kevés igét tartalmaznak. Megjegyeznénk, hogy az *am, are, is* létigék a *stopword* szólistájába szerepelnek, így például a 3.5 táblázat első sora nem tartalmazna igét, ha a POS tags feature-t használnák rá. A POS tags feature-t nem csak igére, hanem a többi alapszófaj leszámlálására is létrehoztuk. Más szófajokat azért számlálunk le az ige mellett, mert ha kevés igét tartalmaz az adott rekord, akkor más szófajból többet, és az is szolgálhat információ értékkel. Viszont a kijövő kvantitatív eredmények miatt a POS tags változó megtartását elvetettük, így más szempontok alapján kezdtük el a félreosztályozott toxikus rekordokat vizsgálni.

Mivel logisztikus regressziós modellt használunk a tanítás során, így a feature-ök súlyát le tudtuk kérdezni a modellből, és a súly-feature kapcsolatát kezdtük el vizsgálni. Jelöljük a súlyokat β-val. A β értékül felvehet

nullánál nagyobb, kisebb számot vagy pontosan nullát is. Ha a β nullával egyenlő, akkor az adott magyarázó változó elhanyagolható, mivel sem a 0-ás, sem az 1-es osztályba való tartozást nem fogja erősíteni. Ha a β pozitív értéket vesz fel, akkor növeli az 1-es osztályba tartozás valószínűséget a 0-ással szemben. Minél nagyobb pozitív értéket vesz fel a β, annál jobban növeli az 1-es osztályba tartozás valószínűségét a 0-ással szemben. Ha a β negatív értékkel egyenlő, az előbb leírtakhoz képest ellentétes változás figyelhető meg. Minél kisebb negatív szám a β, annál jobban csökkenti az 1-es osztályba tartozás valószínűségét és növeli a 0-ásba tartozását. Esetünkben a toxikus osztályba tartozás valószínűségét a nagy pozitív értékű súlyok adják, és az ezekhez tartozó feature-ök számítanak fontos szavaknak.

A súlyok értékeinek az ismeretében a félreosztályzott 85 toxikus rekordról a következő megfigyeléseket tettük:

1. Öt esetben fordult elő, hogy egy toxikus szövegben lévő offenzív szó (például *nigger* szó a 3.6 táblázat 1. és 2. sorából) nem került a feture-ök közé.[2]

2. Az elrontott, nem toxikusnak osztályzott szövegeknél a bűncselekménnyel kapcsolatos szavak (például *gun*, *jail*, *charge* szavak a 3.6 táblázat 3. és 4. sorából) kis súlyú feature-nek számítottak, viszont 15 félreosztályzott toxikus rekordnál szerepeltek.

3. 20 félreosztályozott toxikus szövegben hiába szerepelt nagy súllyal rendelkező offenzív szó (például *silly* vagy *troll* szavak a 3.6 táblázat 5. és 6. sorából), mégsem volt elegendő, hogy a modell helyesen osztályozza.

4. A rekordok között volt két ironikus szöveg is, amik ironikusan de tartalmazták *thank you* kifejezést (a 3.6 táblázat 7. és 8. sora).

[2]Fontos megjegyezni, hogy például a nigger szó azért nem került a feature-ök közé, mert a tanító adatban nem szerepelt ilyen szó, viszont a validációs halmazon igen. Mivel fontos toxikus szónak tartottuk, így a gazetteer változóba bevettük.

Sorszám	Rekord	Eredeti címke	Jósolt címke
1	*kill all the niggers and pedophiles!*	toxikus	nem toxikus
2	*NIGGER*	toxikus	nem toxikus
3	*It isn't funny if you work at the jail and have to clean up after the jerk. I don't figure a judge is going to take kindly to his antics. I wonder if he still thinks all of that was a great idea now that he is, presumably, sober?*	toxikus	nem toxikus
4	*Shoot my dog if he is charging you with teeth bared ! The home owner is 100% wrong blaming Mr. Mellerstig for shooting his dog when he charged at him with teeth bared. Mr. Brailey is negligent having an invisible fence- which is INVISIBLE to someone walking by- all they see is the threat of a charging snarling dog. His body was half outside the property line so that tells you it did NOT stop him. I have 2 huge black dogs inside a fence with 6 ft. gates. They will bark and run the fence to let you know they are standing their ground. If one were to get out and charge at you barking I would not blame you if you shot him. If you had a child with you, I would be remorseful that my dog caused you to fear for your child. To the people that get so bent out of shape over a dog being shot- what would you do if a MOOSE were charging you and your child with his head down, ears back- no doubt what his attempt was-to mow you down and stomp you. You would shoot him...would you not ?*	toxikus	nem toxikus
5	*...dirty energy... speculators..." – yeah, yeah, yeah. What a silly world these people live in. Iḿ surprised that North Dakota and other energy-producing states have not filed interstate-commerce lawsuits against Portland and Oregon. Maybe their legislatures should adopt anti-Oregon boycotts like our state and city are so fond of doing.*	toxikus	nem toxikus
6	*Glorious troll fail!*	toxikus	nem toxikus
7	*I guess I would say that you, as a man, haven't had to deal with systematic oppression based on your gender and that you are part of the dominant group and therefore have the luxury of thinking that gender doesn't play a role in anything. And thank you for telling me who I want to vote for! I won't say it was your testicles that helped you do it, but I don't think they hurt in this situation either*	toxikus	nem toxikus
8	*I'm going to smoke pot just to make this fair. Ten years ago we had just as many stoners as we do today. DUH. For decades the police have had to be trained for impairment, which included pot. DUH. We have spent hundreds of billions of dollars on prohibition that empowered the black market, and caused crime to increase. yet a simpleton like you fails to grasp youre profound stupidity has lost the war on drugs, people no longer buy into your refer madness lies/propaganda, and it totally sucks to be you. On behalf of the stoners of Oregon, thank you for your ability to continue to humiliate yourself in public for our entertainment*	toxikus	nem toxikus

3.6. táblázat. Példa rekordok

Az előbb felsorolt pontokban rosszul osztályzott rekordok kiküszöbölésére vezettük be a *gazetteer* feature-t. A 2.4 részben a változó működését részlete-

sen kifejtettük, viszont fontosnak tartjuk ebben a részben is kihangsúlyozni, hogy a változó szólistájába olyan szavak is kerültek, amik kizárólag a validációs halmazon szerepeltek. Mivel a kiértékelést és a hibaelemzést a validációs halmazon végeztük, így a validációs halmazon elrontott rekordokból vontunk le következtetéseket. Emiatt a 3.2 táblázatban lévő modell_3 mérőszámai minden esetben jobb eredményt adtak, mint az alapmodell mérőszámai, hiszen a validációs halmaz egyedibb hibáit megtanulta kijavítani. Ellenben a modell túltanulásáról nem beszélhetünk, mivel a 3.4 táblázat eredményei hasonló értékeket adtak a tesztelés folyamán, nem rosszabbakat, mint ami a túltanult modelleket jellemzi.

A 3.1 részben a 3.2 táblázat megmutatta a modellek TP,FN,TN,FP értékeit, viszont ezen mérőszámokból nem láttuk, hogy egyes szövegek hogyan kaptak helyes címkét, amíg más szövegek miként osztályozódtak félre a modell_3-ban. A következőkben a tévesztési mátrix mérőszámainak a változásait vizsgáljuk meg részletesebben a két fő modellünknél: az alapmodellnél és a modell_3-nál.

A mérőszámok változásánál négy esetet különböztetünk meg egymástól, ahol az 1-es címke a toxikus, a 0-ás címke a nem toxikus szövegeket jelöli:

1. A szöveg 1-es címkéjű, viszont az alapmodell 0-ásnak, a modell_3 1-esnek osztályozza. Az FN-ből TP-be kerülnek át az ilyen szövegek.

2. A szöveg 0-ás címkéjű, viszont az alapmodell 1-esnek, a modell_3 0-ásnak osztályozza. Az FP-ből TN-be kerülnek át az ilyen szövegek.

3. A szöveg 1-es címkéjű, viszont a modell_3 0-ásnak, az alapmodell 1-esnek osztályozza. Az TP-ből FN-be kerülnek át az ilyen szövegek.

4. A szöveg 0-ás címkéjű, viszont a modell_3 1-esnek, az alapmodell 0-ásnak osztályozza. Az TN-ből FP-be kerülnek át az ilyen szövegek.

A felsoroltak szerint megvizsgáltuk a két modellt és a következő mennyiségeket kaptuk: az 5000 soros validációs halmazon az 1. eset 13, a 2. eset 6, a 3.-4. esetek 3-3 szövegnél fordultak elő. Vezessük le például a 3.2 táblázatban szereplő alapmodell TP értékéből a modell_3 TP értékét az előbb felsorolt számszerűsített változások függvényében. Az alapmodell eredetileg 69 TP szöveget tartalmazott, az új modell másik 13 szöveget osztályozott helyesen, viszont a 69 szövegből 3-at elrontott, így $69 + 13 - 3 = 79$, ami pontosan az új modell TP értéke. A táblázatban szereplő többi értéket hasonló gondolatmenetű levezetéssel meg lehet kapni.

A fent felsoroltak szemléletes bemutatása érdekében, a következőkben néhány példát mutatunk be, ahol arra is kitérünk, hogy a modelleket mi befolyásolta a döntéshozatalukban.

Rekord	Features	Alapmodellbeli súly
Strawman argument. We're talking about the NRA trying to force the University of Alaska to allow carry concealed weapons on campuses. This is a decision for the University Board of Regents to make, not the gun nuts in the Alaska legislature, who hypocritically won't allow guns in the state capitol building.	weapon, gun	0.53, 0.2
No spin, overwhelmingly christians in this nation are the ones creating divisions between people and using their faith as a weapon in order to oppress minority groups they deem inferior. They rightly deserve the heap of blame, Prevo, Minnery and their ilk are the ones opposed to AO96 here in Anchorage and would like to see our LGBT friends and neighbors fired, evicted and denied economic services all because the wizards who interpret their magic spell book say they are bad. Religion may have served a useful purpose at some point in our past but has long since outlived its usefulness.	weapon, fire	0.53, 0.1

3.7. táblázat. Példa FN-ből TP-be kerülő rekordokra

A 3.7 táblázatban két példa látható, amiket az alapmodell helytelenül nem toxikusnak, viszont a modell_3 már helyesen toxikusnak osztályozott, tehát egy FN-ből TP-be történő változás ment végbe. A táblázat szövegeiben kiemeltük a feature szavait, amiknek az alapmodellben szereplő súlyaik az utolsó oszlopban szerepelnek. Ezek a szavak kis súllyal szerepeltek az eredeti modellben, viszont a megfigyeléseink közé tartozott, hogy a bűncselekménnyel kapcsolatos szavak gyakran előfordulnak toxikus szövegekben, így fontosabb szavaknak kellene azokat kezelni. Emiatt a kiemelt szavakat a gazetteer változó szólistájába tettük, ennek köszönhetően a modell fontos szavaknak tekintette azokat, így az új modellben már helyesen osztályozta a táblázatban szereplő rekordokat. A modell_3-ban a gazetteer változó 2.6 súllyal szerepel, ami szintén alátámasztja, hogy az új változó egy fontos feature-t határoz meg, aminek segítségével pontosabb predikciót ad a modellünk.

4. fejezet

Konklúzió

A szakdolgozat folyamán a toxikus szövegek detektálás egyik módszerével ismerkedhettünk meg. A nyers szövegektől eljutottunk a tényleges osztályozási feladathoz. A fejezetek során pontosabb ismereteket szerezhettünk a felügyelt tanulásról, logisztikus regresszióról, illetve a kiértékeléshez szükséges mérőszámokról. Továbbá bemutattuk a szövegfeldolgozás és a modell tanítás lépéseit is.

A dolgozat során implementáltunk egy bináris osztályozási feladatot a megadott adathalmazon. A feladatot nehezítette, hogy nem egy előkészített adaton dolgoztunk, hanem először saját magunknak kellett feltérképeznünk az adatot és a feladatnak megfelelően kiválasztani a szükséges változókat. Továbbá a felhasznált adatot nem bináris osztályozásra készítették, így a magyarázó változó értékeit binárissá kellett alakítanunk.

A szöveges adattal dolgozás egyik kihívása a szövegek számokká konvertálása volt, hiszen a modelljeink kizárólag szám adaton képesek dolgozni. Ezen átalakítást ismert N-gram modellel értük el. Az N-gram feature-jei adták a kiindulási adathalmazt a tanítás folyamán, és az így betanított modellt neveztük alapmodellnek. Az alapmodellt logisztikus regressziós modellen tanítottuk, amit kiértékelve 57.02%-os F-mértéket eredményezett. A modell eredményével nem voltunk megelégedve, így újabb kísérleteket végeztünk. A félreosztályzott rekordokat vizsgáltuk, milyen hasonló hibákat követ el a modell újra és újra, és ezen hibák kijavítása érdekében új attribútumokat generáltunk, mint például number of words, POS tags vagy gazetteer feature-öket. Az új feature-ök közül a gazetteer bevezetésével értünk el javulást, és végül a tesztadaton 63.84%-os F-mértéket eredményezett.

A szakdolgozat témájának kitűzött feladatot gépi tanulással oldottuk meg, viszont a [4] cikkben is olvashattuk, hogy osztályozási feladatok neurális hálós modellekkel is megoldhatóak. Jogosan merülhet fel a kérdés, hogy miért nem neurális hálókkal oldottuk meg a kitűzött feladatot. A neurális hálót

gyakran a szakirodalomban fekete dobozként írják le. A neurális háló döntési folyamatát nehezen lehet nyomon követni, sokkal nehezebben értelmezhetőek és elemezhetőek az ilyen modellek, ezáltal a szakdolgozat folyamán végrehajtott hibaelemzés módszerét nem lehet *mélytanulási* (*angolul deep learning*) modellekkel megoldani. A hibaelemzést súlyok értékei alapján végeztük és a súlyok függvényében konstruáltunk új feature-öket, amiket ilyen formában a neurális hálókkal nem tudtunk volna kivitelezni, így mindenképp gépi tanulás módszerével kellett megoldani a toxikus szövegek detektálását.

A jövőre tekintve a szakdolgozatban leírt eredményeket tovább lehet javítani a hibaelemzés folytatásával, új szabályok keresésével és ezáltal új feature-ök bevezetésével.

Irodalomjegyzék

[1] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, pages 491–500, 2019.

[2] Louise Qianying Huang and Mi Jeremy Yu. Building unbiased comment toxicity classification model with natural language processing.

[3] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[4] Mujahed A Saif, Alexander N Medvedev, Maxim A Medvedev, and Todorka Atanasova. Classification of online toxic comments using the logistic regression and neural networks models. In *AIP conference proceedings*, volume 2048, page 060011. AIP Publishing LLC, 2018.

[5] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[6] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 189–194. IEEE, 2014.

[7] Viera Maslej-Krešňáková, Martin Sarnovskỳ, Peter Butka, and Kristína Machová. Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification. *Applied Sciences*, 10(23):8631, 2020.

[8] George Bebis and Michael Georgiopoulos. Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31, 1994.

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[10] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[11] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[12] Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. *arXiv preprint arXiv:1809.08651*, 2018.

[13] Shan Suthaharan. Support vector machine. In *Machine learning models and algorithms for big data classification*, pages 207–235. Springer, 2016.

[14] Antonio Toral and Rafael Munoz. A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia. In *Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*, 2006.

[15] Andrei Mikheev, Marc Moens, and Claire Grover. Named entity recognition without gazetteers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999.

[16] Buza Krisztián Bodon Ferenc. *Adatbányászat*. 2014.

[17] Mark Tranmer and Mark Elliot. Binary logistic regression. *Cathie Marsh for census and survey research, paper*, 20, 2008.

[18] Sida I Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, 2012.

[19] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[20] Scott A Czepiel. Maximum likelihood estimation of logistic regression models: theory and implementation. *Available at czep. net/stat/mlelr. pdf*, pages 1825252548–1564645290, 2002.

[21] Gábor Hámori. *Predikciós célú klasszifikáló statisztikai modellek gyakorlati kérdései*. PhD thesis, Kaposvári Egyetem, 2015.

[22] Nancy A Obuchowski. Roc analysis. *American Journal of Roentgenology*, 184(2):364–372, 2005.