Adatbányászat: Sentiment analysis on Reddit dataset

Forman Balázs Attila és Michaletzky Tamás Vilmos

2021

Bevezetés

Az adathalmaz jellemzése

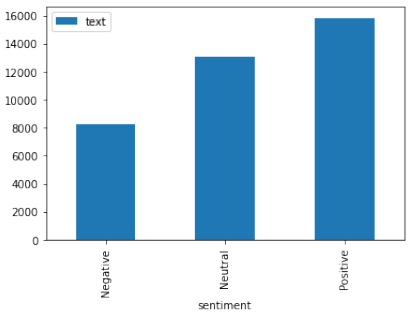
Az általunk választott adathalmaz a **Kaggle**-ről származik, és 37249 darab **Reddit** bejegyzés szerepelt benne, egy-egy hozzájuk rendelt -1, 0, és 1 számmal, annak megfelelően, hogy az adott bejegyzés gyűlöletbeszédet tartalmazott (negatív szentiment), semleges, vagy pozitív jelentéstartalommal bírt (dicsérő, elfogadó szöveg).

A halmaz eleve egészen jól meg volt tisztítva, kevés volt benne az üres bejegyzés (csak 100 darab olyan volt, ami nem tartalmazott szöveget), nem voltak bennük emojik és minden kisbetűs volt.

	text	score
0	family mormon have never tried explain them t...	1
1	buddhism has very much lot compatible with chr...	1
2	seriously don say thing first all they won get...	-1
3	what you have learned yours and only yours wha...	0
4	for your own benefit you may want read living ...	1
5	you should all sit down together and watch the...	-1
6	was teens when discovered zen meditation was ...	1
7	jesus was zen meets jew	0
8	there are two varieties christians dogmatic th...	-1
9	dont worry about trying explain yourself just ...	1

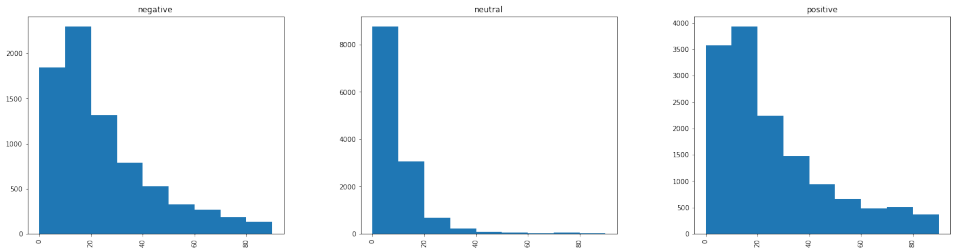
1. ábra. Az adathalmaz első 10 sora.

A negatív, semleges és pozitív tartalmú bejegyzések eloszlása kiegyensúlyozatlan volt, összesen 8277 negatív, 13042 semleges és 15830 pozitív tartalmút találtunk.



2. ábra. Kétszer annyi pozitív bejegyzés van, mint negatív.

A bejegyzések hosszai érdekes módon gyengén összefüggtek a tartalmukkal, bár mindegyikre jellemző, hogy rövidebbekből több volt. Míg a semleges bejegyzéseknél a leggyakoribb hossz a 10 szónál rövidebb, a pozitív és a negatív tartalmú bejegyzések jellemzően kicsit hosszabbak voltak.



3. ábra. A szavak számának eloszlása a 3 osztályban.

A bejegyzéseket szófelhőben ábrázolva (Appendix 10. ábra), jelentéstartalmuk szerint csoportosítva azt látjuk, hogy a leggyakoribb szavak viszonylag függetlenek attól, hogy milyen jelentésű (pozitív, negatív vagy semleges).

Az adatok tisztítása

Eltávolítottuk a 100 hiányzó szövegű bejegyzést, a címkéket pedig áttranszformáltuk a nemnegatív tartományba:

pozitív	0
negatív	1
semleges	2

Ilyen módon könnyű lesz a **binary–multilabel classification** tesztet elvégezni, hiszen csak a 2-es címkét kell majd eltávolítani.

Az így kapott adathalmaz 3 oszlopa a következő típusú volt, ebből a categoryt nem használtuk.

text	string
category	int
label	int

A probléma megfogalmazása, hipotézisek

Azt a feladatot akarjuk megoldani, hogy különböző algoritmusokkal felismerjük egy bejegyzésről, hogy milyen tartalmú (*szentiment analízis*).

Mivel már maga a feladat rosszul definiált, ezért igazán jó eredményt nem várunk egyik algoritmustól sem. Néha maguk az emberek számára is nehéz megmondani, hogy egy adott szöveg milyen tartalommal bír, arról nem is beszélve, hogy mennyit számít a kontextus, az irónia, esetleg az emojik, amiket mind-mind kénytelenek vagyunk figyelmen kívül hagyni. Persze, ha ezeket mind ismernénk se biztos, hogy mindenkinek minden bejegyzés azonos jelentéstartalommal bírna, így az, hogy most milyen adatokkal dolgozunk, valószínűleg az is vitatható lenne.

További probléma, hogy viszonylag kevés negatív tartalmú bejegyzés van (fele annyi, mint pozitív), így azt a célját, hogy gyűlöletbeszédet ismerjen fel az algoritmus, valószínűleg nem fogja tudni maximálisan elérni.Mivel a semlegesek nagyon rövidek, azért lehet, hogy azokat meg azért nem tudjuk majd jól osztályozni, hiszen egyszerűen kevés az adat benne. Mindezek következtében valószínűleg a pozitívak felé fog eltolódni a jóslás.

Nehezíti még a jóslást, hogy nem a gyakori kulcsszavakban különböznek, ahogy az a szófelhőkből is kirajzolódik.

5-féle szóbeágyazást fogunk tesztelni.

- TfIdf
- word2vec:
- doc2vec
- GloVe
- Bert

Ezek az alábbi python-csomagokból érhetők el:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models.word2vec import Word2Vec
from gensim.models.doc2vec import Doc2Vec
from transformers import BertModel, BertForSequenceClassification
```

A GloVe egy nagy fájl, melyet az alábbi címről szedtünk le.

Az összehasonlítás érdekében (a Bertet kivéve) mindegyikre logisztikus regressziót eresztünk, és így hasonlítjuk majd össze a kapott eredményeket. Ezután majd külön a GloVe-on, mint legígéretesebb módszeren más klasszifikáló eljárásokat is finomhangolunk, így pl. **LSTM**-et is megnézünk majd.

A teljesítményt háromféle metrikában fogjuk nézni, ezek a **f1-score, accuracy, roc-auc-score** lesznek. Az f1– és roc-auc–score-okat „weighted” súlyozással vesszük multiclass esetben.

Az alábbi hipotéziseket fogjuk tesztelni:

1. A modelleket ismerve azt várjuk, hogy a következő sorrend áll fenn a modellek jóságára:

$$\texttt{TfIdf} < \texttt{word2vec} < \texttt{doc2vec} < \texttt{GloVe} < \texttt{Bert}.$$

2. A binary modell (neutrális bejegyzések eldobásával) szignifikánsan jobb lesz, mint a multiclass.
3. Várhatóan a pozitív szentiment felé tolódik el majd mindegyik modell.
4. A beágyazás dimenzójáról:
   - Tfidf: valószínű nagy dimenziós beágyazásra lesz szükség
   - neurális háló alapú beágyazások (word2vec, doc2vec, GloVe): közepes dimenzió is elég kell legyen.
5. Túltanulás félelme: a feladat rosszul definiáltsága miatt könnyen lehet, hogy a modellek általánosító képessége gyenge lesz, és csak „rátanul" a tanító halmazra.

Célunk az NLP-ben előforduló szóbeágyazások megismerése és kipróbálása volt a teljesség igénye nélkül, az itt szereplő eredményeket ugyanis még rengeteg felé tovább lehet hangolni, gondolni, javítani. A programozás **Python** nyelven történt, **Anaconda** környezetben és **Google Colab** szervereken egyaránt.

Az eredményeink, notebookok és kódok elérhetőek egy nyilvános **GitHub** repositoryban.

Szöveg előfeldolgozás

A szöveget az összehasonlítás érdekében egységesíteni kellett, ehhez pedig első lépésként minden átírtunk kisbetűsre (bár ránézésre ez már alapból így is volt, de biztos ami biztos). Ezután kidobtuk a nem a latin abc-ben szereplő karaktereket, szétvágtuk a bejegyzéseket szavakra, és elhagytuk a kötőszavakat (stopwords). Végül pedig egységesítettük a főnevek és az igék különböző alakjait, mindegyikből csak a szótöveket véve. A lépéseket javarészt **nltk-toolkit** python-csomaggal végeztük el.

Így lett az alábbi szövegből a következő lista:

- Original tweet: family mormon have never tried explain them they still stare puzzled from time time like some kind strange creature nonetheless they have come admire for the patience calmness equanimity acceptance and compassion have developed all the things buddhism teaches
- Processed tweet: ['famili', 'mormon', 'never', 'tri', 'explain', 'still', 'stare', 'puzzl', 'time', 'time', 'like', 'kind', 'strang', 'creatur', 'nonetheless', 'come', 'admir', 'patienc', 'calm', 'equanim', 'accept', 'compass', 'develop', 'thing', 'buddhism', 'teach']

Algoritmusok, mérések

Nézzük először az algoritmusokat hogyan hangoltuk finomra!

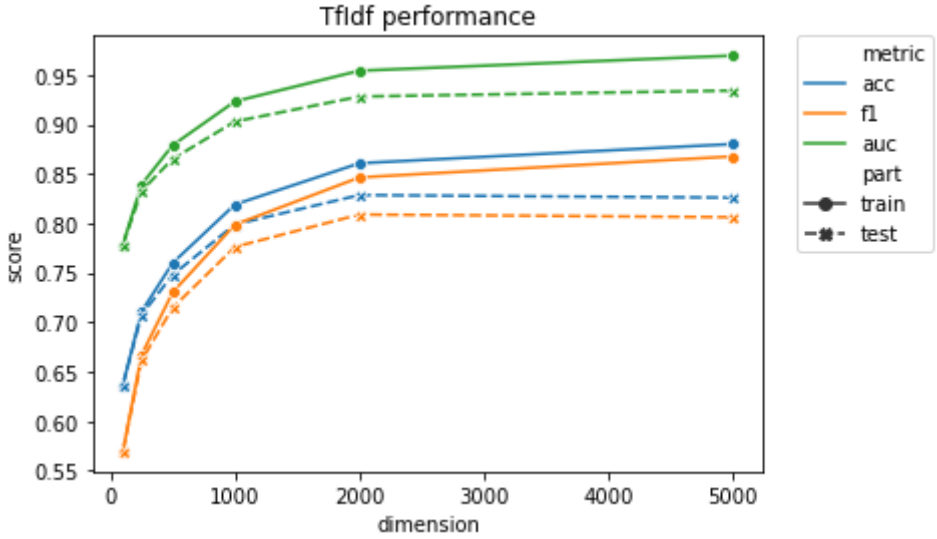
TfIdf

A $TfIdf$, a term-frequency, inverse-document-frequency szavakból képzett mozaikszó, az egyik legrégebbi szövegfeldolgozó algoritmus, mely a korpusz (a dokumentumok összessége) szavait egy általunk megadott dimenziós térbe képezi az alábbi két mérőszám alapján:

- term-frequency: megadja, hogy egy adott milyen gyakori az adott dokumentumban
- inverse-document-frequency: megadja, hogy egy adott szó milyen gyakori az adott korpuszban.

Ezen információk segítségével már hatékonyan elkódolhatóak a korpusz szavai. (Minden dimenzió egy szó lesz, az érték pedig a rá számolt tfidf-score. A dimenzió növelésével így egyre több szót használunk fel.)

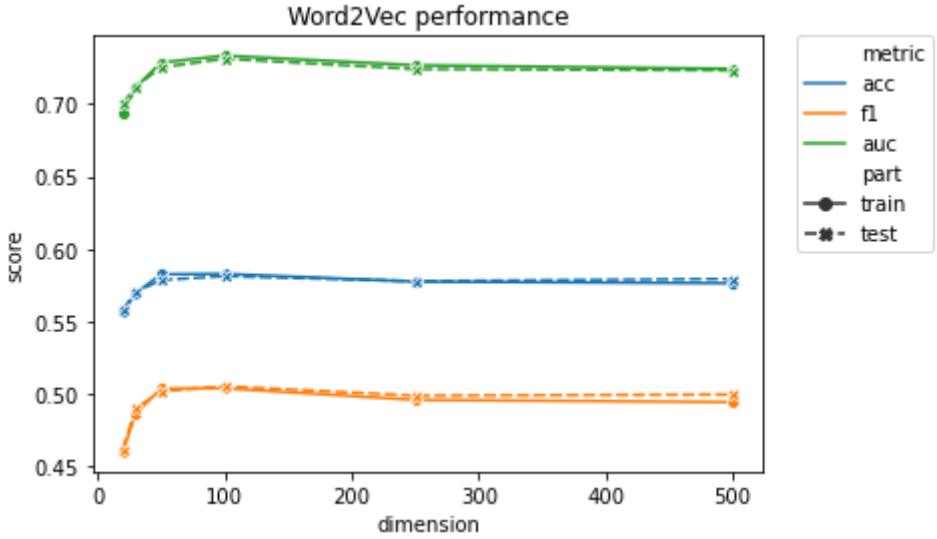
A logisztikus regresszióval kapott eredményeket az alábbi ábra szemlélteti. Látszik, hogy minél nagyobb dimenziós vektort rendelünk egy-egy bejegyzéshez, annál inkább nő mindhárom metrikában az eredmény, ám a 2000-ről 5000-re ugrás már nem tűnik indokoltnak (arról nem is beszélve, hogy egyre jobban nő a számolási költség). A 100 legfontosabb szót a 12. ábra mutatja. Látszik, hogy a különböző népcsoportok (muszlim, hindu), és politikához kötődő szavak előkelő helyen állnak a gyűlöletbeszédben.



Word2Vec

A $Word2Vec$ egy neurális hálót használ arra, hogy beágyazza a szavakat egy általunk kiválasztott dimenziós vektortérbe, oly módon, hogy a hasonló tartalmú szavak közel kerüljenek egymáshoz. A bejegyzésekhez ekkor megint az őt tartalmazó szavak átlagát rendelhetjük.

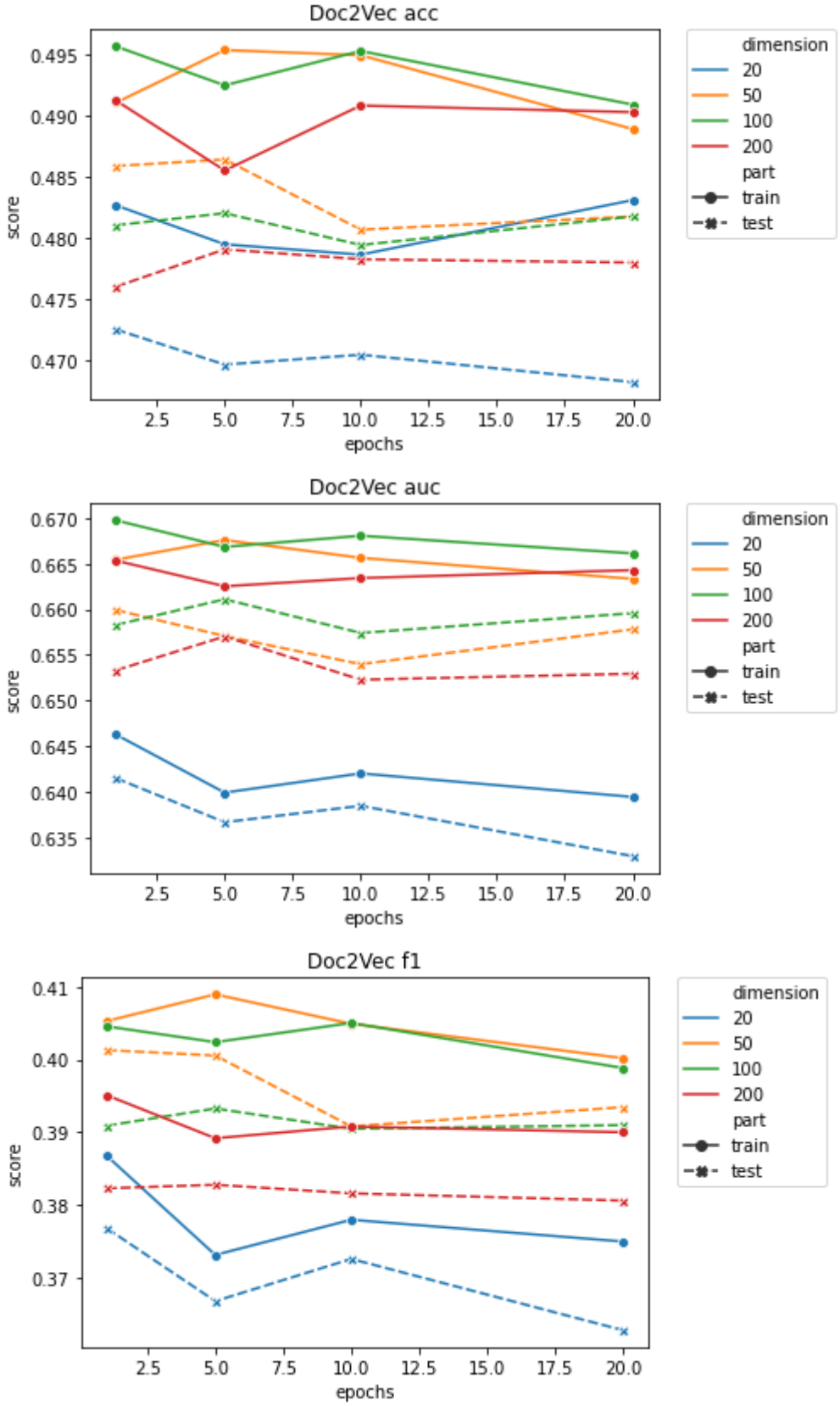
A logisztikus regresszióval kiértékelt eredmények most már érdekesebben alakultak: azt látjuk, hogy a dimenzió drasztikus növelésével csökken a modell magyarázóképessége, az optimálisnak az 50 dimenzió mutatkozott.



Doc2Vec

A $Doc2Vec$ a $Word2Vec$-hez hasonló, de itt már magukhoz a bejegyzésekhez igyekszik vektort rendelni olyan módon, hogy a hasonló tartalmú bejegyzések kerüljenek közel egymáshoz. A beágyazások hatékonyságát több tanulási cikluson keresztül javítva próbálni növelni.

Mivel itt van egy új paraméter, az *epoch*-szám, 3 különböző grafikonon ábrázoljuk a metrikákat. Struktúrájukat tekintve mindhárom grafikon nagyon hasonló, legjobbnak az 5 epochon tanuló 50 dimenziós beágyazás tűnik.



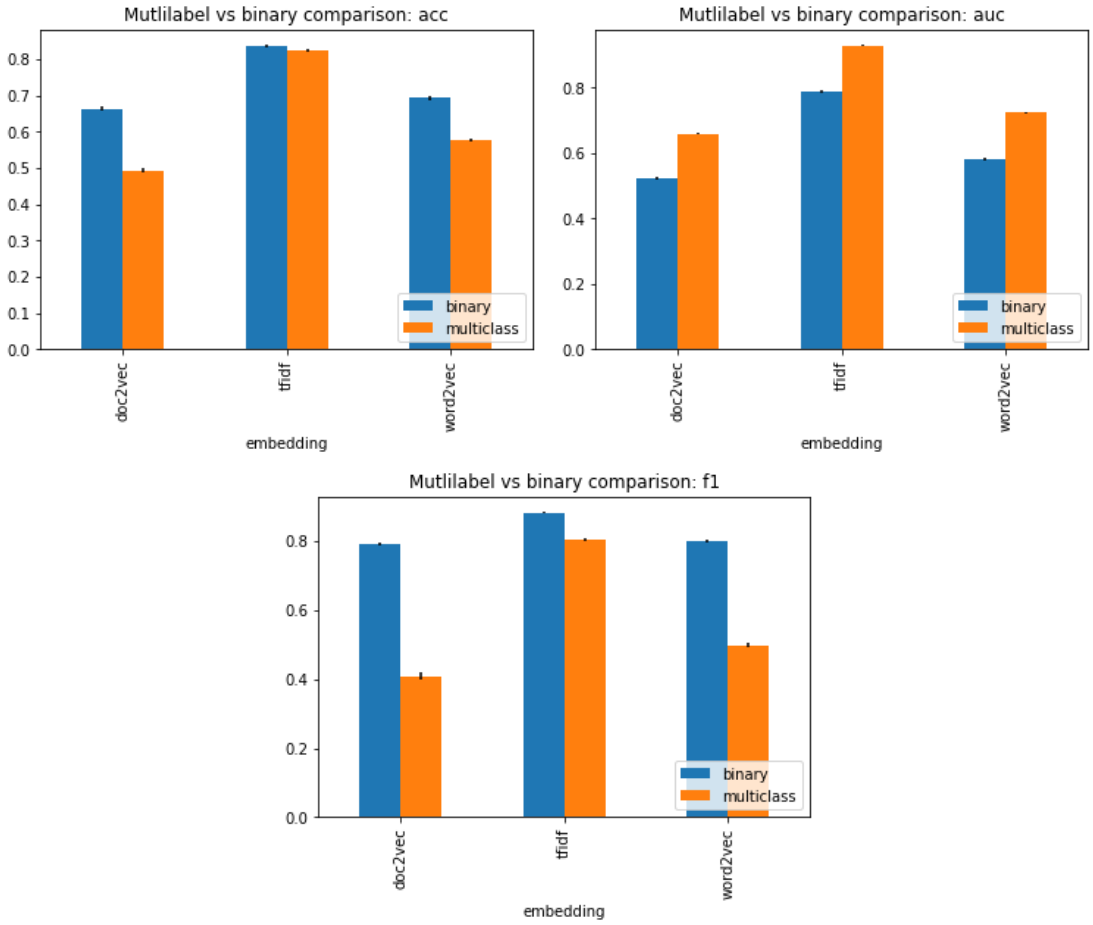
4. ábra. A Doc2Vec beágyazás train-test performanicája különböző epoch számoknál.

A módszerek multi–binary összehasonlítása

Az alábbi konfigurációk bizonyultak a legjobbnak:

modell	dimenzió
tfidf	2000
word2vec	50
doc2vec	50 (5 epoch)

A legjobb konfigurációkat binary klasszifikálásra is lefuttatva, 5 különböző train-test vágásra átlagolva errorbarral, metrikánként külön plotolva az alábbi eredmények mutatkoznak.



5. ábra. Binary vs multilabel klasszifikáció.

Meglepő módon a legrégebbi módszer bizonyult a legjobbnak, a két neurális háló alapú beágyazás között nem észlelhető szignifikáns különbség. Az eredményekről a dolgozat végén mellékeljük a táblázatot is (11. ábra).

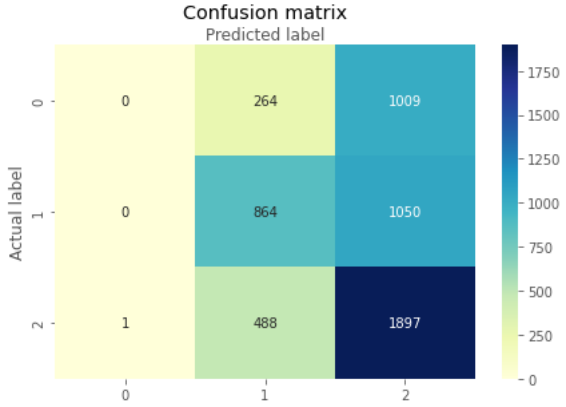
Érdekes, hogy az auc score-oknál a multiclass az eset a jobb. Valószínűleg azonban ez az átlagolás módszere miatt alakulhatott így.

Most nézzük a két üdvöskét, a **GloVe**-ot és a **Bert**et!

GloVe

A *GloVe* a harmadik neurális hálós beágyazásunk, mely a *Word2Vec*-hez hasonlóan szintén megint a szavakat ágyazza be 50, 100, 200 vagy 300 dimenzióba, és ezek mentén átlagolva helyezi el a szövegeket ugyanezen vektortérben. Ez azonban nagyobb adathalmazzal dolgozik, a szótárban szereplő szavak súlyait egy nagyméretű fájlból kellett elérni.

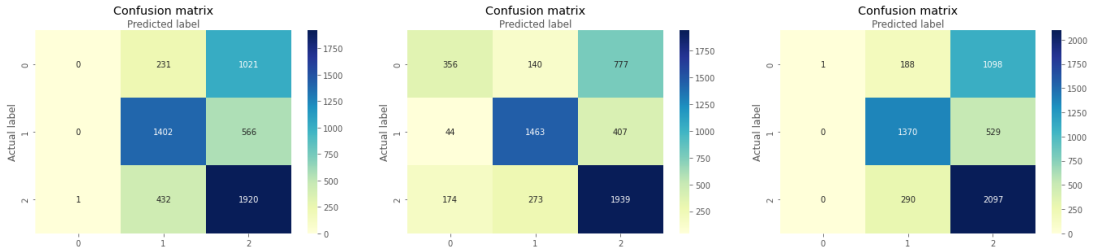
Logisztikus regresszióval az alábbi eredményt sikerült elérni:



GloVe és LSTM

Az LSTM technikát eredetileg idősorok neurális hálókkal való kezelésére találták ki, ám hamar kiderült, hogy a nyelvfeldogozásban is hatékony szerepet játszhatnak (mint ahogy pl. a Bert esetében majd láthatjuk). Az LSTM cella egy speciális háló-struktúrájú neurális modell, melynek különlegessége, hogy van egy úgynevezett *hidden state*-je, melyet a futtatás alatt mindig magunkkal „cipelünk". Ebben tárolódik el reményünk szerint az „információ". (Tehát nem csak a neurális háló súlyait változtatjuk minden epochban, de ez a plusz információ segítheti a klasszifikációt.)

Több különböző dimenziónál is tanítottuk a *GloVe*-ot, az következő tévesztési mátrixokat kaptuk:



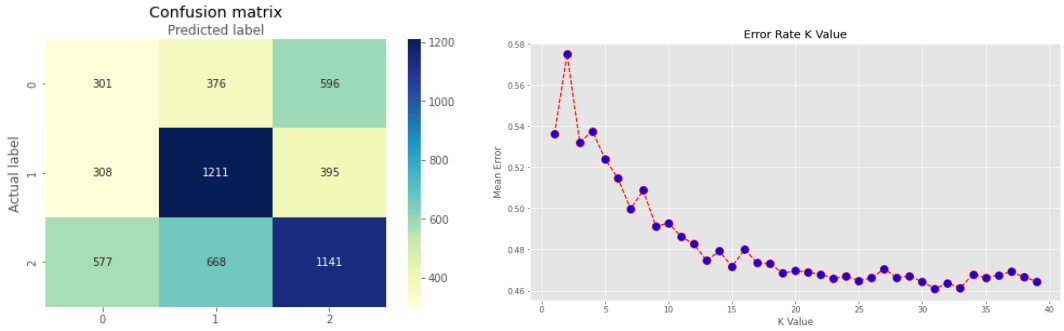
Azt láthatjuk, hogy a mi dokumentumaink között nincsenek időbeli összefüggések (vagy bármilyen olyanok, amikből az LSTM struktúra profitálhatna), így nem kapunk jobb eredményt a base modellnél.

GloVe és a legközelebbi szomszédok

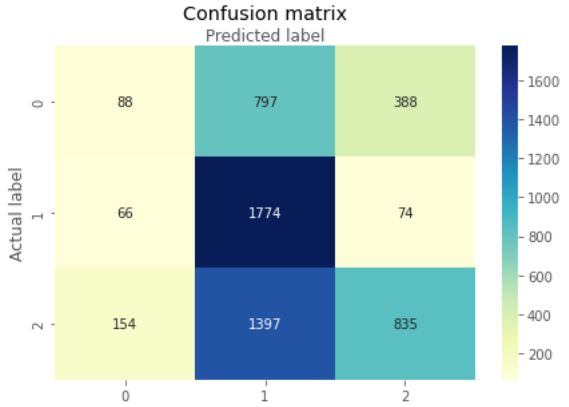
Lefuttattuk 100 dimenzióra a legközelebbi szomszédok klasszifikáló algoritmust is, úgy, hogy a 100 dimenzióba beágyazott vektorok mindegyikéhez megnéztük, hogy mi az 5 legközelebb lévő már ismert jelentéstartalmú bejegyzés, majd megnéztük hogy k szomszédra hogyan változik a mérés jóslás hibája.

GloVe és Bernoulli

A 100 dimenziós *GloVe*-ra lefuttattuk a Bernoulli Naive Bayes algoritmust is, ami a következő tévesztési mátrixot adta:



6. ábra. A kNN hibája különböző szomszédszámra.

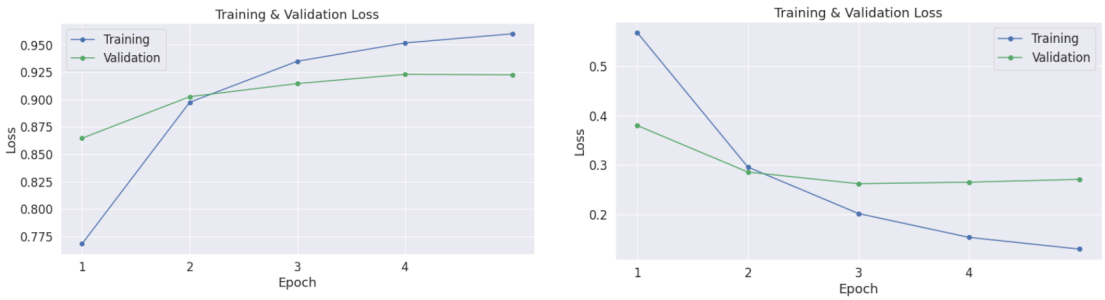


7. ábra. A NaiveBayes algoritmus tévesztési mátrixa.

Bert

Végül megnéztük a terület üdvösékjének számító **Bert**-et, mely egy Bi-LSTM, és jelenleg az egyik leghatékonyabb modell az NLP problémákra. A Bertet eredetileg a Google tudósai tanították a Wikipédia korpuszán, és a kétirányú LSTM struktúrát kihasználva értek el eredményeket. Mint a GloVe-nak, ennek is van egy adott (nagy elemszámú) szótára, minden szótáron kívüli elem a 100-as, UNK tokent kapja. A Bert-et jelenleg leghatékonyabban (és legkönnyebben) **PyTorch** környezetben lehet futtatni a **HuggingFace**-ről letöltött modelleken. Mi az eredeti $bert - base$, 12 rejtett rétegű, 1024 dimenziós Bert helyett memória-problémák miatt (a $bert-base$ nem fér be a GPU memóriájába) az ún. $bert-medium$-ot használtuk, mely 8 db, egyenként 512 dimenziós rétegen keresztül ágyazza be a szöveget. A klasszifikálást a HuggingFace-en elérhető **BertForSequenceClassification** neurális modellel hajtottuk végre, mely a $bert - medium$-ra egy fully-connected layert helyez klasszifikálás céljából.

Az eredményeket az alábbi ábra szemlélteti. A mérés egy 8Gb-s nvidia gpu-n nagyjából 10 percig futott. Azt látjuk, hogy 2 epochnál megegyezik a tanító és teszt halmazon a pontosság és veszteség



8. ábra. A BertClassifier accuracy és loss plotjai.

is, utána pedig szignifikáns javulás már nem is történik a teszt halmazon (túltanulás). Ha nem vagyunk magabiztosak, 3 epochnál akkor sem érdemes tovább tanítani.

Nézzük a többi metrikát! Azt látjuk, hogy az eddigi legjobb modellt kaptuk, mindössze 3 epoch

	precision	recall	f1-score	support
0	0.87	0.86	0.87	2550
1	0.92	0.93	0.93	4691
2	0.96	0.96	0.96	3903
accuracy			0.92	11144
macro avg	0.92	0.92	0.92	11144
weighted avg	0.92	0.92	0.92	11144

	precision	recall	f1-score	support
0	0.63	0.82	0.71	1874
1	0.85	0.84	0.84	4743
2	0.93	0.82	0.87	4528
accuracy			0.83	11145
macro avg	0.80	0.83	0.81	11145
weighted avg	0.84	0.83	0.83	11145

9. ábra. Balra a BertClassifier metrikái, jobbra összehasonlításként a 2000 dimenziós TfIdf logisztikus regresszióval kiértékelve.

után! Sőt! Meglepő módon a pozitív bejegyzések precision és recall score-jai a legkisebbek, várakozásunkkal ellentétben! Ez azt jeleni várhatóan, hogy a negatív szentimentű bejegyzésekben könnyebben felismerhető szavak vannak, amik könnyítik az osztályozást. Erre gondolhattunk volna hamarabb is!

Összefoglalás

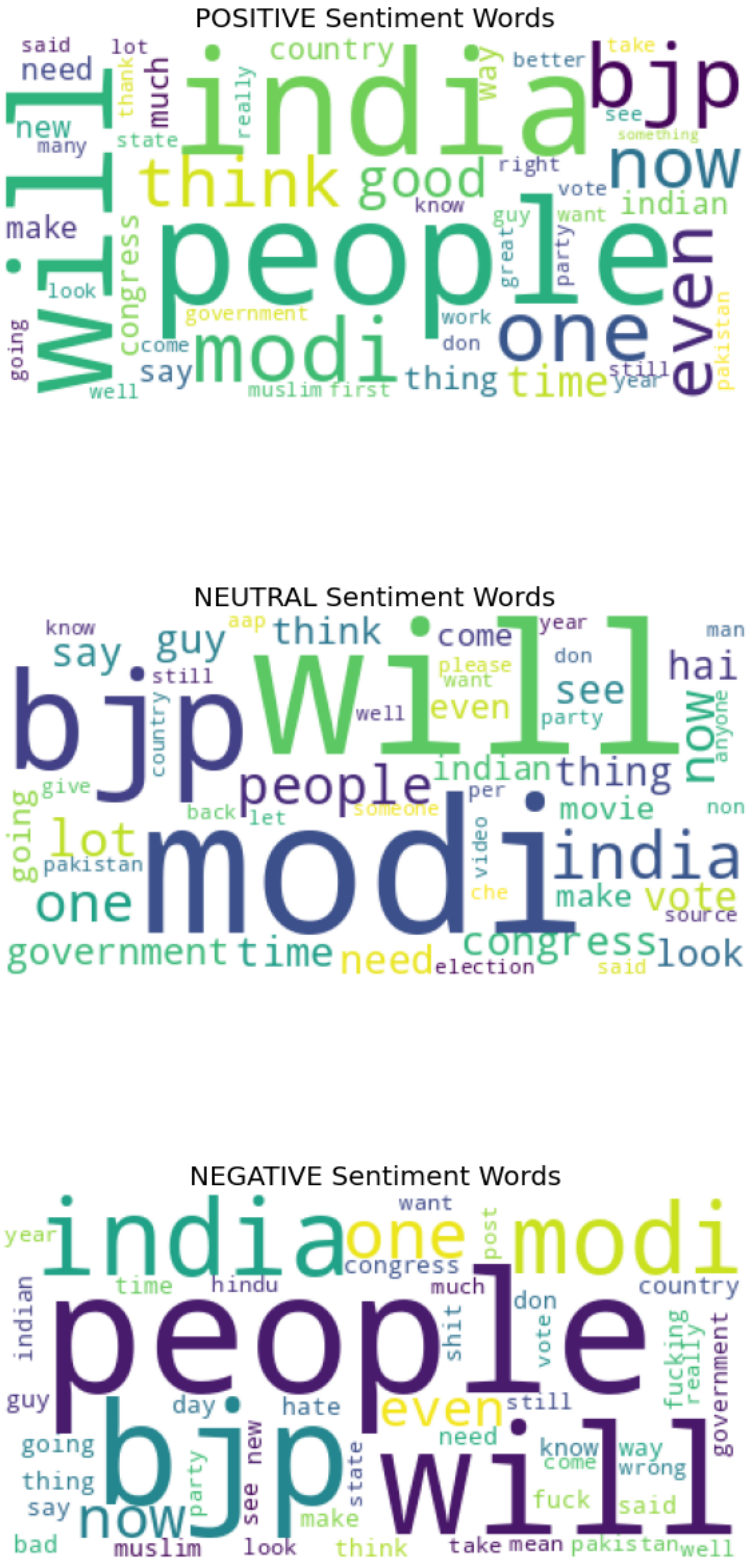
Összefoglalva, hogyan teljesültek a hipotéziseink?

1. A modelleket lefuttatva a következő sorrend áll fenn jóságukra:

$$\texttt{GloVe} < \texttt{word2vec}, \texttt{doc2vec} < \texttt{TfIdf} < \texttt{Bert}.$$

2. A binary modell valóban szignifikánsan jobb, mint a multiclass.
3. Nem volt észlelhető, hogy a pozitív szentiment felé tolódott el volna bármelyik modell.
4. A beágyazás dimenzójáról:
   - Tfidf: valóban nagy dimenziós, optimálisan 2000 dimenziós beágyazásra lett szükség
   - word2vec, doc2vec, GloVe: kifejezetten kis, 50-100 dimenzióban teljesítettek a legjobban.
5. Túltanulás félelme: mindegyik modellben meglepően jól volt kontrollálható a túltanulás. Sőt, a TfIdf és word2vec modellek train-test eredményei fej-fej mellett haladtak.

Táblázatok, ábrák



10. ábra. Szófelhő a 3 osztályról. A leggyakoribb szavakban nagy az átfedés.

	embedding	part	metric	score
0	tfidf	multiclass	acc	0.828443
1	tfidf	multiclass	f1	0.808799
2	tfidf	multiclass	auc	0.928020
3	tfidf	binary	acc	0.835844
4	tfidf	binary	f1	0.882453
5	tfidf	binary	auc	0.788739
6	word2vec	multiclass	acc	0.578914
7	word2vec	multiclass	f1	0.501616
8	word2vec	multiclass	auc	0.725086
9	word2vec	binary	acc	0.696374
10	word2vec	binary	f1	0.802683
11	word2vec	binary	auc	0.585598
12	doc2vec	multiclass	acc	0.482010
13	doc2vec	multiclass	f1	0.400959
14	doc2vec	multiclass	auc	0.654760
15	doc2vec	binary	acc	0.663319
16	doc2vec	binary	f1	0.790815
17	doc2vec	binary	auc	0.524757

11. ábra. Összefoglaló a legjobb mérések eredményeiről 3 beágyazásra.

```
dict_items([('win', 95), ('bjp', 7), ('congress', 11), ('come', 10), ('elect', 16), ('vote', 91), ('take', 82), ('gi
ve', 24), ('start', 78), ('talk', 83), ('muslim', 51), ('well', 94), ('also', 1), ('actual', 0), ('state', 79), ('kn
ow', 36), ('even', 17), ('indian', 34), ('like', 39), ('shit', 74), ('post', 64), ('first', 20), ('say', 71), ('mani
', 45), ('modi', 48), ('thing', 86), ('way', 93), ('india', 33), ('parti', 59), ('read', 67), ('think', 87), ('let',
38), ('lot', 42), ('happen', 30), ('becom', 4), ('peopl', 60), ('use', 90), ('someth', 77), ('right', 69), ('live',
40), ('get', 23), ('point', 62), ('pakistan', 58), ('differ', 15), ('see', 72), ('one', 57), ('new', 55), ('medium',
47), ('chang', 9), ('countri', 13), ('news', 56), ('year', 99), ('issu', 35), ('good', 26), ('feel', 19), ('need', 5
3), ('free', 21), ('would', 98), ('go', 25), ('better', 6), ('ask', 2), ('make', 43), ('hope', 32), ('govern', 28),
('seem', 73), ('fuck', 22), ('money', 49), ('polit', 63), ('leader', 37), ('work', 96), ('everi', 18), ('time', 88),
('hindu', 31), ('world', 97), ('much', 50), ('realli', 68), ('someon', 76), ('said', 70), ('support', 81), ('look',
41), ('guy', 29), ('person', 61), ('back', 3), ('mean', 46), ('call', 8), ('question', 66), ('power', 65), ('still',
80), ('day', 14), ('man', 44), ('got', 27), ('want', 92), ('never', 54), ('show', 75), ('could', 12), ('thank', 85),
('team', 84), ('nation', 52), ('best', 5), ('tri', 89)])
```

12. ábra. A 100 legfontosabb szó a TfIdf módszer alapján.