Social media elemzés természetes nyelvfeldolgozással

Papp Viktor
GN6XOO
weetjoue@gmail.com

Foltányi Bence
CHHHYY
foltanyibence@gmail.com

Gábor Molnár
YUMXHI
mgabor922@gmail.com

A házi feladatunk fő célja egy olyan modell megalkotása volt, amely képes megállapítani egy hosszabb blogbejegyzés írójának a nemét. A feladat során kitértünk a szentiment elemzés területére is, mivel érzelmek, attitűdök megállapítása szöveg alapján némileg egyszerűbb feladat, így ezzel a kezdetleges modellt lehetett felépíteni és tesztelni. Ezek után ezt a modellt fejlesztettük és hangoltuk tovább nemek osztályzására.

I. Bevezető

A mai fogyasztói társadalomban, a cégeknek és marketingeseinek fontos a vásárlók visszajelzése egy adott termékről. A politikusok és politológusok számára fontos az emberek viszonyulása bizonyos közéleti témákhoz. Ezeknek számítógép alapú valós idejű mérése olcsón biztosítaná számukra a kívánt információkat, emiatt egy igen felkapott területnek számít a természetes nyelvfeldolgozás területe.

A természetes nyelvfeldolgozás többféle bonyolultságú területekre osztható. Nehezebb probléma például két nyelv közötti fordítás. Egyszerűbb feladat azonban a szentiment analízis, amivel megállapítható, hogy egy bizonyos megnyilvánulás pozitív, negatív, vagy semleges hangvételű. Mi a kettő közt álló problémát választottunk, mégpedig azt, hogy szöveg alapján meghatározzuk, hogy a szöveg írója nő, vagy férfi. Mivel ennek eredménye kérdéses, először szentiment analízist végeztünk a szövegeken, hogy lássuk hogyan viselkednek a választott modellek, valamint hogyan szükséges az adatot előfeldolgozni.

II. Tématerület ismertetése

A terület nehézsége, hogy az emberi nyelv nem egzakt, többféleképpen kifejezhető ugyanaz a jelentés. Ugyanannak a mondatnak más értelme van ha máshogyan hangsúlyozzuk, vagy ha máshova helyezzük a vesszőt. Egy mondat eleji tagadószó az egész mondat jelentését megnegálhatja. Az irónia szintén csak a kontextus ismeretében értelmezhető. Ezek miatt a hagyományos algoritmusok számára túl nagy kihívást jelent. A számítási teljesítmény rohamos fejlődésével, és a neurális hálózatok folyamatos fejlődése által lehetőség nyílik egyre jobb eredmények elérésére.

Szentiment analízisre léteznek bevált módszerek, akár mélytanulás nélkül is megoldható [9]. A nemek osztályozása nyilvánvalóan bonyolultabb feladat, hiszen egy ember számára egyértelmű, hogy egy bejegyzés pozitív, vagy negatív hangvételű, a bejegyzés írójának neme, ezzel szemben valószínűleg csak tippelhető. Kérdéses, hogy a neurális hálózat mégis talál-e olyan összefüggést a szövegben, ami alapján ezt be tudja azonosítani, az kérdéses. Az erre vonatkozó kutatások 60-85% körüli pontossággal képesek voltak ilyen osztályozást végezni [5][6].

III. Szentiment elemzés

Az ehhez a fejezethez tartozó kód a sentiment.ipynb állományban található meg, és itt ennek a részfeladatnak a bemutatása a cél.

Bár az elsődleges feladatunk a nemek meghatározása volt, hasznosnak tűnt először egy egyszerűbb példán keresztül elkészíteni egy szövegeket feldolgozó és azokat kategorizáló modellt. Olyan feladatra volt szükség, ami könnyebben elvégezhető, de mégis hasonló problémát old meg, ezért esett a választás a szentiment analízisre. Az ötletet alapjául a [1] irodalom szolgált, mely nem csak a szentiment analízist mutatja be, de különböző szövegfeldolgozás szempontjából is fontos módszerekre is kitér.

Ennél a résznél kétfajta modell lett felhasználva. Ezek a következők:

- LSTM alapú
- konvolúciós alapú

Mindkettőnél ki lett próbálva a GloVe embedding rétege is, melyről a [2] irodalom ad átfogó leírást. Ez végeredményben 4 különböző modell kipróbálását és beállítását jelentette.

A. *Az LSTM modell:*

Ehhez alapul a [3] irodalom szolgált alapként, ez adta az ötletet, hogy szövegelemzésre használjunk LSTM réteget, mellyel a szöveg szavainak egymásutániságát lehet modellezni.

A teljes modell a következő rétegekből épült fel:

- Embedding
- SpatialDropout1D
- LSTM
- Dense

B. *A konvolúciós modell:*

A konvolúciós modell kipróbálására a motiváció az volt, hogy bár kisebb szövegeknél még elegendően gyorsan tanítható az LSTM réteg, nagyobb blogbejegyzések jelentősen több szóból állnak. Ilyenkor hasznos lehet a tömörítés, amire az egydimenziós konvolúció kiválóan alkalmazható. Ez a modell az alábbi felépítést használja:

- Embedding
- Convolution1D x4
- Flatten
- Dropout
- Dense
- Dropout
- Dense

C. *Adatok beszerzése*

A modellek két különböző adatbázis segítségével lettek kipróbálva. Az egyik pozitív, negatív és semleges címkéket tárol egy légitársaság említéseihez Twitteren. A másik adatbázis szintén tweeteket tartalmaz, de itt a címkézés 13 alapvető érzelmet tart nyilván.

Az utóbbi adatbázissal nem sikerült jó eredményt elérni, amit az osztályok sokasága és azoknak eloszlása is okozott. Bizonyos osztályokhoz ugyanis csak pár száz, míg a többihez akár több ezer bejegyzés is tartozott.

Az első adatbázis segítségével viszont mindkét modellel 80-70 százalék közötti pontosságot már rövidebb tanítási ciklusokkal is sikerült elérni validációs és teszt adatokon, így a végleges kódban ez maradt benne, és a modell további hangolásához a sentiment analízis során ez lett felhasználva.

D. *Adatok feldolgozása*

Az adatok feldolgozásához meg kellett tisztítani a szövegeket, hogy a különböző nyilvántartott szavak száma ne legyen túl magas, illetve a nagyon ritka szavak és karakterek ne gátolják a tanítást. Ehhez a legtöbb kötőszót, különleges karaktereket és a Twitter által használt referenciákat mind el kellet távolítani.

A következő lépés az így keletkezett szövegek átkódolása, olyan formára, hogy az embedding rétegbe ezeket be lehessen adni. Ehhez one-hot módszert alkalmaztunk, mely a szavakhoz egy szövegben egy egész számot rendel egy adott tartományon belül. Ez a tartomány jelen esetben a szövegekben található különböző szavak száma megnövelve egyel, mivel így minden szó leképezhetővé válik. Ezen kívül a kimenetként használt címkéket is one-hot kódolni kell, mivel ezek az adatbázisban szövegesen szerepeltek. Az alábbi táblázat a kódolás módját ábrázolja.

Címkék kódolása	
Érzelem	***Kód***
Negatív	1,0,0
Semleges	0,1,0
Pozitív	0,0,1

Legvégül fontos kiemelni, hogy az embedding réteg azért különösen fontos, mert ez úgy ágyazza be a szavakat a modellben, hogy egy adott dimenziójú vektorhoz rendeli az összes létező szót az adatbázisban, ami egyben tömörítést is végez. A szavak vektorizálásához ötletet a [4] irodalomban bemutatott módszer adta, mely egyben segített megérteni is a vektoros módszerek jelentőségét.

E. *GloVe felhasználás*

A feladat során mindkét modellhez ki lett próbálva saját embedding réteg helyett az előtanított GloVe réteg felhasználása. Ehhez néhány extra lépést kellet elvégezni, amelyek a következők voltak:

1. A szó – vektor összerendelések betöltése.
2. Ezeknek átformálása feldolgozáshoz.
3. Csak a saját adatbázisban előforduló szavak kiválasztása és ezekhez tartozó vektorok kimentése.
4. Ezekből a súlymátrix létrehozása.
5. Végül e mátrix felhasználásával egy új embedding réteg létrehozása.

F. *Tanítás*

A tanítás során early stopping, checkpoint és history callback objektumokat alkalmaztunk és a modellek ki lettek próbálva SGD és Adam optimalizációval is.

Problémát okozott, hogy Adam optimalizáció esetén a modell már akár a második epoch után elkezdett túltanulni, erre abból következtettünk, hogy bár a tanító adatokon a pontosság tovább növekedett, a validációs adatoknál ez az érték csak csökkent. Ennek meggátlására dropout rétegeket használtunk fel, illetve SGD optimalizációt is kipróbáltuk.

Az SGD viszonylag több epoch után nagyjából ugyanazt az eredményt érte el (sok esetben rosszabbat pár százalékkal), mint Adam esetében.

G. *Eredmények*

Megfigyelhető volt, hogy a GloVe felhasználása befagyaszott rétegként rosszabb eredményt adott, mint a tanított embedding réteg, viszont a tanítást felgyorsította.

Ezzel ellenben nagyjából 1-2 százalékos javulást mutatott a továbbtanított GloVe, mint az azt egyáltalán nem használó esetben (validációs adatokon). Ezen fenti

megfigyelés miatt a GloVe rétegeket nem befagyasztva használtuk a továbbiakban.

Mindkét modell hangolásával nagyjából hasonló eredményeket sikerült elérni, a következő táblázat ezek eredményeit foglalja össze.

A modellek eredményei		
Modell név	***Teszt Loss:***	***Teszt pontosság:***
LSTM	0.54	0.78
LSTM (GloVe)	0.57	0.77
CNN	0.59	0.77
CNN (GloVe)	0.62	0.76

Végül a kód legutolsó része egy olyan blokkot tartalmaz, ahol a felhasználó a saját szövegeit töltheti be, és elemezheti a modell segítségével. Itt néhány frissen begyűjött tweetre alkalmaztuk is a legjobb modellünket, és kiírattuk annak eredményeit, illetve, a három osztály közötti valószínűségek szétoszlását.

IV. Nemek meghatározása

A szentiment elemzésnél kikísérletezett módszerek (embedding, háló-architektúrák, stb.) segítségével folytattuk a feladatot a nemek meghatározásához. A megoldáshoz konvolúciós hálózattal próbálkoztunk.

A. Adatbázis keresés

A nemek meghatározásához először szükségünk volt egy olyan adatbázisra, mely tartalmazza a szöveg írójának nemét. Alapvetően social media, vagy valamilyen rövidebb bejegyzéseket tartalmazó adatbázist kerestünk. Végül találtunk olyan blogbejegyzéseket tartalmazó adatbázist, ami megfelelt elvárásainknak, és rendelkezett a megfelelő nemeket tartalmazó címkékkel [7].

B. Adatok előfeldolgozása

A feladat megoldásához a szentiment analízis esetén használt módszereket használtuk. Először csökkentettük az adatbázis méretét, ugyanis a 170000 blogbejegyzés előfeldolgozása túl sok időt vett igénybe. Emiatt 10000 blogbejegyzést használtunk fel, amit felosztottunk tanító, validációs és teszt adatbázisra 0.7-0.2-0.1 arányban.

A szövegen további előfeldolgozást végeztünk, a nagybetűket kisbetűkre cseréltük, valamint a NLTK tokenizálójával szétbontottuk a szavakat listába. Ez szűrést is végez a szavakon, így kiszedi a felesleges írásjeleket.

Megszámoltuk a kiadódó adatbázis szókincsét, vagyis, hogy ez egyes szavak hányszor fordulnak elő. Hogy gyorsítsunk a futáson, illetve a jelentéktelen szavakat eldobjuk, kiszűrtük a legritkábban előforduló szavakat, így csökkentve a szókincs méretét.

Később további technikákat is kipróbáltunk ebben a fázisban. Ilyen volt az úgynevezett stopword-ök kiszedése a szövegből. Ez kiszedi az angol nyelv leggyakrabban használt szavait, például a (have, the, stb.) szavakat. Ez azért lehet jó, mert ezek a kötőszavak nem rendelkeznek valódi jelentéssel. Ezt a funkciót kódból kapcsolhatóvá tettük, ugyanis egyes források szerint segít a szöveg értelmének kinyerésén, egyes források szerint viszont nem [8].

Másik ilyen technika volt a stemming. Ennek segítségével a szavak végéről a toldalékok levághatóak, így minden szónak csak a szótövét tartja meg, ami a jelentés alapját hordozza. Ezzel a szókincset is tovább tudtuk csökkenteni, hiszen az ugyanolyan jelentéssel, azonban stilisztikailag máshogyan megjelenő szavakat egy kategóriába lehetett kódolni.

Miután ez készen volt, a már ismertetett módon one-hot kódolással kódoltuk a szavakat. Ezután azt tapasztaltuk, hogy az egyes blogbejegyzések hossza jelentősen eltér, emiatt 500 szóban maximalizáltuk az egyes blogbejegyzések méretét. Amelyik blogbejegyzés nem tartalmazott 500 szót, ott nullákkal töltöttük fel az üres helyeket.

Ezután az adatok készen álltak arra, hogy a már ismertetett embedding rétegnek megadjuk az adatokat. Mivel a GloVe használata nem javított a szentiment analízis eredményén, itt nem használtunk előre tanított embedding réteget.

C. Konvolúciós hálózat

A hálózat építéséhez a keras nyílt forráskódú python könyvtárat használtuk fel. A keras több különböző backend felett is képes futni, ezek közül mi a TensorFlow backend-et használtuk. A one-hot kódolt adatainkat elsőként egy keras embedding layer-nek adtuk be. A NLP fontos részét képezi, hogy a szavak jelentését, relatív jelentésüket valamilyen formában reprezentáljuk. A keras embedding layer ezt a már korábban is említett módon teszi, vagyis, hogy a szavakhoz vektorokat rendel. Ennek a megoldásnak következménye, hogy a jelentésben közel álló szavakat reprezentáló vektorok közötti távolság kicsi lesz.

A nemek meghatározásához a szentiment elemzésből levont következtetések alapján úgy döntöttünk, hogy a keras embedding layert követően egy konvolúciós hálózatot fogunk használni. A konvolúciós háló felépítését az alábbi táblázat mutatja.

Konvolúciós háló	
Réteg	***Neuronok száma***
Conv 1D	256
Conv 1D	128
Conv 1D	64
Conv 1D	32
Conv 1D	16
Dense	180
Dense	1

A konvolúciós rétegek és a dense rétegek között egy flatten réteget is elhelyeztünk annak érdekében, hogy a

két réteg közötti dimenziók megfeleljenek. Az első dense réteg esetében Relu aktivizációs függvényt használtunk, a másodiknál pedig szigmoid függvényt. Mindkét dense rétegnél a dropout értékét 0.5-re állítottuk. Optimalizációhoz az SGD (Stochastic Gradient Descent) módszer nesterov momentum variációját használtuk. A tanításhoz early stoppingot használtunk, valamint a súlyok elmentéséhez a keras checkpoint callback metódusát.

Az első tanítást tízezer adaton végeztük el. Az adatok 20%-át használtuk validációra és 10%-át tesztelésre. A tanítást a collab segítségével végeztük el, a google által rendelkezésre bocsájtott GPU-kon. A teszt adatokon 55% körüli eredményt sikerült elérni, amiből azt a következtetést vontuk le, hogy a háló nem tanul. Emiatt az adatfeldolgozást újra elvégeztük, immr ötvenezer adaton. A tanító, validációs és teszt adatok változatlanul hagyása mellett 67 epoch alatt a hálózatunk betanult és a teszt adatokon 64%-os pontosságot sikerült elérni.

V. Összefoglalás

A projekt során megismerkedtünk természetes nyelvfeldolgozás alapjaival és megtapasztaltuk a terület nehézségeit, problémáit. A kihívások már az adatfleldolgozáskor jelentkeztek. A blogbejegyzések, szövegek komoly előkészítésre szorulnak ahhoz, hogy hatékonyan tudjuk tanítani a hálózatunkat. Egyszerűbb, de nélkülözhetetlen lépések például az írásjelek kiszedése, valamint az egész szöveg kisbetűssé alakítása. Nehezebb feladat volt a kötőszavak (stopword-ök) kiszedése, valamint a stemming, vagyis a szó toldalékok, ragok eliminálása. Az előfeldolgozást követően a one-hot kódolt szavak vektorrá alakítására volt szükség, ehhez a keras word embedding layer-ét használtuk fel, melyet az előtanított GloVe réteggel is teszteltünk.

A NLP területén használatos háló architektúrákkal először egy szentiment analízises feladat során ismerkedtünk meg. Konvolúciós és LSTM felépítésű hálókat is kipróbáltunk a feladat megoldásához. Az adam és az SGD optimalizációs módszereket is kipróbáltuk, és összehasonlítottuk őket. A fent említett módszerekkel a teszt adatokon 77-78%-os pontosságot sikerült elérnünk. Az itt megszerzett tapasztalatokat használtuk fel a nemek meghatározásához. Az útóbbi egy nehezebb feladatnak bizonyult, ugyanis itt csak 63%-os pontosságot sikerült elérni a teszt adatokon. Ezen az adatok további előfeldolgozásával, valamint összetettebb háló architektúrával lehetne javítani.

[1] Lei Zhang, Shuai Wang, Bing Liu: *Deep Learning for Sentiment Analysis: A Survey*

[2] Jeffrey Pennington, Richard Socher, Christopher D. Manning: *GloVe: Global Vectors for Word Representation*

[3] Martin Sundermeyer, Ralf Schluter, and Hermann Ney: *LSTM Neural Networks for Language Modeling*

[4] Yoav Goldberg, Omer Levy: *word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method*

[5] Aric Bartle, Jim Zheng: Gender Classification with Deep Learning

[6] Liang Wang, Qi Li, Xuan Chen, Suijan Li: Multi-task Learning for Gender and Age Prediction on Chinese Microblog

[7] https://www.kaggle.com/tomlisankie/blog-posts-labeled-with-age-and-gender/

[8] Hassan Saif, Miriam Fernandez, Yulan He, Harith Alani: On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter

[9] Walaa Medhat, Ahmed Hassan, Hoda Korashy: Sentiment Analysis algorithms and applications: A survey