UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Antrenarea unui model pentru detectarea efectelor adverse ale medicamentelor in mediul online

propusă de

Dragoș-Constantin Ciocan

Sesiunea: iulie, 2020

Coordonator științific

Conf. Dr. Mădălina Răschip

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Antrenarea unui model pentru detectarea efectelor adverse ale medicamentelor in mediul online

propusă de

Dragoș-Constantin Ciocan

Sesiunea: iulie, 2020

Coordonator științific

Conf. Dr. Mădălina Răschip

Cuprins

Motivație

În anul 2017, în cadrul Workshop-ului AMIA-2017 despre Exploatarea Rețelelor de Socializare pentru Aplicații în Sănătate, a fost propusă, printre altele, următoarea temă: depistarea efectelor adverse ale medicamentelor lansate pe piață, din postările consumatorilor acestora pe rețele de socializare. M-am decis ca aceasta este și tema mea de licență și voi încerca să antrenez un model care să îndeplinească această sarcină. Motivul pentru care

Introducere

Problema reacțiilor adverse ale medicamentelor este una de mare notorietate în lumea medicinală. Cei care au cel mai mult de suferit și de pierdut de pe urma acestei probleme sunt consumatorii acestora, aceștia putându-și pune viața în pericol în anumite cazuri (prescripții greșite, nerespectarea prescripției), însă afectate sunt și companiile, producătorii și toți cei care lucrează în sfera farmaceuticelor.

Reacțiile adverse ale medicamentelor înseamna orice simptome neprevăzute, malicioase pe care pacientul le simte după un anumit interval de timp de la ingerarea medicamentului. Pentru a se preveni această problemă, se efectuează seturi de teste, pe pacienți care se înscriu voluntar la această acțiune, însă, există cazuri în care aceste teste nu sunt suficiente, iar la momentul lansării medicamentului, oamenii să experimenteze reacții adverse. Acest fapt este bine cunoscut de toată lumea, de aceea se colectează mereu date despre efectele medicamentelor și după lansare. Colectarea se dovedește a fi destul de minuțioasă, deoarece nu toți pacienții vorbesc cu doctorii lor despre aceste lucruri, dar și din alte motive. Ideea este ca nu există o metodă standardizată pentru colectarea datelor referitoare la efectele acestor medicamente.

S-a observat că unii pacienți preferă să-și exprime nemulțumirea, alături de efectele adverse ale medicamentelor, în mediul online. Dar, desigur că și de aici este anevoios de extras anumite informații, din cauza unor exprimări precare, utilizarea argourilor, etc. La Workshop-ul AMIA-2017, prin intermediul unui concurs, s-a încercat rezolvarea acestei ramuri a problemei. La concurs au putut participa echipe din toată lumea, iar în total au fost desemnate trei probleme. Eu în această lucrare mă voi axa doar pe una, cea descrisă mai sus.

Pentru rezolvarea acestei probleme, am încercat antrenarea unui model care să aibă în final un scor cât mai mare la testare, inspirându-mă din lucrarea de aici. Am încercat prin două metode de transformare a textului într-un format ușor de înțeles de către calculator (transformând în vectori de numere), acestea fiind n-grame-ele, care

s-au dovedit a nu da un scor foarte bun, și transformarea word2vec, combinată cu un algoritm SVM din biblioteca sci-kit learn, care a dat și rezultatele cele mai bune.

Contribuții

În antrenarea modelului dorit a trebuit să folosesc mai multe tehnologii deja existente, pe care le voi enumera în acest capitol.

Pentru început, ca limbaj de programare am folosit Python 3.8, deoarece există foarte multe librării din domeniul de învățare automată. Pe deasupra, acestea, ca toate librăriile din Python, se instalează foarte ușor folosind-une de modului pip. Așadar, am foloist următoarele biblioteci:

- scikit-learn
- gensim
- fasttext
- tensorflow
- keras

, pe lângă librariile standard.

Pentru modelarea limbajului am folosit următoarele tehnici:

- n-grame
- Word2Vec, din cadrul librăriei gensim
- fasttext, din cadrul librăriei fasstext

În final, ca și clasificatori, am folosit următorii algoritmi:

- SVM
- regresie logistică
- bayes naiv
- rețele neuronale

În următoarele capitole, voi prezenta aceste tehnici în amănunt.

Capitolul 1

Setul de date

Setul de date pentru această sarcină a fost preluat din cadrul unui proiect de detectare a efectelor adverse ale medicamentelor de către DIEGO Lab, din cadrul universtății "Arizona State University". Acestea au fost preluate făcându-se căutări pe baza numelor de medicamente, după care niște experți în domeniul farmaceutic le-au clasificat în cele două categorii (conțin sau nu conțin reacții adverse ale medicamentelor).

Tweet-urile au fost furnizate pe bază de ID-uri, ele trebuind descărcate utilizând un script în python 2.*. Lucrând în python 3.* a trebuit să-mi creez eu acest script (pe baza celui dat). Descărcarea acestora a durat aproximativ o oră, iar din această cauză le-am salvat local, nemaifiind nevoie să-l rulez din nou. Din cele 15667 de ID-uri primite, doar 9257 au rămas valabile. Din acestea 9257, am folosit 70% din acestea pentru setul de antrenament, iar restul de 30% pentrul setul de test.

O mare problemă cu acest set de date este faptul că există o mare diferență dintre numarul de date de clasă 0 si numărul de date de clasă 1 (datele sunt nebalansate).

Capitolul 2

Preprocesarea limbajului

Primul pas pe care l-am folosit pentru antrenarea modelului a fost preprocesarea textului. Deoarece textele noastre, ca date de antrenare și de testare, sunt postări de pe Twitter, ele vor avea un limbaj informal, cuvinte prescurtate, argouri, ceea ce face antrenarea modelului nostru dificilă. Ca urmare a acestui fapt, am preprocesat textul pentru a-l aduce într-o formă relativ mai ușoară de înțeles pentru calculator.

Primul pas, alături de cel de-al doilea, au fost inspirați din lucrarea aceasta, aceștia constând în înlocuirea unor termeni specifici (medicamentele și reacțiile adverse) cu un simbol, deoarece acestea nu sunt relevante în clasificarea tweet-urilor. Am folosit o listă de medicamente și de reacții adverse de pe situl celor de la Diego Lab. Am înlocuit medicamentele cu simbolul ”MED”, iar reacțiile adverse cu ”ADR”.

Următorul pas, cel de-al treilea, a fost înlocuirea cuvintelor prescurtate, de tipul ”i’m”, ”you’re”, în cuvinte întregi, pentru a restrânge numărul cuvintelor totale, dar și pentru a le da un sens mai puternic acestora.

Al patrulea pas, unul care nu este important de unul singur, dar prinde valoare datorită următorilor pași, este cel de înlocuirea url-urilor cu un simbol specific. Url-urile având caractere speciale (”:”, ”/”) care trebuie și ele eliminate din text, se vor crea cuvinte nefolositoare, de exemplu ”http”. Simbolul folosit pentru acestea este ”LINK”.

Al cincilea pas constă în înlocuirea referințelor către alte persoane, de forma ”@cineva”, într-un simbol specific (”REF”), deoarece acestea conțin diferite nume, care nu au nicio importanță.

Ultimul pas cuprinde eliminarea din text a majorității caracterelor non-alfanumerice pentru a mai aerisi textul și a despărți unele cuvinte de acestea, existând situații când aceste caractere, fiind lipite de cuvinte, construiesc altele noi din punctul de vedere al

calculatorului.

Capitolul 3

Balansarea datelor

Cum am spus într-un capitol anterior, datele din setul de date dat are datele extrem de nebalansate. Mai precis, 10% din date sunt de clasă 1, iar restul de 90% sunt de clasă 0. Inițial am încercat să antrenez clasificatorul fără a balansa datele, dar rezultatul, evident, a fost foarte slab, clasificatorul prezincând în majoritatea cazurilor (peste 95 la sută) că datele sunt de clasă 0.

Am folosit două tehnici de balansare a datelor: metoda "undersampling" și algoritmul SMOTE (Syntethic Minority Oversample Technique). Metoda undersampling presupune folosirea mai puținor date din clasa majoritară, raportate la clasa minoritară. Am făcut mai multe teste, folosind rapoartele 1:2, 1:3, 1:4 și 1:5, iar tweet-urile nefolosite le-am folosit în setul de test, cele mai bune rezultate fiind date de cel dintâi. Rezultatele obținute, folosind toate cele trei metode de encodare a tweet-urilor în vectori (n-grame, Word2Vec și fasttext), au fost decente, mai bune decât în cazul în care dar nu am folosit nicio tehnică, dar nu a fost destul, așa că am încercat și algoritmul SMOTE.

Algoritmul SMOTE reprezintă o metodă ingenioasă de a genera noi date, folosindu-se de datele actuale. Această generare se face pe baza alegerii de puncte aleatoare în hiperspațiul definit de către caracteristicle tweet-urilor. Mai exact, se alege aleator un punct (adică un tweet), după care se alege aleator dintre cei mai apropiați k vecini un punct. În final, pe dreapta care trece prin aceste două puncte alese, se alege aleator un punct între acestea, acest nou punct desemnând noul tweet generat artificial. Pentru folosirea acestei metode, am ales să nu folosesc nicio librărie și să-mi fac singur implementarea, care în idee pare extraordinar de simplă și elegantă. Într-adevăr, nu am întâmpinat dificultăți, doar faptul că a trebuit să deduc singur formula pentru o găsi

un punct pe o dreaptă într-un hiperspațiu. Formula se bazează pe un exemplu găsit aici, eu doar generalizându-l.

Rezultatele obținute folosind acest algoritm au fost cele mai bune, așa că am decis ca acesta să fie algoritmul final de balansare a datelor.

Capitolul 4

Prelucrarea limbajului

Prelucrarea limbajului natural reprezinta cea mai importanta parte când lucrăm cu acesta. Pentru noi, ca oameni, este ușor spre foarte ușor să înțelegem limbajul , însă, pentru calculator, este aproape imposibil. Limbajul oamenilor conține o multitudine de informații, acesta nu doar bazându-se doar pe cuvintele spuse (sau scrise), ci și pe alți factori cum ar fi tonalitatea vocii, așezarea cuvintelor, etc. Calculatorul nu este capabil să înțeleaga cuvintele, legăturile dintre acestea, în felul în care le știm noi, așa că limbajul trebuie transformat astfel încât să fie înțeles și de către acesta.

Există mai multe tehnici de prelucrare a limbajului natural. Toate acestea au un lucru comun, transformarea acestuia în vectori. Dimensiunea acestor vectori reprezintă caracteristicile unității noastre (cuvânt, propoziție), ca în majoritatea problemelor de învățare automată. Neștiindu-se exact caracteristicile limbajului, dimensiunea acestor vectori poate să varieze, în funcție de problemă, iar numai prin mai multe teste și încercări putem ajungea la dimensiunea optimă.

În continuare voi prezenta reprezentările vectoriale ale cuvintelor folosite în încercarea mea de a antrena modelul dorit.

4.1 N-grame

Reprezentarea propozițiilor în această formă este una destul de simplă, dar neeficientă când se lucrează cu texte mici și cuvinte multe (ca în cazul nostru). Pentru început, am parcurs fiecare tweet, în ordinea în care a fost salvat în setul de date, iar fiecare tweet a fost împarțit în bucăți de dimensiune n (din n-grame). Am salvat aceste n-grame într-un dicționar, pentru a putea contoriza și numărul aparițiilor aces-

tora, după care am ales un număr de cele mai întâlnite n-grame (10.000). După această selecționare, vom transforma fiecare propoziție în vectori de 10.000 de elemente, care vor avea 1 (sau numărul de apariții în tweet-ul respectiv) pe pozițiile din vector corespunzătoare n-gramelor prezente în tweet-ul respectiv, iar în rest 0.

Cum am specificat mai devreme, tweet-urile sunt texte de dimensiune mică, 10 - 20 de cuvinte, asta însemnând că într-un vector de lungime 10.000, 10 - 20 de elemente diferite de 0 nu vor face mai deloc diferența, ceea ce se vede și în rezultate, modelul prezicând în proporție de 99% că tweet-urile sunt de clasă 0.

O altă problemă în această reprezentare în cazul nostru este faptul că este foarte probabil ca în setul de antrenament să existe cuvinte care să nu se regăsească și în setul de test, ceea ce ar face face și mai greu de prezis pentru model cărei clasă aparține tweet-ul respectiv, deoarece acesta ar putea să aibă până la niciun element diferit de 0.

Ultima problemă adusă de utilizarea n-gramelor este că, deoarece vectorii sunt extrem de mari, antrenarea se face foarte lent (și ineficient!). Am ales să abandonez această metodă rapid și am încercat și alte transformări.

Pentru folosirea acestei metode nu am utilizat nicio librărie, folosind o variantă proprie, deoarece este o metodă simplă și ușor de implementat.

4.2 Word2Vec

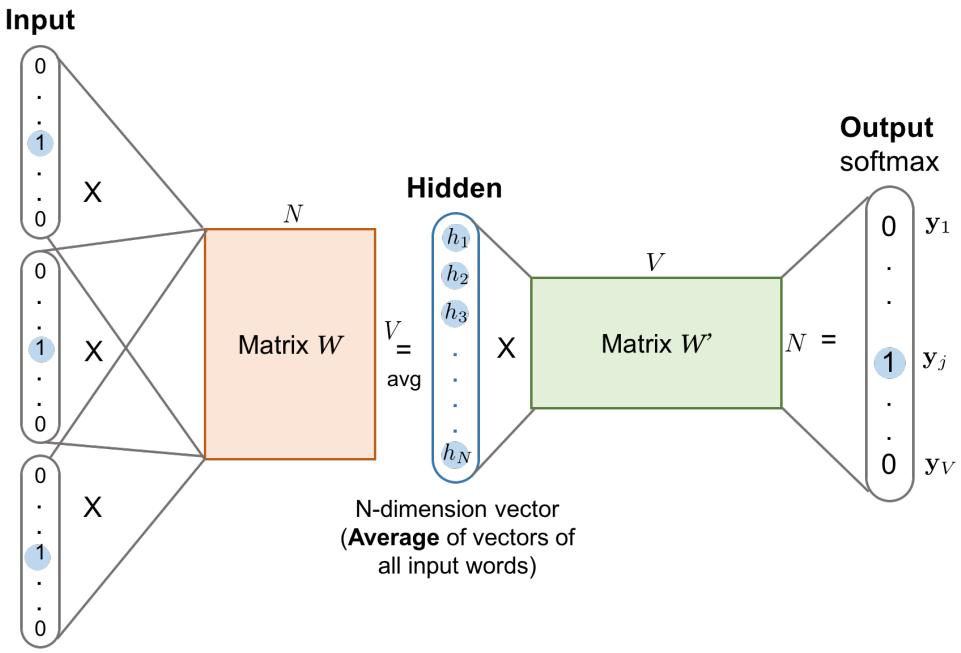
Metoda Word2Vec, introdusă la începutul anilor 2000, schimbă destul de mult modul de abordare al informaticienilor (sau al oamenilor de știință) asupra problemelor de procesare a limbajului natural. Astfel, poate cel mai important aspect este faptul că se micșorează de mai multe ori numărul de elemente din vectori, în schimbul măririi densității (elementele vectorului nu mai reprezintă apariții sau eventual probabilități), care se caracterizează prin elemente de tip ”float”, care pot fi negative sau pozitive, în funcție de diferiți factori. Deci, față de metoda folosită mai sus, este o imbunătățire enormă, cel puțin din punctul de vedere al memoriei folosite.

Sunt două mari metode care folosesc ideea de Word2Vec, CBOW și Skip-Gram. CBOW este metoda prin care se calculează probabilitatea unui cuvânt (vectorul asociat acestuia) pe baza unui context, iar Skip-Gram este exact opusul, adică se pleacă de la un cuvânt și se calculeaza probabilitatea unui anumit context (vectorii cuvintelor din context).

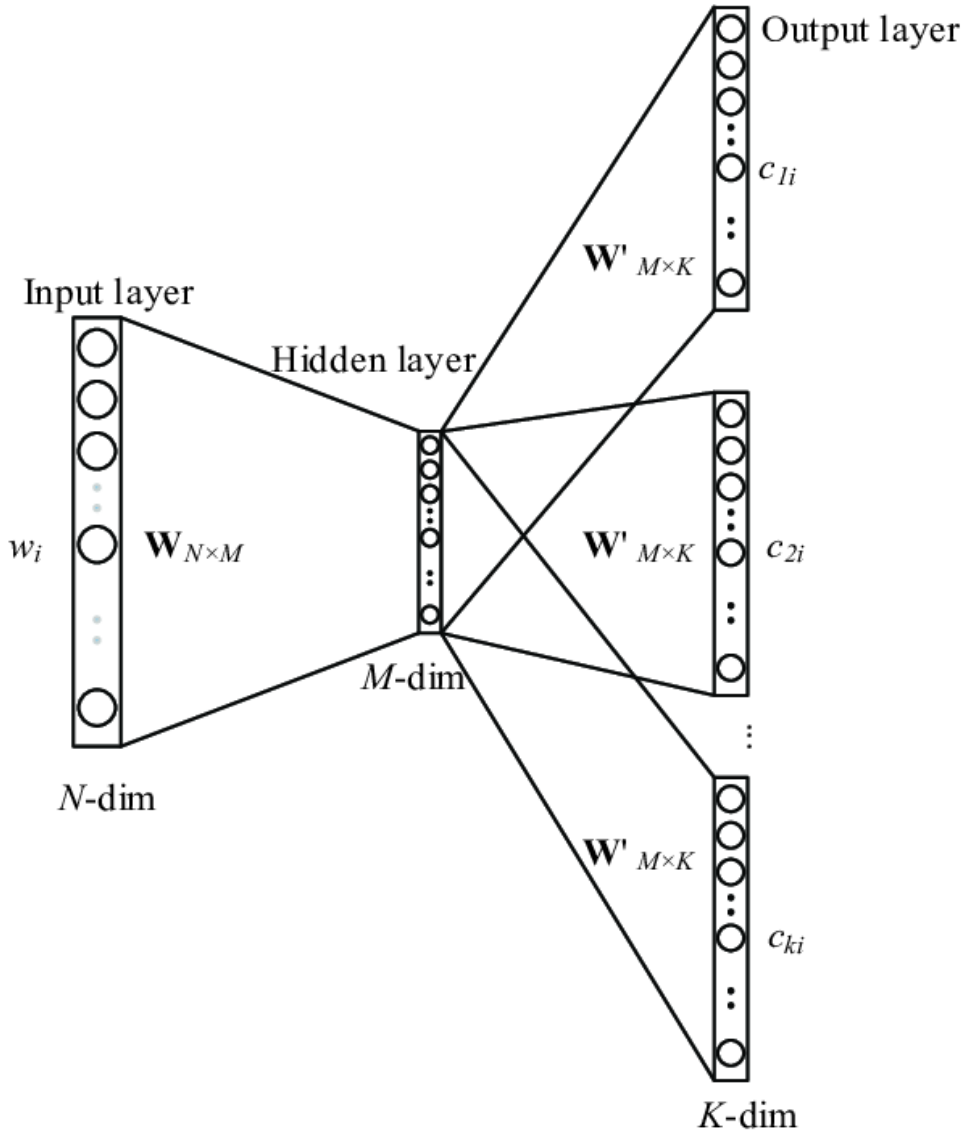
Cum reușește metoda Word2Vec să obțină aceste rezultate (sau encodări)?

Această tehnică se bazează pe o antrenarea nesupervizată a unei rețele neuronale, care are ca intrare un ”one-hot-vector” corespunzător cuvântului care este antrenat, iar ca ieșire vectori de aceeași dimensiune ca vectorul de intrare, unde C reprezintă numărul de cuvinte din context. Deoarece este o antrenare nesupervizată, de interes sunt valorile ”weight-urilor” dintre stratul ascuns și stratul de final.

În următoarea schemă se observă modelul de antrenare al metodei CBOW:



iar în continuare, metoda Skip-Gram:



Inițial când am folosit această tehnică, am antrenat modelul Word2Vec doar cu tweet-urile din setul de antrenament, iar acest lucru s-a dovedit destul de ineficient deoarece, după cum am specificat mai sus, setul de date este format din tweet-uri scurte, cu multe cuvinte diferite, sub diferite forme, fapt care face și modelul Word2Vec să nu acapareze multe cuvinte, sau să nu stabilească așa de bine vectorii cuvintelor găsite. Am avut evident rezultate mai bune decât în cazul n-gramelor, însă f-score-ul clasei 1 a crescut considerabil. Pentru a mări eficiența clasificatorului final, am decis să antrenez modelul Word2Vec folosind mai multe seturi de tweet-uri, ajungând până la aproximativ un milion de tweet-uri. Într-adevăr s-a văzut o diferență considerabilă, f-score-ul final pe setul de test dublându-se.

Pentru folosirea acestei metode, am utilizat librăria gensim din python, deoarece se folosește ușor.

4.3 FastText

Metoda fasttext este o metodă ingenioasă, creată de FAIR (Facebook's AI Research Lab). Această metodă, asemănătoare modelului Word2Vec, prin metoda CBOW sau Skip-Gram, antrenează o rețea neuronală supervizat sau nesupervizat, pentru a descoperi caractersticle cuvintelor. Diferența de Word2Vec este faptul că nu se antrenează vectorul unui cuvânt în funcție de context (sau contextul în funcție de cuvânt), ci se antrenează n-gramele cuvintelor (cuvântul este împărțit în bucăți de lungime n), după care, pentru aflarea vectorului unui cuvânt, se adună element cu element vectorii n-gramelor din care acesta este format.

Cei de la Facebook nu a făcut publice detaliile de implementare ale acestei metode, însă pune la dispoziție gratuit bibliotecile pe care aceștia l-au creat.

Rezultatele obținute au fost puțin mai bune decât cele obținute folosind Word2Vec. Pentru partea de folosire, este evident faptul că am folosit o bibliotecă în python, aceasta fiind fasttext.

4.4 BERT

Metoda BERT (Bidirectional Encoder Representations from Transformers), este o nouă metodă, publicată de către cercetători de la Google AI Language. Această inovație are ca idee de bază antrenarea bidirecțională a Transformatorilor, în care n-o să intru în detaliu deoarece nu i-am folosit și nici nu am avut intenția. De fapt, această metodă "calculează" sensurile cuvintelor în ambele direcții, spre deosebire de Transformatorii simpli, care o făceau ori de la dreapta la stânga, ori invers.

Această metodă a uimit prin faptul că a reușit să obțină "state of the art" (cele mai bune rezultate) în mai multe probleme de modelare a limbajului natural. Din păcate nu am reușit să rulez această metodă, deoarece folosește prea multă memorie RAM, dar am zis că merită menționată atât timp cât am încercat.

Capitolul 5

Clasificatori

Clasificatorii, în învățarea automată, reprezintă algoritmi care, pe baza unor seturi de date de diferite tipuri, pot prezice tipurile unor date noi, pe care algoritmul "nu le-a văzut". În continuare voi prezenta clasficatorii folosiți de mine. Toți algoritmii folosiți sunt din biblioteca scikit-learn, din Python.

5.1 SVC

SVC (Support Vector Classifier) este un clasificator cu o idee de bază simplă și elegantă. Acesta, în combinație cu SVM (Support Vector Machine), clasifică datele printr-un hiperplan într-un hiperspațiu (mai mare decât cel format caracteristicile datelor). De exemplu, există posibilitatea ca două seturi de puncte să nu poată fi despărțite de un singur hiperplan, în hiperspațiul definit de caracteristici, iar pentru a rezolva această problemă, prin intermediul unei funcții nucleu (polinomială sau rbf), se proiectează punctele într-un hiperspațiu superior, acestea putând fi clasificate de către un hiperplan.

Hipeplanul separator definit de acest clasificator se bazează pe găsirea unei margini maxime, dar care să nu ia în considerare punctele de tip "outlier", această margine numindu-se "moale" (soft margin). Pentru a realiza acest lucru, se face o antrenare de tip cross-validation, după care se alege cea mai bună margine.

Aceste funcții nu proiectează propriu-zis punctele într-un hiperplan superior, ci calculează relația acestora în hiperplanul respectiv. Funcția polinomială, descrisă de formula

$$(a \times b + r)^d$$

, are ca parametru d, care reprezintă dimensiunea proiecției, r este o constantă, iar a și b sunt coordonatele punctelor, adică vectorii elementelor pentru care se dorește calcularea relației. Această funcție este limitată la un număr finit de dimensiuni.

Rbf este o funcție mai complexă, aceasta nefiind limitată de numărul de dimensiuni, calculul acesteia reprezentând un hiperspațiu infinit. Este definită de formula

$$e^{-\gamma(a-b)^2}$$

, în care a și b reprezintă același lucru (vectorii pentru care se calculează relația), iar γ este o constantă.

În continuare vom demonstra că funcția rbf se poate scrie ca un produs infinit. Pentru început, alegem $\gamma = \frac{1}{2}$, iar funcția va deveni:

$$e^{-\frac{1}{2}(a-b)^2} = e^{-\frac{1}{2}(a^2+b^2)+\frac{1}{2}2ab} = e^{-\frac{1}{2}(a^2+b^2)} \cdot e^{ab}$$

. Următorul pas este aproximarea funcției e^{ab} cu o serie Taylor, cu a = 0. Fie următoarea serie pentru e^x, ținându-se cont de faptul că derivatele acesteia sunt egale cu funcția exponențială:

$$e^x = e^0 + e^0 \cdot x + \frac{e^0}{2!}x + \frac{e^0}{3!}x^2 + \cdots + \frac{e^0}{n!}x^n + \cdots$$

Înlocuim x cu ab, iar ecuația devine:

$$e^{ab} = 1 + ab + \frac{1}{2!}(ab)^2 + \frac{1}{3!}(ab)^3 + \cdots + \frac{1}{n!}(ab)^n + \cdots$$

, care poate fi scrisă:

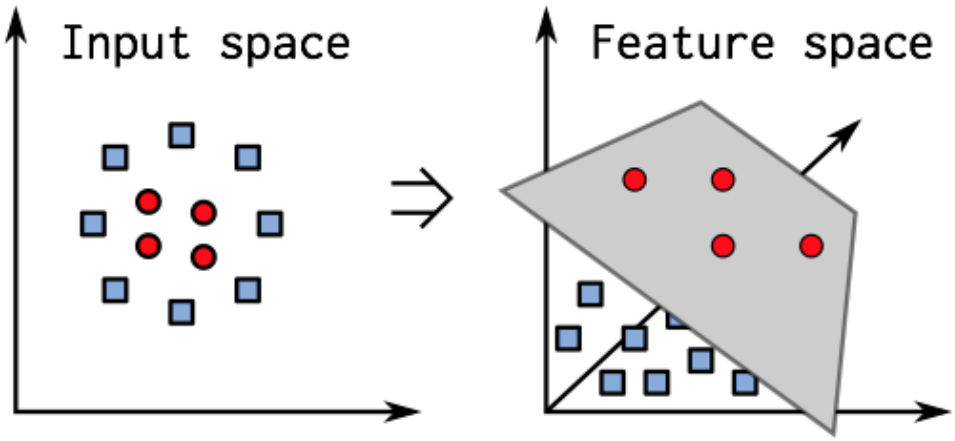
$$e^{ab} = (1, a, \frac{1}{\sqrt{2!}}a^2, \frac{1}{\sqrt{3!}}a^3, \cdots, \frac{1}{\sqrt{n!}}a^n, \cdots) \cdot (1, b, \frac{1}{\sqrt{2!}}b^2, \frac{1}{\sqrt{3!}}b^3, \cdots, \frac{1}{\sqrt{n!}}b^n, \cdots)$$

Folosind notația $e^{-\frac{1}{2}}(a^2 + b^2) = s$, ecuația inițială va arăta astfel:

$$e^{-\frac{1}{2}(a-b)^2} = (s, sa, \frac{1}{\sqrt{2!}}sa^2, \frac{1}{\sqrt{3!}}sa^3, \cdots, \frac{1}{\sqrt{n!}}sa^n, \cdots) \cdot (1, sb, \frac{1}{\sqrt{2!}}sb^2, \frac{1}{\sqrt{3!}}sb^3, \cdots, \frac{1}{\sqrt{n!}}sb^n, \cdots)$$

Drept urmare, am putut scrie funcția rbf ca un produs de doi vectori infiniți.

În următoarea imagine se poate observa cum se face trecerea unor puncte dintr-un spațiu 2-dimensional într-unul 3-dimensional, după care se găsește planul despărțitor.



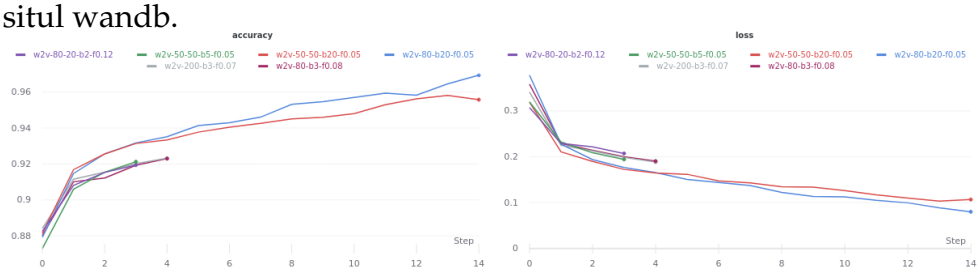
Am încercat ambele funcții nucleu, însă rbf (radial basis function) a avut rezultate mai bune. Diferența între cele două este că funcția rbf proiectează punctele într-un hiperplan infinit, folosindu-se de o aproximare Taylor (serie Taylor), iar cea polinomială le proiectează într-un hiperspațiu de dimensiune n.

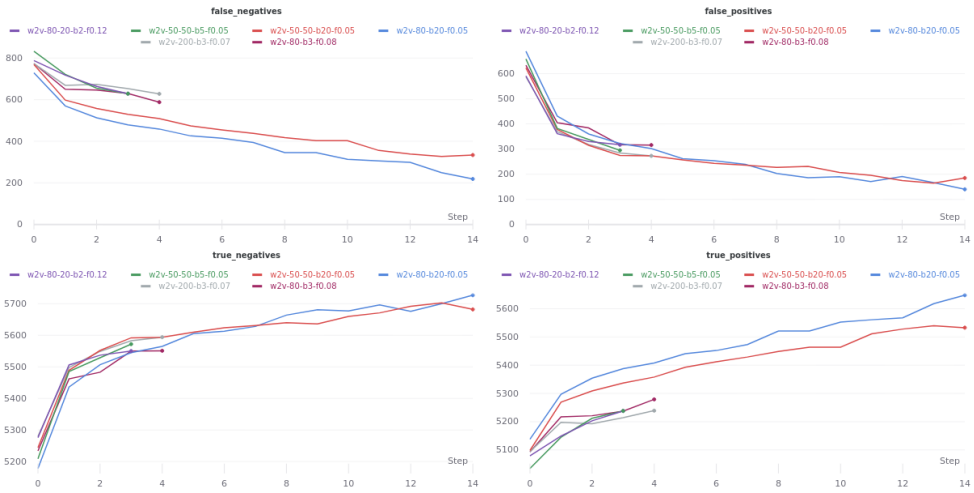
Acesta a fost clasificatorul care a obținut cele mai bune rezultate, însă a fost mai lent decât ceilalți.

5.2 Rețea neuronală

Am încercat antrenarea și cu o rețea neuronală, folosind biblioteca keras, însă nu prea am avut rezultate bune. Am încercat mai multe structuri (1, 2, 3 straturi ascunse), diferite funcții de activare pe ultimul strat, pentru straturile din interior folosind funcția RELU, diferiți opimiziatori (adam, SGD, etc.). Configurația optimă este alcătuită din 2 staturi interioare, de 80 și respectiv 20 de neuroni, optimizatorul adam, funcția de cost crosentropie binară și funcția de activare de pe ultimul strat sigmoid. Această configurație a produs un f-score de 0.12, care este foarte slab. Numărul de epoci l-am ținut cât mai mic, deoarece face overfit foarte rapid, în această problemă fiind nevoie de găsirea unor caracteristici mai generale.

Pentru realizarea unor comparații între mai multe configurații ale rețelei, am folosit situl wandb.





După cum se vede în aceste grafice, rețeaua neuronală are cam aceeași direcție.

5.3 Bayes Naiv

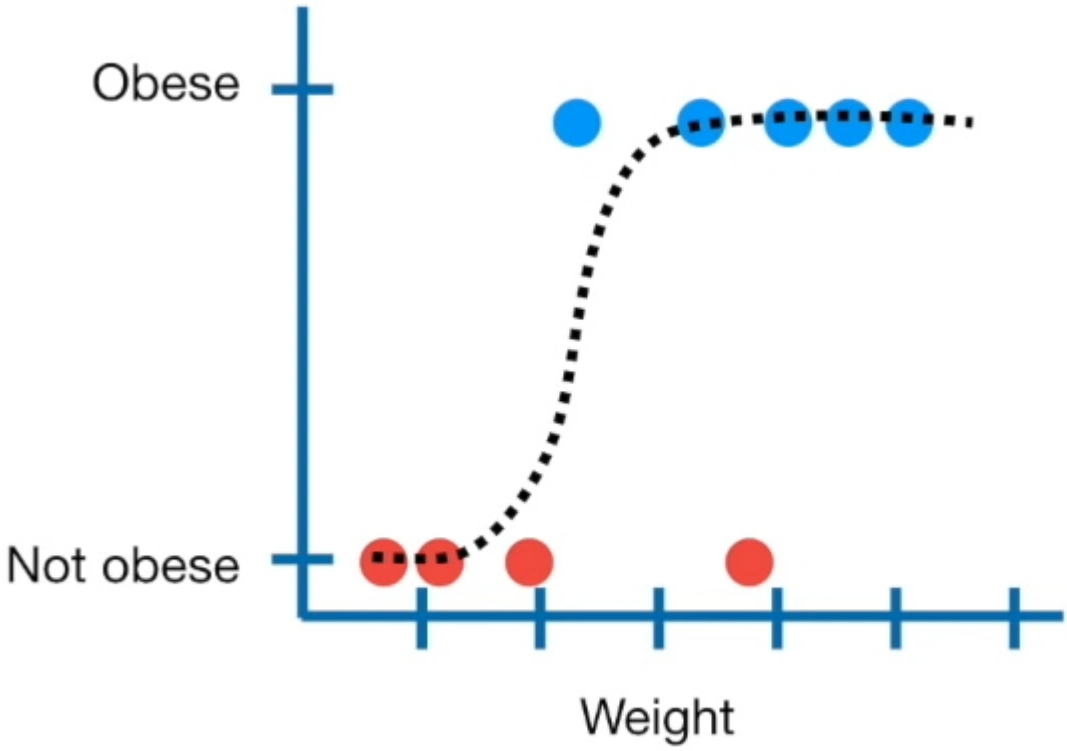
În încercarea de a găsi un clasificator mai bun decât SVC, am vrut să aleg ceva simplu, care nu e predispus la overfitting, și știam că metoda Bayes Naiv respectă aceste lucruri. Acesta se bazează pe formula lui Bayes, însă face o presupunere puternică, și aceea că proprietățile sunt independente. Acest lucru face și timpul de antrenare mult mai mic deoarece, având variablile independente, formula lui Bayes devine un produs al probabilităților fiecărei caracteristici.

Folosind această metodă, am obținut rezultate chiar bune, puțin mai slabe decât ale clasificatorului SVC, însă mult mai rapid decât acesta.

5.4 Regresie Logistică

În ciuda numelui, regresia logistică este un algoritm de clasificare, nu de regresie. Denumirea acestei metode vine de la funcția logistică (sigmoidă) care stă la bază. Deoarece calculează probabilitatea ca o intrare să fie de un anumit tip, acest algoritm este bun pentru o clasificare binară. El se bazează pe folosirea funcții sigmoide, pentru a calcula probabilitatea menționată.

În exemplul de mai jos, se observă curba determinată de funcția logistică.



Antrenarea acestui clasificator constă în aproximarea coeficienților funcției, folosind MLE (Maximum Likelihood Estimation). Formula de mai jos reprezintă un exemplu de ecuație logistică:

$$y = \frac{e^{b_0 + b_1 \cdot x}}{1 + e^{b_0 + b_1 \cdot x}}$$

, unde y reprezintă outputul prezis, b_0 reprezintă bias-ul, iar b_1 este coeficientul pentru o intrare x. Pentru fiecare coloană (caracteristică) există un coeficient b, care trebuie învățat din setul de antrenare.

Această metodă calculează probabilitatea ca o intrarea x să fie de o clasă y, prestabilită, după formula

$$P(X) = P(Y = 1|X)$$

Această probabilitate va fi comparată cu un prag, de obicei 0.5, după care se va stabili clasa vectorului respectiv. Regresia logistică este o metodă liniară, dar predicțiile sunt transformate folosind funcția logistică. Impactul acestei metode este că nu mai putem înțelege prezicerile ca o combinație liniară a inputurilor, ca în cazul regresiei liniare, de exemplu. În continuare, modelul poate fi descris de formula:

$$p(X) = \frac{e^{b_0 + b_1 X}}{1 + e^{b_0 + b_1 X}}$$

Logaritmând, obținem:

$$ln(p(X)) = b_0 + b_1 X - ln(1 + e^{b_0 + b_1 X})$$

$$ln(p(X)) + ln(\frac{1}{\frac{1}{1+e^{b_0+b_1X}}}) = b_0 + b_1X$$

$$ln(\frac{p(X)}{1 - \frac{e^{b_0+b_1X}}{1+e^{b_0+b_1X}}}) = b_0 + b_1X$$

$$ln(\frac{p(X)}{1 - p(X)}) = b_0 + b_1X$$

Ecuația a devenit din nou liniară, iar raportul din stânga reprezintă șansele.

$$ln(odds) = b_0 + b_1X$$

, după care exponențiem, și obținem:

$$odds = e^{b_0+b_1X}$$

Asta arată că, chiar dacă este neliniară, pleacă de la liniaritatea log-șanselor.

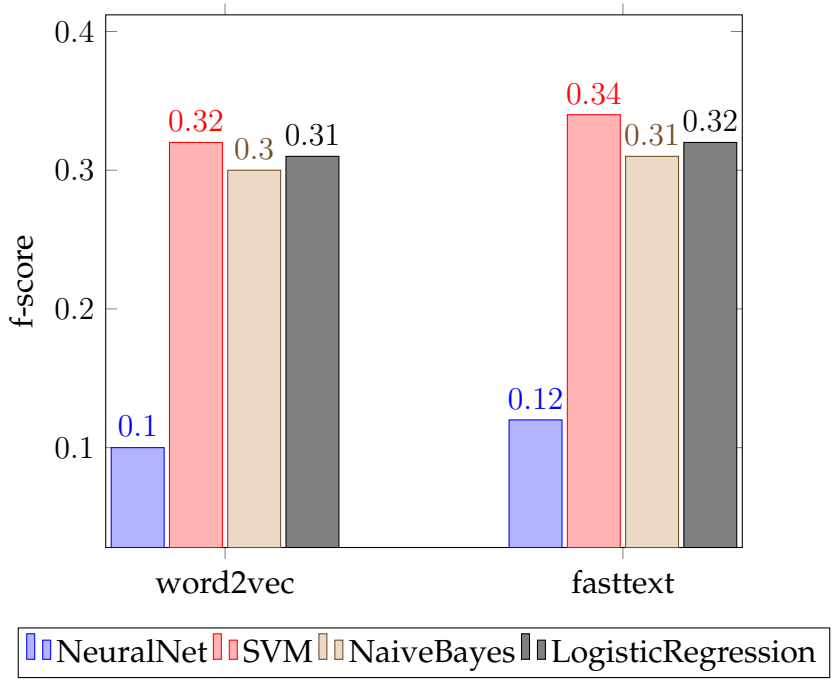
Rezultatele acestei metode au fost mai bune decât ale Bayesului Naive, dar mai slabe decât ale SVC-ului.

Capitolul 6

Rezultate

Rezultatele clasificatorilor pe setul de test au fost destul de mici, însă și faptul că sunt puține date de clasă 1 în acest set influențează acest lucru. Astfel, sunt doar 263 de tweet-uri de clasă 1, care, poate ar avea caracteristici diferite de tweet-urile din setul de antrenament. Din păcate nu am avut acces la setul de test oficial al concursului, care, din lucrarea celor de la echipa NRC-Canada, au fost 9961 de tweet-uri, dintre care 771 de clasă 1. Ar fi fost de mare ajutor acest set de date, deoarece și pentru antrenare aș fi avut mai multe tweet-uri de folosit.

În graficul de mai jos se poate vedea diferența dintre rezultate, între cele două metode de transformare a textului în vectori folosite, Word2Vec și fasttext, și pe baza clasificatorilor folosiți. De precizat este că setul de antrenament folosit este cel secționat la 70%, după care i-a fost aplicat algoritmul SMOTE pentru balansarea acestuia, până când datele de clasă 1 au devenit egale cu cele de clasă 0.



Am considerat că acest grafic pune cel mai bine în valoare fiecare metodă, fiind folosită cea mai bună metodă de balansare a datelor, celelalte metode nefiind la fel de eficiente.

Concluzii

În concluzie, sunt de părere că am obținut rezultate destul de bune, însă, dacă aș fi găsit un set de date mai mare, aș fi obținut rezultate mai bune. Din păcate nici metoda BERT a celor de la Google n-am putut-o folosi, neavând destulă memorie RAM în calculator, și consider că ar fi avut rezultate mai bune decât Word2Vec și fasttext.

Problema detectării efectelor adverse în mediul online, într-un mod automat, este în continuare o problemă deschisă, cu un grad de importanță sporit. Îmi doresc să revin asupra problemei când voi avea posibilitatea, și să încerc mai multe metode.

Bibliografie

- Algoritmul SMOTE
- Algoritmul folosit pentru determinarea unei linii în hiperspațiu
- fasttext
- word2vec
- n-grame
- lucrarea echipe NRC-Canada
- SVC, SVM
- Bayes Naiv
- Regresie logistică
- Documentație Gensim
- Documentație Keras
- Documentație scikit-learn