Documentație NitroNLP

Anton Marius Alexandru, Dumitrache Flavian,
Ilie Andrei-Virgil, Tălpigă Andrei-Vlad

May 2024

1 Prezentarea taskului

Concursul NitroNLP a avut ca cerință clasificarea articolelor de știri ca fiind articole de tip satiră sau nu. Datele de antrenare se aflau într-un fișier de tip csv și conțineau: titlul articolului, conținutul articolului și eticheta clasei din care făceau parte(0 pentru articole de tip non-satiră, respectiv 1 pentru satiră).

2 Preprocesarea și analizarea datelor

Pe parcursul analizării datelor am observat ca tag-ul apare mult mai frecvent(aproximativ de 30 de ori mai des) în știrile de tip satiră față de articolele non-satiră. Motivele pentru care acest tag apare mai des în satiră pot fi multiple, dar de obicei ar accentua ironia și sarcasmul pentru a sublinia anumite aspecte ale societății. Tag-ul este utilizat pentru a face aceste secvențe de text mai vizibile pentru cititor. De asemenea, satira nu ar trebui doar să amuze și să informeze publicul, dar să și capteze și să mențină atenția cititorului. Utilizarea elementelor vizuale puternice pot facilita acest lucru.

Alte elemente observate când am analizat datele au fost:

- Predispoziția utilizării multiplelor semne de exclamație și întrebare în articolele de tip satiră
- Utilizarea tagului **doar în non-satiră**. Acesta este utilizat pentru a evidenția cuvinte sau fraze importante într-un articol, ajutând la sublinierea aspectelor importante, dar și pentru a modula tonul unui articol, pentru a influența cum este interpretat textul de cititor.

Preprocesarea datelor este o etapă critică în fluxul de lucru al învățării automate. Înainte de a antrena modele, este esențial să ne asigurăm că datele sunt curate, relevante și bine structurate. Preprocesarea eficientă a datelor poate duce la îmbunătățirea semnificativă a performanței modelului și la accelerarea procesului de învățare.

Datele din setul de antrenare aveau valori lipsă, care au trebuit înlocuite cu valoarea "". Completarea valorilor lipsă și corectarea erorilor sunt pași extrem de importanți în preprocesarea informațiilor. Aceste acțiuni ajută la reducerea riscului de a introduce bias în model și facilitează învățarea de pattern-uri relevante mai degrabă decât anomalii.

Preprocesarea adecvată poate reduce semnificativ timpul necesar pentru antrenarea modelului, deoarece modelul poate converge mai rapid când datele sunt bine pregătite. Mai mult, acest lucru poate îmbunătăți acuratețea și fiabilitatea predicțiilor modelului, permițându-i să identifice corect și eficient pattern-urile din date.

Metodele încercate de noi pentru preprocesarea datelor au fost următoarele:

1. Tag-urile HTML au fost înlocuite cu echivalentele lor traduse în limba română, astfel încât modelul să poată înțelege mai bine structura și conținutul textului. Scopul acestei preprocesări are scopul de a curăța datele și de a le aduce la o formă mai ușor de înțeles pentru model. Traducerea tagurilor în limba română a fost pentru a putea face modelul să înțeleagă contextul mai bine. Un alt motiv este pentru că modelul este antrenat pe limba română, astfel este familiarizat cu subtilitățile limbii.

2. NER - Deoarece numele entităților nu aduc absolut niciun fel de plusvaloare în înțelegerea articolelor, le-am înlocuit cu tagurile specifice modelelor de NER - nume de localitate, nume de persoane, indici temporali etc. Am încercat să înlocuim entitățile cu tag-urile specifice în limba engleză, dar și în limba română. În mod surprinzător performanțele au fost mai ridicate atunci când entitățile erau în limba engleză.

3. Cuvintele de legătură - Eliminarea prepozițiilor, conjuncțiilor, sau articolelor este adesea practicată în preprocesarea datelor deoarece elimină zgomotoul și simplifică textul pentru model. Cu toate acestea, în contextul taskului nostru s-a dovedit că aceste cuvinte sunt relevante. Ele contribuie la înțelegerea contextului și a nuanțelor semantice ale textelor, facilitând astfel interpretări mai precise și recunoașterea relațiilor sintactice complexe. Prin urmare, menținerea acestor cuvinte în analiza noastră sporește capacitatea modelului de a înțelege și de a procesa limbajul natural într-un mod mai exact.

4. Eliminarea diacriticelor - Preprocesarea textului prin eliminarea diacriticelor este o tehnică foarte utilă, mai ales în limbile în care acestea sunt foarte întâlnite, precum limbile latine. Scopul acestei strategii este de a uniformiza forma cuvintelor, reducând astfel vocabularul și variabilitatea lexicală care poate apărea din cauza utilizării sau neutilizării diacriticelor. De exemplu, în română cuvintele: sărbătoare și sarbatoare ar putea fi considerate variante diferite ale aceluiași termen, dacă diacriticele nu sunt standardizate. Eliminarea acestora este mai ales utilă în sarcinile de analiză a sentimentelor - așa cum a fost task-ul acestui concurs.

Una dintre tehnicile de preprocesare care a generat cele mai bune rezultate în cadrul proiectului nostru a implicat înlocuirea tag-urilor HTML cu echivalentele lor în limba română. De exemplu, tag-ul a fost transformat în 'ÎNCEPUT_PUTERNIC'. Această abordare se bazează pe premisa că modelul, preantrenat pe limba română, este deja familiarizat cu semnificațiile anumitor structuri lingvistice. Prin înlocuirea tag-urilor, modelul este încurajat să asocieze aceste cuvinte cu contextul specific al clasei predominante în care acestea apar.

În același context, am ales să nu reducem multiplele semne de întrebare și exclamație la un singur caracter. Această decizie este fundamentată pe înțelegerea că modelul poate recunoaște și interpreta potențialul sarcastic sau accentuat al repetiției acestor semne, o caracteristică des întâlnită în articolele de tip satiră. Menținerea lor în forma originală ajută la păstrarea nuanțelor intenționate de autor, facilitând astfel o clasificare mai precisă a textelor.

3 Modelul 1: RoBERT

În procesul de învățare automată, mărimea și complexitatea modelului pot avea un impact semnificativ asupra performanței și timpului necesar pentru antrenare. Modelul RoBERT vine în trei variante de mărime: small, base și large, cu 19 milioane, 114 milioane, respectiv 341 milioane de parametri. Testele noastre inițiale au arătat că varianta large oferă cea mai bună acuratețe, datorită capacității sale de a capta și procesa un număr mai mare de caracteristici fine ale datelor.

Totuși, antrenarea modelului large este time-consuming și necesită resurse computaționale considerabile. Ca soluție, am început prin a experimenta și a optimiza tehnici de preprocesare a datelor pe modelul small. Aceasta strategie ne-a permis să identificăm combinațiile de preprocesare care îmbunătățesc cel mai mult performanța modelului, fără a consuma resurse excesive în faza de testare.

După ce am determinat care tehnici de preprocesare sunt cele mai eficiente pe modelul small, am aplicat aceste tehnici optimizate în antrenarea modelului RoBERT large. Antrenarea modelului large cu date preprocesate eficient a condus la o îmbunătățire substanțială a performanței, confirmând ipoteza că o pregătire adecvată a datelor poate reduce semnificativ varianța și erorile de predicție.

Am utilizat, de asemenea, tehnici de validare încrucișată pentru a asigura robustețea și fiabilitatea modelului final. Validarea încrucișată ne-a permis să evaluăm modelul în diverse scenarii de testare, asigurându-ne că performanța acestuia nu este specifică doar unui subset particular de date. Am încercat

împărțirea datelor în proporție de 10% validare și 90% antrenare, dar ulterior am realizat că o împărțire de 30% - 70% este mai reprezentativă pentru acuratețe pe întreg setul de date de testare.

4 Modelul 2: SVM (Support Vector Machines)

În primul rând, este necesar să prelucrăm datele pentru a le putea folosi în construirea modelului. Aceasta implică încărcarea datelor de antrenament și de test, eliminarea valorilor lipsă și extragerea caracteristicilor relevante din text. În acest scop, am folosit biblioteca **pandas** pentru manipularea datelor și biblioteca **scikit-learn** pentru prelucrarea textului.

Pentru a transforma textul într-o reprezentare numerică utilizabilă pentru algoritmi de învățare automată, am aplicat metoda **TF-IDF** (Term Frequency-Inverse Document Frequency). **TF-IDF** este o tehnică utilizată pentru a evalua importanța unui cuvânt dintr-un document într-un corpus de texte. Este format din două componente principale:

1. Term Frequency (TF): Măsoară cât de des apare un cuvânt într-un document. Este calculat ca raportul dintre numărul de apariții ale unui cuvânt și numărul total de cuvinte din document.

$$TF(t,d) = \frac{f(t,d)}{\sum_{i \in d} f(i,d)}$$

   Unde:

   - t este cuvântul
   - d este documentul
   - $f(t,d)$ este frecvența cuvântului t în documentul d
   - $\sum_{i \in d} f(i,d)$ este numărul total de cuvinte din documentul d

2. Inverse Document Frequency (IDF): Măsoară cât de rar apare un cuvânt în întregul corpus. Este calculat ca raportul dintre numărul total de documente și numărul de documente care conțin cuvântul.

$$IDF(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

   Unde:

   - N este numărul total de documente în corpus
   - $|\{d \in D : t \in d\}|$ este numărul de documente care conțin cuvântul t

Scorul TF-IDF este rezultatul înmulțirii scorului TF cu scorul IDF pentru fiecare cuvânt. Această tehnică favorizează cuvintele care apar frecvent într-un document, dar sunt rare în întregul corpus, ceea ce le face semnificative pentru conținutul documentului respectiv.

Pentru clasificarea textelor, am decis să utilizăm o mașină de vectori de suport (SVM - Support Vector Machine) cu un kernel liniar. **SVM**-urile (Mașinile de vectori suport) sunt o clasă de algoritmi de învățare supervizată folosite pentru probleme de clasificare și regresie. În cazul nostru, ne concentrăm pe utilizarea SVM pentru clasificarea textelor. Principiul de bază al SVM constă în găsirea unui hiperplan care să poată separa în mod optim două clase diferite de puncte în spațiul caracteristicilor. Pentru problemele de clasificare cu mai mult de două clase, SVM poate fi extins folosind tehnici precum One-vs-Rest sau One-vs-One. În cazul SVM cu kernel liniar, ca în soluția noastră, clasificarea este realizată prin identificarea unui hiperplan liniar care să separe în mod optim exemplele din diferite clase. Kernelul liniar este ales pentru că este eficient din punct de vedere computațional și se comportă bine în cazul unor seturi de date cu un număr mare de caracteristici.

Modelul SVM este antrenat folosind datele de training preprocesate. În acest proces, modelul ajustează parametrii săi pentru a găsi cea mai bună separare liniară între clasele de texte. Pentru a evalua performanța modelului, putem folosi diverse metrici, cum ar fi acuratețea sau precizia și recall-ul pentru fiecare clasă. Aceste metrici ne oferă o înțelegere a cât de bine se potrivesc clasificările modelului cu clasele reale. După antrenarea modelului, acesta poate fi folosit pentru a face predicții pe datele de test. Textele de test sunt prelucrate similar cu cele de antrenament și apoi sunt introduse în model pentru a prezice clasele corespunzătoare. Rezultatele prezicerilor sunt salvate într-un fișier CSV pentru a fi ulterior evaluate și comparate cu etichetele reale.

În concluzie, utilizând tehnici precum TF-IDF și SVM, putem construi modele eficiente pentru clasificarea textelor. TF-IDF ne permite să transformăm textul într-o reprezentare numerică care poate fi folosită de algoritmi de învățare automată, în timp ce SVM ne oferă un cadru robust pentru clasificarea acestor reprezentări. În această secțiune, am demonstrat cum aceste componente teoretice sunt implementate în practică pentru a obține o soluție de clasificare a textelor.