MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DETECTAREA AUTOMATĂ A LIMBAJULUI INSTIGATOR ȘI OFENSATOR PE TWITTER

PROIECT DE SEMESTRU

Student: **Morar Emilian Cristian**

Disciplina: **Sisteme bazate pe Cunoaștere**

2025

Cuprins

1 Introducere

1.1 Context general

Rețelele sociale au devenit un mediu predominant pentru exprimarea opiniilor, interacțiuni între oameni și distribuire de informatii. Aceste platforme au dus și la o creștere a discursului instigator și a limbajului ofensator, putand avea efecte negative asupra sanatatii mentale al utilizatorilor sau comunităților. Proiectul de Detectare Automată a Limbajului Instigator și Ofensator pe Twitter , abordează problema instigării și a injuriilor prin dezvoltarea unui model de învățare automată al acestora capabil sa identifice și sa cenzureze conținutul dăunător.

1.2 Obiective

Dezvoltarea unui Model de învățare automată: Scopul principal este de a dezvolta un model de învățare automată capabil sa detecteze limbajul instigator și limbajul ofensator. Acest model va fi antrenat pe un set de date bine structurat , care contine tweet-uri annotate.

Implementarea unei soluții de moderare a conținutului: Creearea unei aplicații care sa permita moderarea automată a conținutului. Aceasta va ajuta la identificarea și cenzurarea conținutului în timp real, asigurand un mediu sigur pentru utilizatori.

Cenzură limbajului ofensator: implementarea unei functionalitati care sa cenzureze automat limbajul ofensator din postarile utilizatorilor.

1.3 Specificații

Proiectul realizat este alcatuit din mai multe componente, fiecare avand cerințe specifice.Functionalitatile aplicației sunt:

1.Detectarea limbajului instigator și ofensator: Aplicație utilizează un model de învățare automată pe un set de date bine structurat pentru a clasifica tweet-urile în trei categorii: limbaj instigator, limbaj ofensator și neutru.

2.Cenzură limbajului ofensator: Utilizând biblioteca better profanity, aplicația va cenzură automat cuvintele ofensatoare din tweet-uri.(Neimplementat , poate fi adaugata aceasta functionalitate cu ajutorul librariei better-profanity).

3.Calcularea procentajului de tweet-uri: Acesta va calcula și va afișa procentajul de tweet-uri din fiecare categorie (limbaj instigator, limbaj ofensator și neutru) pentru a oferi o imagine de ansamblu asupra conținutului analizat.

4.Partea în care am testat tweet-uri introduse manual.

5.Salvarea rezultatelor: Rezultatele analizei,inclusiv tweet-urile originale,cenzurate și clasificările, vor fi salvate într-un fișier CSV pentru revizuire ulterioară.

Nivelul de performanță atins de model are o acuratete de minimum 88.94% in clasificarile tweet-urilor.

Limitele modelului sunt incapabilitatea lui de detectare a unui limbaj instigator sau ofensator subtil sau contextual. De asemenea, calitatea datelor de intrare poate influența performanța modelului. Altă limitare a modelului este antrenarea modelului cu date noi, putand fii implementata ulterior.

2 Cunoașterea și analiza setului de date

2.1 Detalii despre mediul de implementare

Setul de date train.csv este esențial pentru antrenarea modelului, acesta contine tweet-uri annotate, fiecare avand etichete care indica natura conținutului. Setul de date contine următoarele coloane:

- tweet: Textul tweet-ului. Aceasta este informația principala care va fi analizata si clasificata.
- hate speech count: Numărul de notari care clasifica tweetul ca limbaj instigator.
- offensive language count: Numărul de notari care clasifica tweetul ca limbaj ofensator.
- neither count: Numărul de notari care clasifica tweetul ca fiind nici limbaj instigator sau ofensator.
- count: Numărul total de notari pentru tweet.Acesta este suma tuturor notarilor din celelalte coloane.

Bibliotecile și instrumentele utilizate pentru crearea aplicației:

- Pandas: utilizata pentru manipularea și analiza datelor, permite citirea seturilor de date CSV și efectuarea de operații de prelucrare a datelor.
- NumPy: folosită pentru operații numerice și gestionarea array-urilor.
- Scikit Learn: folosită pentru implementarea algoritmilor de învățare automată,inclusiv regresia logistică și evaluarea modelului.
- NLTK: utilizata pentru procesarea limbajului natural, inclusiv pentru eliminarea cuvintelor frecvențe și reducerea dimensionalitatii prin aducerea cuvintelor la forma lor de baza.
- Matplotlib și Seaborn: folosite pentru vizualizarea datelor, graficelor de distribuție și matricea de corelație.
- Joblib: utilizata pentru salvarea și încărcarea modelului antrenat.

Proiectul a fost dezvoltat în Python 3.11.9, folosind ca IDE VSCode. De asemenea resursele necesare NLTK, cum ar fi stopwords si wordnet au fost descarcate pentru a sprijinii prelucrarea textului.

2.2 Analiza setului de date

Statisticile descriptive ale setului de date, acesta avand 27,783 de tweeturi sunt prezentate mai jos în figura.1 , incluzând numărul total de notari, media, deviația standard, valorile minime și maxime și procentele.. Și distribuția numerică pentru fiecare clasa este afișată în Figura.2.

Statistici descriptive:

	count	hate_speech_count	offensive_language_count	neither_count	class	clasa
count	24783.000000	24783.000000	24783.000000	24783.000000	24783.000000	24783.000000
mean	3.243473	0.280515	2.413711	0.549247	1.110277	1.110277
std	0.883060	0.631851	1.399459	1.113299	0.462089	0.462089
min	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	3.000000	0.000000	2.000000	0.000000	1.000000	1.000000
50%	3.000000	0.000000	3.000000	0.000000	1.000000	1.000000
75%	3.000000	0.000000	3.000000	0.000000	1.000000	1.000000
max	9.000000	7.000000	9.000000	9.000000	2.000000	2.000000

Figura 2.1:Statisticile descriptive

Distribuția claselor:
1: 19190
2: 4163
0: 1430

Figura 2.2: Distributia tweet-urilor pentru fiecare clasa.

În figura 2.3 este prezentă distribuirea datelor în setul de antrenare și setul de testare:

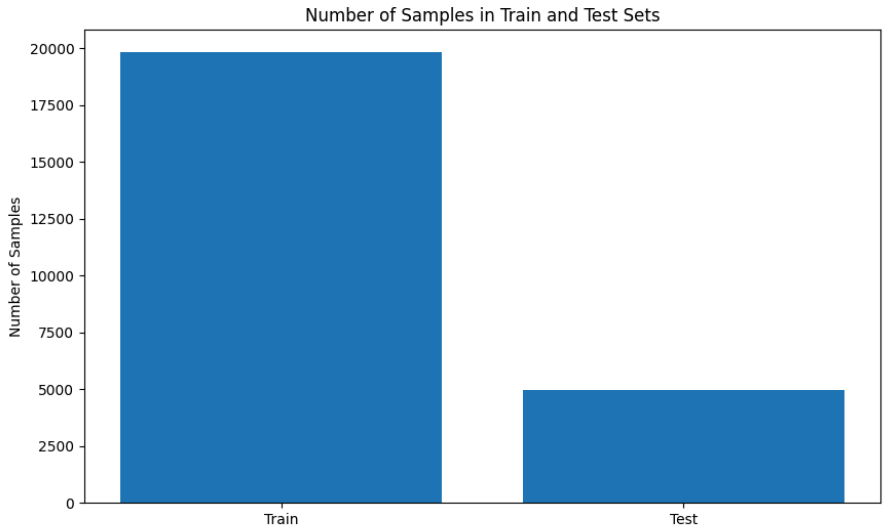

Figura 2.3:Distribuția datelor de train și test

Matricea de confuzie este utilă pentru evaluarea performanței modelului de clasificare.Aceasta reprezinta vizual predictiile modelului comparativ cu valorile reale. (Figura 2.4)

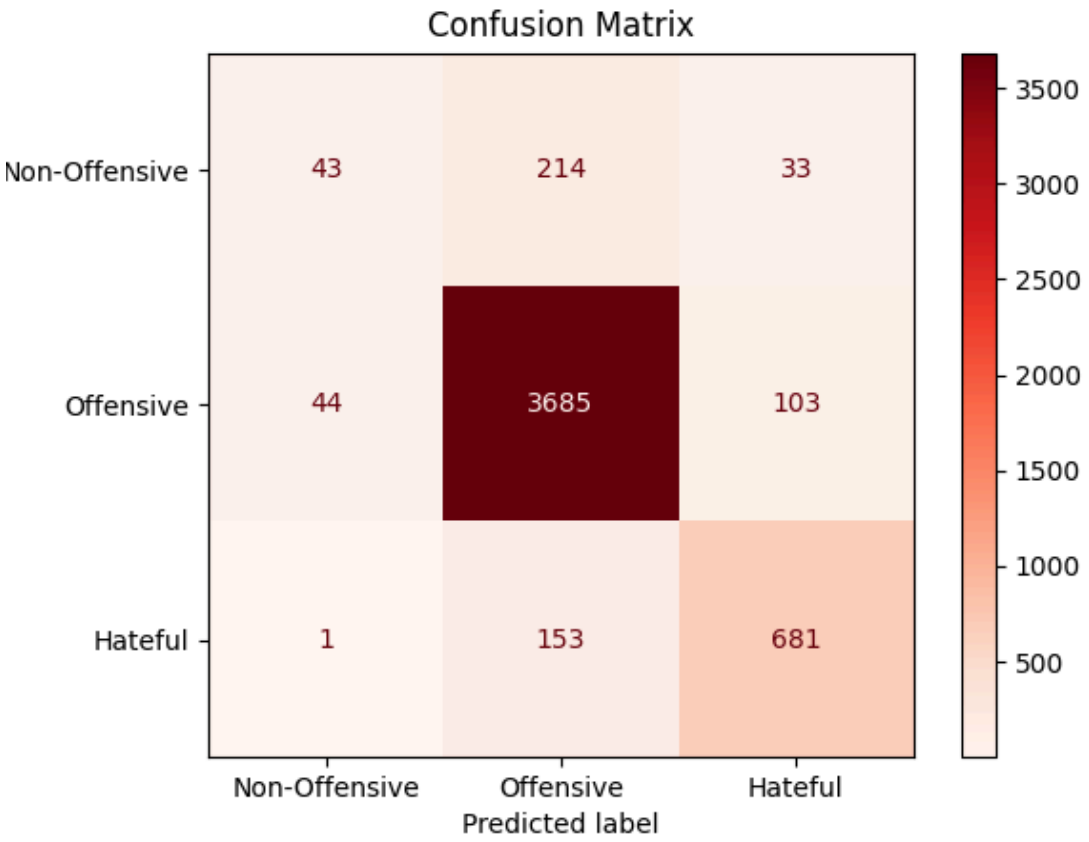

Figura 2.4:Matricea de confuzie

Analiza corelațiilor măsoară corelatia dintre două variabile, indicând dacă și cât de mult se influențează reciproc. Valorile apropiate de 1 sugerează o corelatie pozitiva puternica si -1 reprezinta o relatie inversa. Analiza de corelație a setului de date train.csv este reprezentată în Figura 2.5.

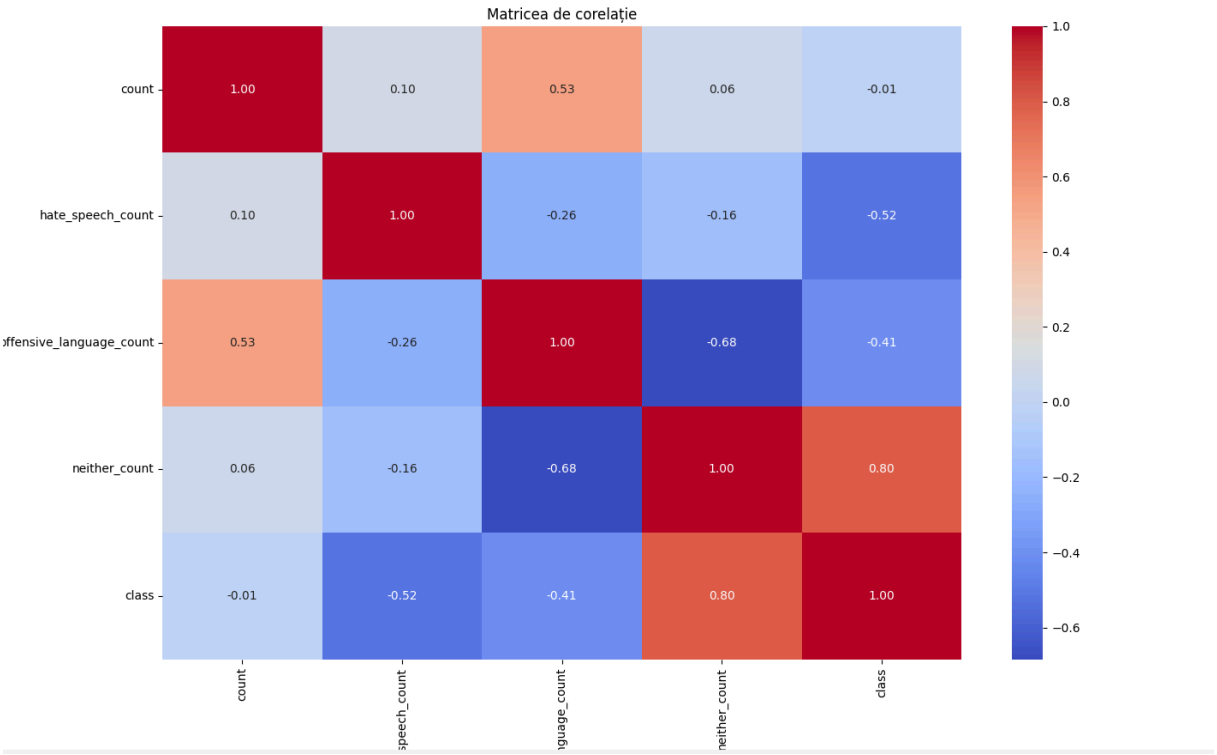

Figura 2.5:Matricea de corelație

Rezultate observate în toate graficele , datele din train.csv se axează mai mult pe limbajul ofensator. Așadar, ne așteptam ca acuratețea următoarelor experimente să nu funcționeze foarte bine pentru mesaje neutre și mesaje instigatoare la ura, dar o sa funcționeze foarte bine pentru mesaje ofensatoare.

Modelul are o acuratețe de: 88.94%. Folosind formula:

Acuratete=Nr. Predictii Corecte/Nr. total de predictii x 100;

In cod :

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

3 Preprocesarea setului de date

3.1 Metodele utilizate pentru preprocesarea setului de date

Eliminarea zgomotului din text este un pas esențial în preprocesarea datelor pentru modelele de procesare a limbajului natural, contribuind la îmbunătățirea acuratetii clasificarii. Eliminarea manerelor de utilizator (@user) ajuta la reducerea informațiilor relevante, prevenind influența numelor specifice asupra analizei sentimentelor sau clasificarii textului. Eliminarea link-urilor și etichetelor HTML reduce conținutul inutil, asigurand că modelul analizează doar textul esențial și nu elemente externe nesemnificative. Inlaturarea semnelor de punctuație și a caracterelor speciale ajuta la uniformizarea textului și previne confuziile generate de simboluri fără valoare semantica în contextul clasificarii. Stop Words Urile sunt eliminate pentru a reduce dimensiunea textului procesat și a îmbunătăți performanța modelului, pastrand doar cuvintele relevante pentru interpretarea semantica.

Dimensionality reduction reprezintă un proces important în preprocesarea datelor de text, care are rolul de a reduce numărul de caracteristici într-un set de date, păstrând totuși informația relevantă. În cazul acestui cod, reducerea dimensionalitatii poate fi realizată folosind tehnici precum PCA (Principal Component Analysis) sau TruncatedSVD, aplicate pe vectorul de caracteristici obținut de la vectorizer. Aceasta ajută la accelerarea procesului de clasificare și la îmbunătățirea performanței modelului prin eliminarea zgomotului din datele de intrare. În cazul unor modele de clasificare text, reducerea dimensionalitatii poate contribui la îmbunătățirea acurateței, prin eliminarea redundanței dintre caracteristici. Un avantaj major al acestei tehnici este că, prin reducerea numărului de dimensiuni ale datelor de intrare, se poate evita overfitting ul, contribuind astfel la generalizarea mai bună a modelului pe date noi.

Librăriile folosite pentru preprocesare:

- Pandas: O biblioteca esențiale pentru manipularea și analiza datelor. Este utilizata pentru a citi fișier CSV și pentru a gestiona datele într-un format tabular (DataFrame).

- NLTK: O biblioteca pentru procesarea limbajului natural. Este folosită pentru tokenizare, eliminarea cuvintelor de legatura și alte sarcini de prelucrare a textului.
- re: O biblioteca standard în Python pentru manipularea textului folosind expresii regulate. Este utilizata pentru a cauta si a inlocui modele în text, cum ar fi eliminarea URL-urilor si a mentiunilor de utilizator.
- NumPy: O biblioteca pentru manipularea eficienta a array-urilor si a calculului numeric. Este adesea utilizata impreuna cu Pandas.

3.2 Rezultatele obtinute si informații necesare pentru realizarea proiectului

Se importa librăriile necesare pentru preprocesarea datelor și antrenarea modelului de clasificare, cum ar fi nltk si sklearn. Datele sunt curățate prin eliminarea handle-urilor de pe rețelele sociale, URL-urilor, semnelor de punctuatie si a cuvintelor de legatura folosind funcții dedicate. Setul de date este împărțit în seturi de antrenament și testare, iar textul este vectorizat folosind TfidfVectorizer pentru a-l transforma într-un format numeric. Modelul de regresie logistica este antrenat pe datele vectorizate și evaluat pe setul de testare prin acuratețe și raport de clasificare. La final, modelul și vectorizatorul sunt salvate pentru a fi utilizate ulterior în predicții pe date noi.Rezultate după pre procesarea textului este textul filtrat care asigura performanțe mai bune.

4 Modelarea sistemului

4.1 antrenareModel1.py – Antrenarea Modelului

Se utilizează mai multe librării esentiale:

- pandas – pentru citirea și manipularea datelor
- nltk – pentru preprocesarea textului
- sklearn – pentru modelarea și evaluarea performanțelor modelului
- matplotlib și seaborn – pentru vizualizarea datelor
- joblib – pentru salvarea și încărcarea modelului antrenat

Preprocesarea datelor:

- Citirea setului de date: Datele sunt încărcate într-un fișier CSV (train.csv).
- Curatarea textului: Se elimina elemente nedorite precum:
- Mentiuni de utilizatori (@user)
- URL-uri
- Punctuatie excesiva
- HTML/unicode entities
- Stop Words (cuvinte fără semnificație semantică importantă)

Tokenizare și vectorizare:

- Se utilizează TfidfVectorizer pentru a transforma textul într-o reprezentare numerică

Antrenarea modelului:

- Se împarte setul de date în date de antrenare și date de testare
- Se antrenează un model de clasificare folosind Logistic Regression
- Se evaluează performanța modelului utilizând:

    Acuratețea (accuracy_score)

    Raportul de clasificare (classification_report)

Matricea de confuzie

Matricea de corelație

Salvarea modelului:

Modelul antrenat și vectorizatorul sunt salvate în fișiere model clasificare.joblib și vectorizator.joblib pentru a fi utilizate ulterior

4.2 predictie_manual.py - Clasificarea unui tweet introdus manual

Acest script permite utilizatorului sa introduca manual un tweet pentru a fi clasificat de model.

Pașii executați:

- Încărcarea modelului antrenat:Se încarcă modelul și vectorizatorul utilizând joblib.load()
- Preprocesarea textului introdus:Se aplica aceleasi transformari ca in antrenareModel1.py (eliminare mentiuni, URL-uri, stopwords etc.)
- Vectorizarea textului:Textul este transformat în vector numeric folosind TfidfVectorizer
- Predicția:Se utilizează model.predict() pentru a clasifica tweetul
- Afișarea rezultatului:Tweetul este clasificat în una dintre cele trei categorii: Non-Offensive, Offensive, Hateful.

4.3 preluare_dateTw.py - Colectarea de tweet-uri de pe Twitter

În acest script se folosește API-ul de la platforma X pentru a extrage Tweet-uri.

Etapele procesului:

- Citirea credentialelor:Credentialele sunt încărcate din credentials.csv pentru autentificare
- Configurarea API-ului Twitter:Se autentifica accesul utilizand cheile API
- Extragerea tweet-urilor:Se colecteaza tweet-uri in limba engleza folosind tweepy.Cursor().Tweet-urile sunt salvate într-un fișier CSV (date_noi_tweeter.csv)

4.4 predictie.py - Clasificarea unui set de tweet-uri

Datele colectate de pe platforma X vor ajunge în csv-ul date noi tweeter.csv de unde cu ajutorul script-ului predictie.py vom categoriza fiecare tweet.

Pașii executați:

- Încărcarea datelor din date noi tweeter.csv
- Preprocesarea textului:Aplicarea aceleiasi metode de curatare ca in antrenareModel1.py
- Vectorizarea textului:Transformarea textului in reprezentare numerica
- Clasificarea tweet-urilor:Aplicarea modelului antrenat pentru predictie
- Afișarea rezultatului:Tweet-urile sunt etichetate și aflate alături de predicțiile respective

5 Concluzii

5.1 Rezultate obținute

Prin testări succesive ale aplicației în diverse condiții și utilizând scriptul predicție manual.pe, care permite introducerea manuala a propozitiilor, am observat ca aceasta are o performanta foarte buna în detectarea cuvintelor obscene în limba engleza. Cu toate acestea, cea mai mare provocare intampinata de model consta în diferențierea dintre limbajul instigator și cel ofensator. În unele cazuri, propozitii simple sunt clasificate eronat ca fiind ofensatoare sau instigatoare. O soluție eficienta pentru îmbunătățirea acuratetii modelului este antrenarea acestuia pe un set de date mai extins și mai diversificat, care sa includă un număr mai mare de exemple de limbaj neutru și instigator, contribuind astfel la o clasificare mai precisă și mai nuanțată.

5.2 Direcții de dezvoltare

Pentru a imbunatati performanta sistemului de clasificare a tweet-urilor, se pot explora mai multe directii. Se poate extinde setul de date prin colectarea de tweet-uri suplimentare din diverse surse și limbi. Un model mai complex, precum RNN sau BERT, ar permite o analiza mai detaliată a textului. Combinarea analizei textuale cu imagini și metadate (detecție multimodala) ar duce la o clasificare mai precisă. Un sistem de feedback ar ajuta la imbunatatirea continua a modelului, iar optimizarea performanței ar reduce timpul de procesare. Introducerea unor categorii suplimentare, ca sarcasm sau dezinformare, ar oferi o analiza mai nuanțată. Aceste îmbunătățiri ar crește acuratețea și fiabilitatea sistemului.

6 Bibliografie

Api Twitter: https://docs.x.com/x-api/tools-and-libraries/overview

Antrenarea modelului:

https://www.kaggle.com/code/saramoha88/hate-speech-and-offensive-language-detection

Libraria NLTK: https://www.nltk.org/

Librarie joblib: https://joblib.readthedocs.io/en/stable/