Hate speech, offensive language identification, misogyny / stereotype detection

-Hate speech Analysis-

Echipa 8

Ianuarie 2025

Contents

1 Introducere **3**

2 Setul de date **3**

3 Extragerea feature-urilor **5**

3.1 CBoW 5

3.2 TF-IDF 5

4 Modele de clasificatori **5**

4.1 Multinomial Naive Bayes 5

4.2 Ridge Classifier 6

4.3 SVC (kernel ‘rbf’) 7

4.4 Linear SVC 8

4.5 Gradient Boosting 9

4.6 MLP Classifier 10

4.7 XGBoost 11

4.8 Modele preantrenate (BERT) 12

4.9 Deep Neural Network 15

5 Observații **16**

6 Interpretarea rezultatelor **17**

7 Bibliografie **18**

1 Introducere

Echipa noastră (Cărpineanu Alexandru, Cuciureanu Dragoș-Adrian și Vasile George-Cristian din grupa 506) a ales tema ”#15 Hate speech, offensive language identification, misogyny / stereotype detection”.

Discursul ce instigă ură - denumit în continuare hate speech - este diferit de libera exprimare, deoarece nu este limitat la exprimarea unor opinii controversate, în schimb implică propagarea discriminării și a violenței. Hate speech-ul a fost întotdeauna o problemă cu care societatea a trebuit să se confrunte, însă, de la apariția internetului, acesta a fost propulsat în ochii tuturor oamenilor. Orice utilizator putea să întâlnească vorbe ofensive fără sa vrea.

Problema hate speech-ului este prevăzută în legile multor țări, deși interpretarea acesteia poate fi variată și identificarea sa mai dificilă. În special în mediul online, unde volumul la care sunt transmise informațiile, filtrarea canalelor de convorbiri și restricționarea accesului pentru entitățile ce vorbesc cu ură este o problemă ce necesită o soluție eficientă.

Majoritatea platformelor sociale folosesc deja inteligența artificială pentru a detecta hate speech-ul și a-l cenzura automat, însă mereu se pot aduce îmbunătățiri acestor modele pentru a ne asigura că cenzurarea se aplică doar asupra hate speech-ului, orice greșeală a modelului putând să însemne o încălcare a dreptului la liberă exprimare a utilizatorilor.

2 Setul de date

Pentru alegerea setului de date am folosit platforma Hugging Face datorită numărului variat de opțiuni pe care îl oferă. Am ales setul de date “tweets_hate_speech_detection”.

Am ales platforma Twitter deoarece postările care puteau fi făcute pe aceasta până recent erau limitate la 280 caractere, astfel tweet-urile fiind un mijloc simplu și bun pentru detectarea hate speech-ului. În contextul nostru, am definit problema clasificării unui tweet ca fiind unul de hate speech prin identificarea în conținutul unor texte ce au conotații rasiste și/sau sexiste.

Astfel, fiecare tweet din setul de date este etichetat cu 0 sau cu 1, eticheta 1 semnificând prezența rasismului/sexismului. Cele 31.962 de tweet-uri valabile sunt scrise majoritar în limba engleză. Dintre

acestea doar 2.242 sunt clasificate cu 1, astfel tweeturile cu 0 fiind predominante. Pentru acest fapt, am selectat doar 10.000 dintre acestea.

Din punct de vedere al impărțirii datelor, am selectat 60% pentru antrenare si 40% pentru testare (ponderile fiind ușor de modificat, dacă se dorește mărirea setului de antrenare și diminuarea celui de testare). Totodată, setul de date a fost preprocesat pentru a ajunge la o formă normalizată potrivită pentru clasificatori.

Pentru preprocesare am aplicat următoarele operații:

- Transpunerea textelor unicode în caractere cu cod ASCII.
- Transformarea literelor mari în litere mici, deoarece această caracteristică a lor nu este relevantă în procesul nostru.
- Înlăturarea hyperlinkurilor, ele fiind adrese către resurse web și neprezentând nicio semnificație lingvistică.
- Eliminarea numelor de utilizator, identificate prin caracterul "@" urmat de un cuvant format doar din litere, numere și simbolul "_".
- Eliminarea hashtag-urilor. În postări de hate speech hashtag-urile pot fi utilizate pentru a organiza întâlniri bazate pe promovarea unor ideologii rasiste sau instigative la ură, poluand analiza conținutului propriu-zis al postarii.
- Înlăturarea punctuatiei, neavând o influența în identificarea hate speech-ului.
- Înlăturarea caracterelor numerice.
- Normalizarea caracterelor repetate. Utilizatorii serviciilor de chat au tendința de a multiplica vocale sau uneori și consoane din diferite cuvinte pentru a întări anumite idei. În orice caz, aceasta forma a mesajelor nu este relevantă în identificarea hate speech-ului, de asemenea pot îngreuna procesul de analiză al modelelor prin nerecunoașterea anumitor cuvinte.
- Lematizarea, având rolul de a reduce varietatea lingvistică redundantă prin proiecția diferitelor forme ale cuvintelor (conjugări, declinări, articulații) în forma lor de baza.
- Eliminarea stopwords. Acestea reprezintă cele mai comune cuvinte dintr-o limbă și în general sunt prepozitii, conjunctii, articole, neavând sens de sine stătător.
- Eliminarea "caracterelor albe", precum "space", "tab", "enter".

3 Extragerea feature-urilor

Am ales două metode de a extrage feature-urile, **TF-IDF** (Term Frequency - Inverse Document Frequency) și **CBoW** (Continuous Bag of Words). Am antrenat și testat fiecare model pe ambele tipuri de feature.

3.1 CBoW

Continuous Bag of Words este o metodă de extragere a feature-urilor ce utilizează o rețea neuronală pre-antrenată cu 2 layere, care reconstruiește contextul cuvintelor primite ca input dupa vectorii creați în urma antrenării. Aceasta ajustează n-vectorii, astfel încat cuvintele găsite în contexte similare apar apropiate unele de altele in spațiul vectorial creat la antrenare.

3.2 TF-IDF

Term frequency - Inverse document frequency este un mod de extragere a feature-urilor ce evaluează importanța fiecărui cuvânt din corpus, în relație cu numărul său de apariții în fiecare document.

Spre exemplu, cuvinte precum ‘the’ sau ‘is’ au o importanță mult mai mică față de ‘blacklives’ într-un corpus de hate-speech.

TF-IDF nu ia in considerare contextul în care regăsim cuvintele in propoziție.

4 Modele de clasificatori

Am ales o gamă variată de modele, pentru a observa diferențele dintre ele. Mai mult, am folosit și modele pre-antrenate preluate tot de pe platforma HuggingFace pentru a avea un spațiu de comparatie mai mare.

4.1 Multinomial Naive Bayes

Un algoritm probabilist care utilizează teorema lui Bayes ce presupune independența caracteristicilor pentru a decide clasa din care face parte input-ul.

Parametrii:

- **alfa**: Un parametru de netezire. Ajută la evitarea overfitting ului.

	precision	recall	f1-score	support
0	0.84	1.00	0.91	4021
1	0.99	0.12	0.22	876
accuracy			0.84	4897
macro avg	0.92	0.56	0.57	4897
weighted avg	0.87	0.84	0.79	4897

Figure 1: Tf-Idf

	precision	recall	f1-score	support
0	0.85	0.97	0.91	4021
1	0.64	0.22	0.32	876
accuracy			0.84	4897
macro avg	0.75	0.60	0.62	4897
weighted avg	0.81	0.84	0.80	4897

Figure 2: CBoW

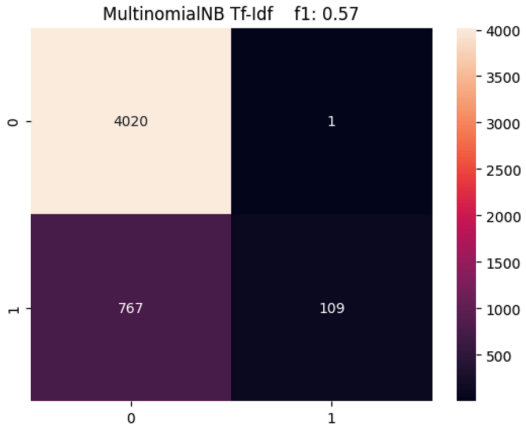

Figure 3: Tf-Idf

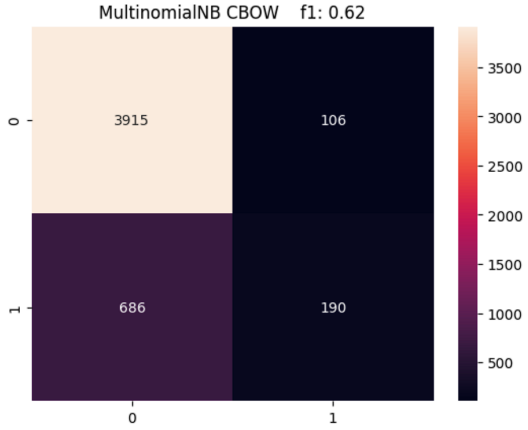

Figure 4: CBoW

4.2 Ridge Classifier

Clasificator binar care folosește regresia Ridge. Acesta penalizează complexitatea pentru a evita overfitting ul.

Parametrii:

- **alfa**: Utilizat pentru a controla puterea regularizarii. Reduce variația aproximărilor.
- **solver**: Algoritmul folosit pentru a rezolva problema de optimizare asociată modelului.

	precision	recall	f1-score	support
0	0.93	0.99	0.96	4021
1	0.91	0.67	0.77	876
accuracy			0.93	4897
macro avg	0.92	0.83	0.86	4897
weighted avg	0.93	0.93	0.92	4897

Figure 5: Tf-Idf

	precision	recall	f1-score	support
0	0.88	0.98	0.92	4021
1	0.77	0.36	0.49	876
accuracy			0.87	4897
macro avg	0.82	0.67	0.71	4897
weighted avg	0.86	0.87	0.85	4897

Figure 6: CBoW

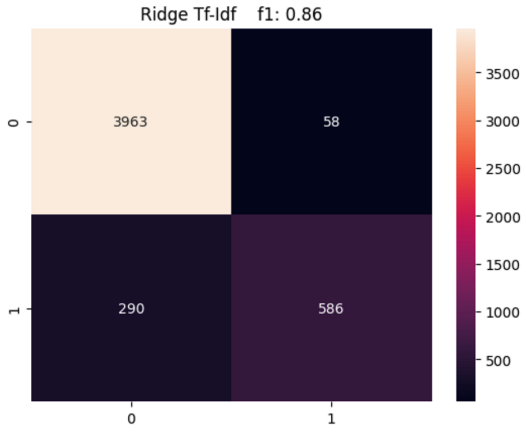

Figure 7: Tf-Idf

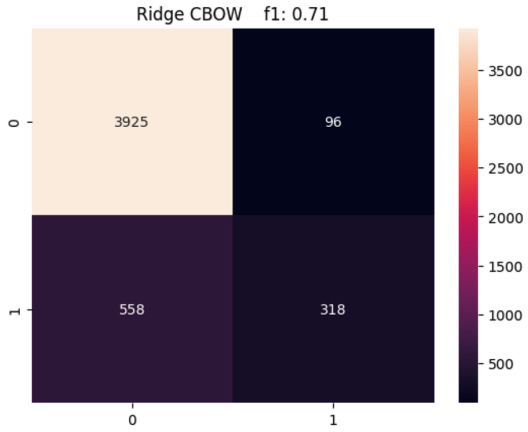

Figure 8: CBoW

4.3 SVC (kernel 'rbf')

Un algoritm de clasificare binară ce funcționează prin transformarea inputului într-un spațiu dimensional mai mare folosind funcții neliniare numite funcții radiale de baza (RBF) kernel. În acest spațiu mai mare, algoritmul încearca sa găsească un hyperplan optim care separa cele doua clase de date maximizand separarea dintre clase (distanta minima dintre un element al primei clase și un element al celei de-a doua).

Câțiva parametrii cheie sunt:

- **C**: Parametrul de regularizare, ce determină balanța dintre a obține o margine mai mare și permiterea unor clasificări eronate. O valoare mai mică a lui C permite o margine mai mare, iar o valoare mai mică a sa implică o clasificare mai strictă.
- **gamma** (coeficientul kernel): Defineste influența unui singur exemplu de antrenament asupra liniei de decizie. Un gamma mai mic înseamnă o influență mai largă și poate rezulta într-o linie de decizie mai uniforma, in timp ce un gamma mare inseamna o influenta mai restransa si poate duce la o linie de decizie mai complexă și mai ondulată.
- **kernel**: Alegerea kernelului poate avea un impact semnificativ asupra performanței algoritmului.

RBF SVC utilizează implicit kernelul RBF, dar pot fi utilizate și alte kerneluri, cum ar fi cel liniar, polinomial sau sigmoidal. Fiecare kernel are proprii sai hyperparametri, cum ar fi gradul pentru kernelurile polinomiale sau coef0 pentru kernelurile sigmoidale, care controlează comportamentul acestora.

	precision	recall	f1-score	support
0	0.91	0.99	0.95	4021
1	0.93	0.56	0.70	876
accuracy			0.91	4897
macro avg	0.92	0.78	0.83	4897
weighted avg	0.92	0.91	0.91	4897

Figure 9: Tf-Idf

	precision	recall	f1-score	support
0	0.89	0.97	0.93	4021
1	0.79	0.44	0.56	876
accuracy			0.88	4897
macro avg	0.84	0.71	0.75	4897
weighted avg	0.87	0.88	0.86	4897

Figure 10: CBoW

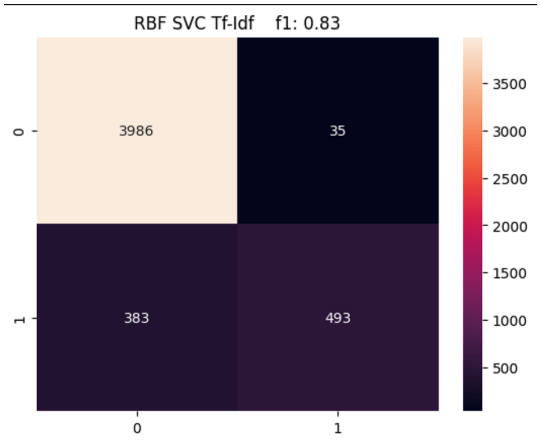

Figure 11: Tf-Idf

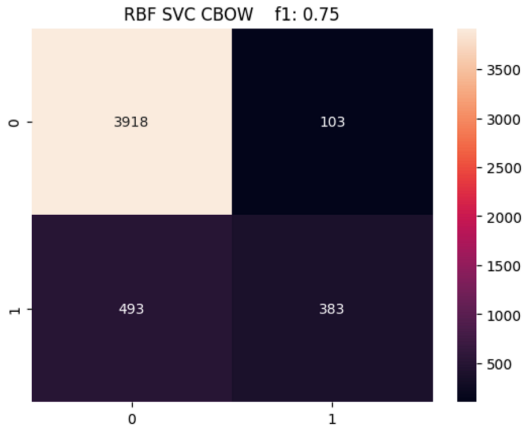

Figure 12: CBoW

4.4 Linear SVC

Algoritm de clasificare folosind masini cu vectori suport. Se ocupă cu căutarea hyperplanelor care separă clasele de date într-un spațiu dimensional mai mare. Este asemanator cu RBF SVC, însă folosește un kernel linear, iar hyperplanul rezultat este o linie dreaptă.

	precision	recall	f1-score	support
0	0.94	0.98	0.96	4021
1	0.89	0.71	0.79	876
accuracy			0.93	4897
macro avg	0.91	0.85	0.88	4897
weighted avg	0.93	0.93	0.93	4897

Figure 13: Tf-Idf

	precision	recall	f1-score	support
0	0.89	0.97	0.93	4021
1	0.76	0.47	0.58	876
accuracy			0.88	4897
macro avg	0.83	0.72	0.75	4897
weighted avg	0.87	0.88	0.87	4897

Figure 14: CBoW

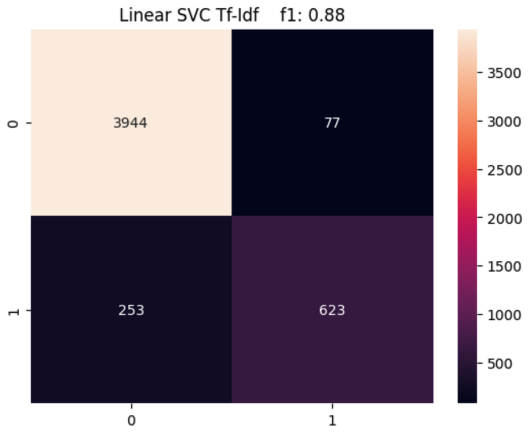

Figure 15: Tf-Idf

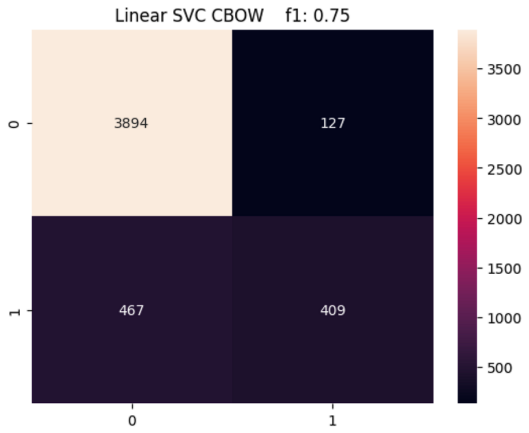

Figure 16: CBoW

4.5 Gradient Boosting

Acest algoritm construiește un model aditiv prin etape, permițând optimizarea funcțiilor de pierdere diferențiabile arbitrare.

	precision	recall	f1-score	support
0	0.90	0.98	0.94	4021
1	0.87	0.51	0.64	876
accuracy			0.90	4897
macro avg	0.88	0.74	0.79	4897
weighted avg	0.90	0.90	0.89	4897

Figure 17: Tf-Idf

	precision	recall	f1-score	support
0	0.89	0.97	0.93	4021
1	0.76	0.47	0.58	876
accuracy			0.88	4897
macro avg	0.83	0.72	0.76	4897
weighted avg	0.87	0.88	0.87	4897

Figure 18: CBoW

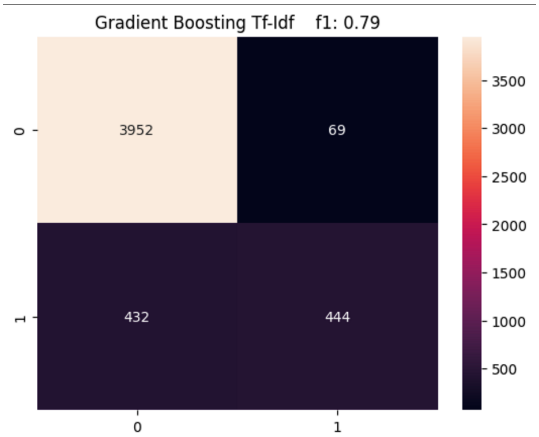

Figure 19: Tf-Idf

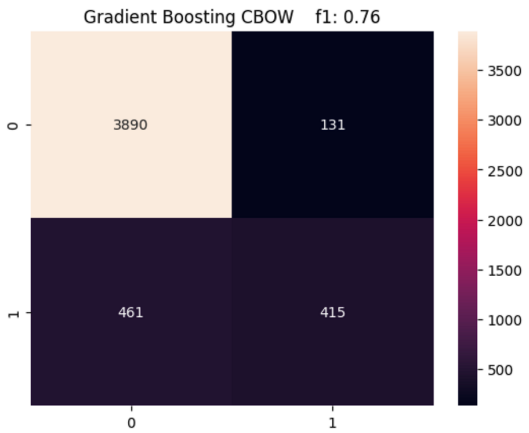

Figure 20: CBoW

4.6 MLP Classifier

Constă din mai multe straturi de neuroni artificiali. Orice rețea neuronală are un strat de intrare. Pe lângă acestea doua, un MLP are un număr variat de straturi ascunse. Fiecare neuron primește input de la un alt neuron dintr-un strat precedent, îl modifică și îl trimite mai departe până la stratul de ieșire, unde se obține rezultatul final al rețelei.

Câțiva parametrii cheie sunt:

- **solver**: Algoritmul care rezolvă problema de optimizare.
- **alfa**: Puterea termenului de regresie Ridge.
- **hidden_layer_sizes**: Numărul de neuroni din straturile ascunse.
- **max_iter**: Numărul maxim de iterații.
- **random_state**: Determină generarea de numere aleatoare pentru inițializare.

	precision	recall	f1-score	support
0	0.95	0.97	0.96	4021
1	0.83	0.76	0.80	876
accuracy			0.93	4897
macro avg	0.89	0.86	0.88	4897
weighted avg	0.93	0.93	0.93	4897

Figure 21: Tf-Idf

	precision	recall	f1-score	support
0	0.91	0.90	0.91	4021
1	0.57	0.60	0.58	876
accuracy			0.85	4897
macro avg	0.74	0.75	0.75	4897
weighted avg	0.85	0.85	0.85	4897

Figure 22: CBoW

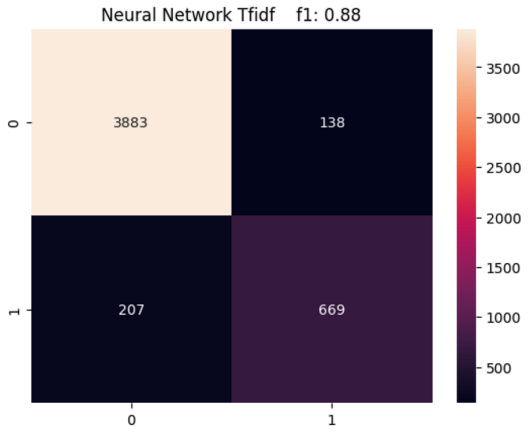

Figure 23: Tf-Idf

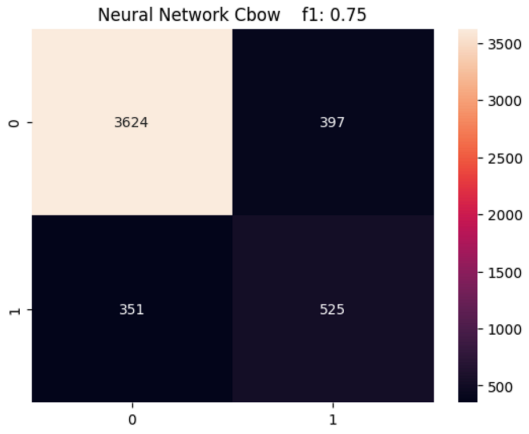

Figure 24: CBoW

4.7 XGBoost

Extreme Gradient Boosting este un algoritm proiectat pentru a optimiza performanța modelelor de gradient boosting ce folosește combinații de tehnici de boosting și regularizare. XGBoost îmbunătățește algoritmii tradiționali de gradient boosting prin introducerea mai multor inovații cheie: Optimizarea bazată pe gradient, regularizarea, tăierea arborilor, tratarea valorilor lipsa. Este eficient în rezolvarea problemelor de clasificare, regresie și clasificare a rangului.

Câțiva parametrii cheie sunt:

- **eta** (sau learning_rate): Are rolul de a preveni overfittingul.
- **gamma** (sau min_split_loss): Specifica reducerea minima a pierderii necesara pentru a produce o divizare.
- **max_depth**: Adâncimea maximă a unui arbore. Este folosit în controlul overfitting-ului, intrucat adâncimea mai mare va permite modelului sa invete relații foarte specifice unui sample particular. Crescand aceasta valoare modelul devine mai complex și este foarte probabil sa produca overfitting.
- **subsample**: Rata de subsampling a instanțelor de învățare. Valori mai mici fac algoritmul mai conservator si previn overfittingul, insa valori prea mici pot conduce la underfitting.
- **colsample_bytree**: Rata de subsampling a coloanelor cand se construiește fiecare arbore. Subsampling-ul are loc odată pentru fiecare arbore construit.
- **n_estimators**: Numărul de estimatori.

	precision	recall	f1-score	support
0	0.93	0.97	0.95	4021
1	0.83	0.64	0.73	876
accuracy			0.91	4897
macro avg	0.88	0.81	0.84	4897
weighted avg	0.91	0.91	0.91	4897

Figure 25:

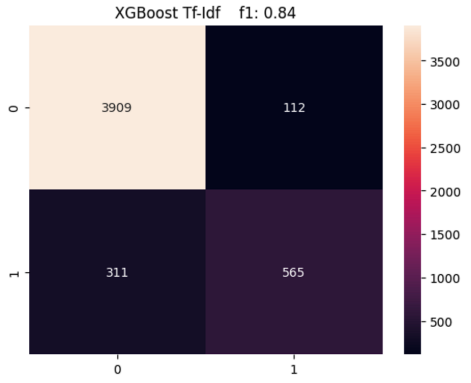

Figure 26:

4.8 Modele preantrenate (BERT)

Modele preantrenate folosite:

1. BERTicelli: Obținut prin rafinarea modelului BERT (Bidirectional Encoder Representations from Transformers) modelul din [3] a fost antrenat pe urmatoarele date: Offensive Language Identification Dataset. Link model.

	precision	recall	f1-score	support
0	0.85	0.92	0.89	4021
1	0.43	0.26	0.32	876
accuracy			0.80	4897
macro avg	0.64	0.59	0.60	4897
weighted avg	0.78	0.80	0.79	4897

Figure 27: Raport de clasificare

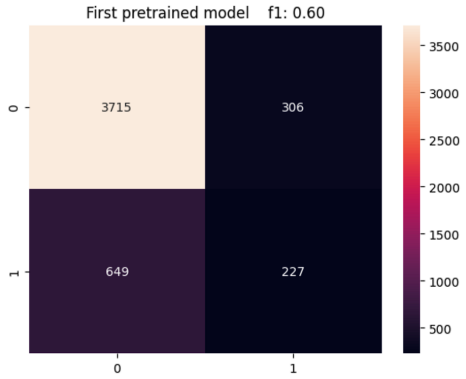

Figure 28: Matrice de confuzie

2. bert-base-uncased-ear-mlma: O alta rafinare a modelului BERT, modelul din [2] a fost antrenat pe acest dataset: MLMA, folosind regularizarea atentiei bazata pe entropie. Link model.

	precision	recall	f1-score	support
0	0.91	0.49	0.63	4021
1	0.25	0.77	0.37	876
accuracy			0.54	4897
macro avg	0.58	0.63	0.50	4897
weighted avg	0.79	0.54	0.59	4897

Figure 29: Raport de clasificare

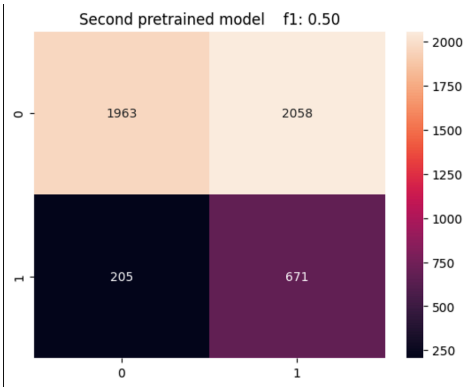

Figure 30: Matrice de confuzie

3. roberta-hate-speech-dynabench-r4-target: Model BERT preantrenat din [5] trained on dynamically generated datasets. Link model.

	precision	recall	f1-score	support
0	0.85	0.96	0.90	4021
1	0.53	0.19	0.28	876
accuracy			0.82	4897
macro avg	0.69	0.58	0.59	4897
weighted avg	0.79	0.82	0.79	4897

Figure 31: Raport de clasificare

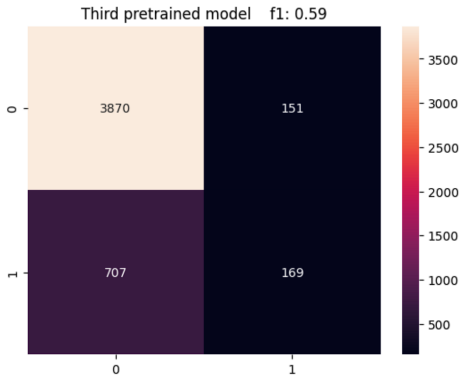

Figure 32: Matrice de confuzie

4. bert-base-uncased-hatexplain: Forma de baza a modelului BERT din [4] andtrenat pe urmatorul Dataset. Link model.

	precision	recall	f1-score	support
0	0.83	0.98	0.90	4021
1	0.51	0.07	0.13	876
accuracy			0.82	4897
macro avg	0.67	0.53	0.51	4897
weighted avg	0.77	0.82	0.76	4897

Figure 33: Raport de clasificare

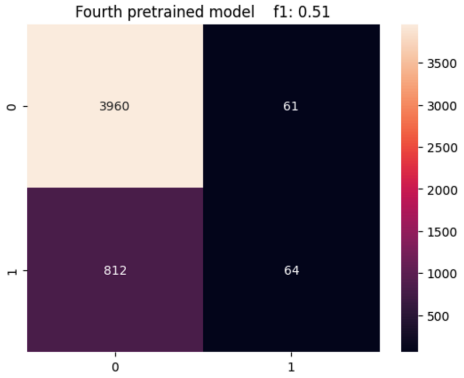

Figure 34: Matrice de confuzie

5. dehatebert-mono-english: O alta rafinare pentru limba engleza a modelului BERT, modelul din [1] a fost antrenat pe mai multe dataseturi, precum: Dataset 1, Dataset 2, Dataset 3, etc. Link model.

	precision	recall	f1-score	support
0	0.86	0.84	0.85	4021
1	0.34	0.38	0.36	876
accuracy			0.76	4897
macro avg	0.60	0.61	0.60	4897
weighted avg	0.77	0.76	0.76	4897

Figure 35: Raport de clasificare

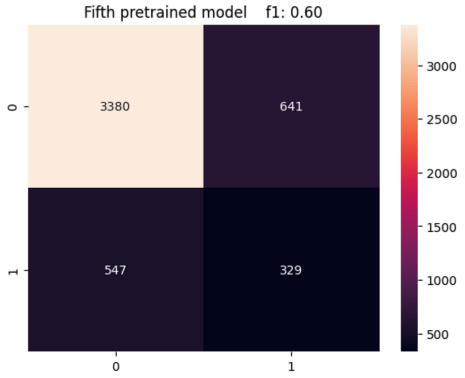

Figure 36: Matrice de confuzie

Apoi, pe baza rezultelor, am luat cele mai bune 2 BERT-uri in functie de Macro F1 (primul, respectiv al cincilea model) si le-am dat fine-tune. Urmatoarele sunt rezultatele

1. Fine-tuned BERTicelli

	precision	recall	f1-score	support
0	0.95	0.96	0.95	4021
1	0.80	0.76	0.78	876
accuracy			0.92	4897
macro avg	0.87	0.86	0.87	4897
weighted avg	0.92	0.92	0.92	4897

Figure 37: Raport de clasificare

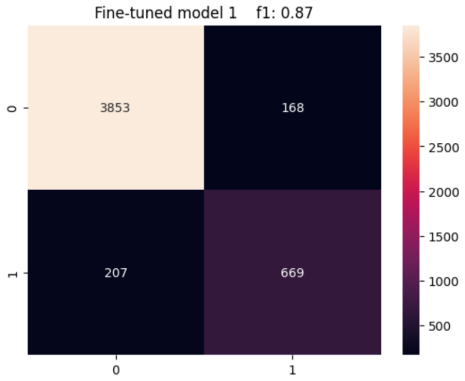

Figure 38: Matrice de confuzie

2. Fine-tuned dehatebert-mono-english

	precision	recall	f1-score	support
0	0.94	0.95	0.94	4021
1	0.75	0.74	0.75	876
accuracy			0.91	4897
macro avg	0.85	0.84	0.85	4897
weighted avg	0.91	0.91	0.91	4897

Figure 39: Raport de clasificare

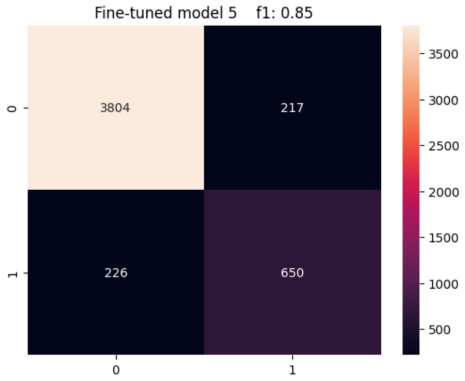

Figure 40: Matrice de confuzie

Aceste 2 BERTuri fine-tuned au avut perfomanțe foarte bune, primul model (BERTicelli) având overall una dintre cele mai cea mai bune performanțe din cadrul proiectului.

4.9 Deep Neural Network

Rețelele Neuronale, cunoscute și ca Rețele Neuronale Artificiale sau Rețele Neuronale Simulate, sunt "inima" algoritmilor de învățare în profunzime (deep learning). Structura și numele lor sunt inspirate de creierul uman, imitând modul în care neuronii transmit semnale de la unul la altul.

În compoziția lor, rețelele neuronale artificiale au straturi de noduri (sau neuroni), dintre care unul este stratul de input, unul de output si celelalte sunt staturile ascunse. Fiecare nod este conectat la altul și are asociată o pondere și un prag. Dacă outputul unui neuron depășește valoarea pragului, neuronul este activat și trimite informația la următorul neuron din rețea, aflându-se în următorul strat. Structura rețelei create de către noi este: 300 x 256 x 256 x 2 (300 fiind dimensiunea unui feature, iar 2 este pentru outputul binar: hate, non-hate). Cum datele noastre nu sunt excepțional de complexe am utilizat 2 hidden layers și am obținut rezultate bune, dar în mod optim am fi avut mai multe hidden layers (între 3 și 5). Totuși, această modificare ar mări foarte mult timpul de antrenare al modelului, am încercat asta. însă după ce a rulat câteva ore nu a avut un increase substanțial al eficienței.

Rețelele neuronale se bazează pe învatarea cu date de antrenare și își îmbunătățesc capacitațile treptat, însă odată rafinate, devin aparate puternice de mare viteză de clasificare si grupare în clustere a datelor. Am folosit CrossEntropyLoss pe post de funcție de pierdere și functia Adam (Adaptive Moment Estimation) pentru optimizarea modelului cu $learning\_rate = 0.001$.

Un model bazat pe învatarea în profunzime este Word2vec, recunsocut pentru utilizările sale în NLP. El învată asocierile dintre grupuri mari de cuvinte, utilizând un vector numeric pentru reținerea fiecărui cuvant, redând similaritatea acestora. Printre algoritmii utilizati se află skip-gram și CBoW.

Câțiva parametrii cheie sunt:

- **vector_size**: Dimensiunea vectorilor de cuvinte.
- **feature_size**: Dimensiunea vectorului de feature-uri.
- **workers**: Determină numărul de threaduri folosite pentru antrenarea modelului.
- **epochs**: Numărul de epoci în care se iterează textul.

- **total_examples**: Numărul total de propoziții.

	precision	recall	f1-score	support
0	0.92	0.97	0.95	4021
1	0.81	0.64	0.71	876
accuracy			0.91	4897
macro avg	0.87	0.80	0.83	4897
weighted avg	0.90	0.91	0.90	4897

Figure 41: Raport de clasificare

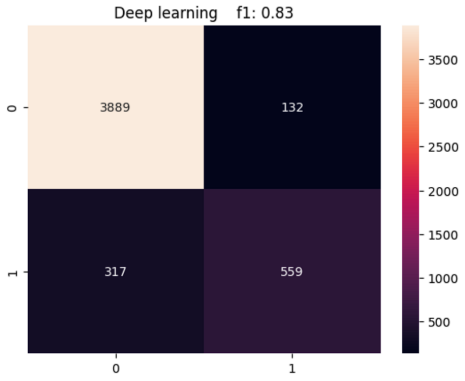

Figure 42: Matrice de confuzie

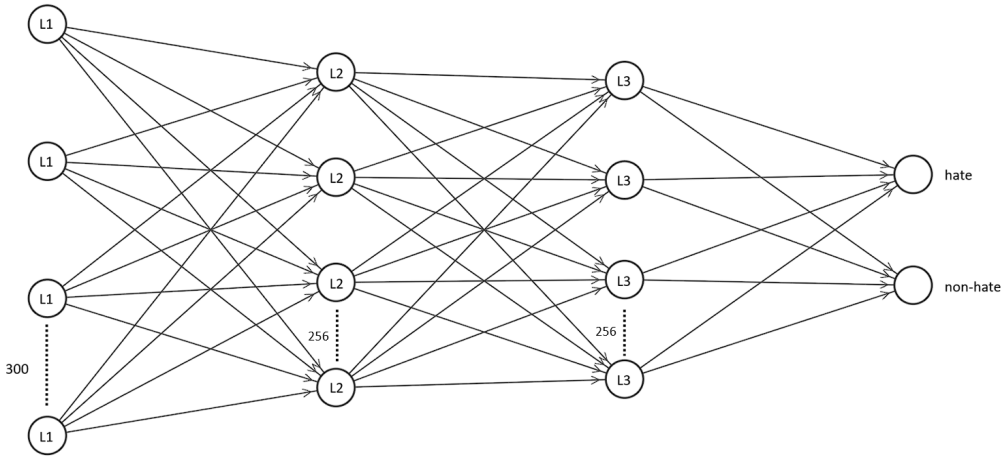

Figure 43: Reprezentare vizuală

5 Observații

Modificările principale ce au avut cel mai mare impact asupra rezultatelor au fost:

- Utilizarea n-gramelor în TF-IDF a crescut semnificativ performanța modelelor, cu $\approx 6\%$.
- Păstrarea conținutului hashtag-urilor în TF-IDF a adus o îmbunătățire de $\approx 5\%$ în majoritatea modelelor.
- Am utilizat GridSearchCV pentru a da fine tune la clasificatori.

6 Interpretarea rezultatelor

Cele mai bune rezultate din perspectiva macro f1 le-au avut următoarele modele:

- **Linear SVC** $\approx 88\%$.
- **MLP Classifier** $\approx 88\%$.
- **Ridge Classifier** $\approx 86\%$.
- **XGBoost** $\approx 83\%$.
- **DNN (2 hidden layers și 100 de epoci)** $\approx 83\%$.

Setul de date a fost adnotat de către 3 oameni și majoritatea decidea labelul final al unui mesaj. Chiar și așa foarte multe dintre comentarii sunt controversate din punct de vedere al clasificării, depinzând foarte mult de subiectivitate. Din această cauză, modele preantrenate ne fine-tuned au avut o eficiență mai redusă decât celelalte.

Câteva dintre datele controversate ar fi (următoarele comentarii sunt marcate ca fiind hate speech, iar marea majoritate din modele nu le-au clasificat corect):

- ”so proud to have a mayor like @user who isnt afraid to speak the truth..and yes he is completely right”
- ”@user some interesting #aicles #education”
- ”beware in marketing tainting messages”

Din perspectiva unei automatizări ce dorește să faciliteze moderarea unei aplicații de socializare, ar trebui să avem grijă cum tratăm următoarele cazuri. În funcție de cum arată diagonala secundară a matricei de confuzie, dacă avem predominant false pozitive vom avea mesaje normale ce au fost marcate ca hate speech, iar în cazul false negative vice versa. Din perspectiva aplicației, prima speță ar însemna că se elimină mesaje normale incorect, iar în cea de a doua că păstrăm și mesaje ce instigă hate speech. În opinia noastră, modelul ar trebui să fie mai restrictiv (cu mai multe intrări pe false pozitive decât pe false negative sau minimizând false negative aka maximizând recallul), dar doar ca marcare a mesajelor și acestea să fie mai departe moderate de către un om. Astfel, bazat pe analiza anterioară considerăm că cel mai bun model dintre cele utlizate a fost BERTul **BERTicelli**.

Bibliografie

[1] S. S. Aluru, B. Mathew, P. Saha, and A. Mukherjee. Deep learning models for multilingual hate speech detection. arXiv preprint arXiv:2004.06465, 2020.

[2] G. Attanasio, D. Nozza, D. Hovy, and E. Baralis. Entropy-based attention regularization frees unintended bias mitigation from lists. arXiv preprint arXiv:2203.09192, 2022.

[3] L. Grotti and P. Quick. Berticelli at haspeede 3: Fine-tuning and cross-validating large language models for hate speech detection. world, 2(3):4, 2021.

[4] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee. Hatexplain: A benchmark dataset for explainable hate speech detection. arXiv preprint arXiv:2012.10289, 2020.

[5] B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela. Learning from the worst: Dynamically generated datasets to improve online hate detection. In ACL, 2021.