



Departamentul Automatică și Informatică Industrială

**Facultatea Automatică și Calculatoare
Universitatea Națională de Știință și Tehnologie**

POLITEHNICA din București

Splaiul Independenței 313, 060042, București, România

Sala ED 412, Tel. 021/402.92.69

www.aii.pub.ro, email: secretariat.aii@upb.ro



Sesiunea de Comunicări Științifice Studențești

Ediția 2025

Filtru pentru emailuri bazat pe RN și LLM

Autor : Sorin-Ioan-Alexandru Bîrchi, Facultatea de Automatica și Calculatoare, anul III, Grupa 332AB

Adresa e-mail : sorin_ioan.birchi@stud.acs.upb.ro

Îndrumător științific : Conf.dr.ing. Ștefan MOCANU

Cuprins:

1. Introducere	pag. 3
1.1 Context	pag. 3
1.2 Obiective	pag. 3
2. Prezentarea domeniului din care face parte lucrarea	pag. 4
3. Descrierea problemei și prezentarea soluției	pag. 4
3.1. Implementari similare	pag. 4
3.2. Arhitectura lucrării	pag. 4
3.3. Interacțiunea dintre utilizator si aplicație(Front end)	pag. 5
3.4. Interacțiunea dintre Front end si Back end	pag. 7
3.4.1. Conectarea la server-ul de mail	pag. 8
3.5. Clasificarea textului folosind LLM-uri	pag. 9
3.6. Clasificarea fișierelor audio	pag.11
3.7. Clasificarea fișierelor foto	pag.12
3.8. Clasificarea fișierelor video	pag.12
4. Prezentarea aplicației	pag.13
4.1. Framework-uri utilizate	pag.13
4.2. Descrierea modelului	pag.13
5. Concluzii	pag.14
6. Bibliografie	pag.15

1. Introducere

1.1. Context

Securitatea și integritatea datelor, cât și a vieții noastre în mediul online, reprezintă un subiect destul de important, care, de-a lungul timpului, a devenit din ce în ce mai relevant și, totodată, mai îngrijorător. Digitalizarea și transmiterea informației în mediul online aduc un avantaj absolut, acesta fiind accesibilitatea, dar vin și cu dezavantaje, cum ar fi pericolul covârșitor asupra datelor și a integrității online. Informațiile la care avem acces pe internet nu sunt întotdeauna benefice pentru noi, uneori având chiar nenorocul de a primi sau de a găsi informații cu caracter tulburător.

Un exemplu foarte bun pentru afirmația de mai sus este reprezentat de e-mailurile cu tentă ofensatoare sau tulburătoare, fie prin textul prezentat, fie prin conținutul atașamentelor din e-mailurile respective.

1.2. Obiective

Proiectul vizează găsirea unei soluții pentru filtrarea mail-urilor care au un caracter tulburător utilizând tehnici de inteligență artificială, astfel scalabilitatea sa să fie una sporită din punct de vedere al clasificării mail-urilor. Această metoda utilizează diferite tehnici pentru analiza de

text, cea foto/video și audio, care includ LLM (Large Language Models) și CNN (Convolutional Neural Networks).

De ce tehnici de inteligență artificială?

- Antrenarea utilizând diferite seturi de date ce presupun texte, imagini și sunete, având posibilitatea testării și îmbunătățirii modelelor viitoare folosind aceste seturi de antrenare și testare care pot fi actualizate cu timpul.
- Paralelizarea proceselor utilizând containere docker și orchestratoare pentru utilizarea modelelor într-un mod paralel și distribuit.
- Scalabilitatea și deschiderea către îmbunătățirea ulterioară a modelelor într-o manieră mult mai simplă.
- Excluderea erorilor grosolane datorită fundamentelor matematice utilizate în construirea modelelor.
- Generalizarea oferită de modelele de machine learning în favoarea noilor date ținând cont de contextul acesta

2. Prezentarea domeniului din care face parte lucrarea

Proiectul face parte din domeniul **Inteligenței Artificiale** mai precis, al **tehnichilor deep learning**. Cu o creștere covârșitoare în ultimii ani, AI-ul s-a dezvoltat extrem de mult ajungând să fie în prim planul tuturor oamenilor de la apariția transformer-elor pe care majoritatea oamenilor le folosesc pentru task-uri zilnice. Un exemplu foarte bun în acest sens este ChatGPT, numele menționând chiar acest lucru, Generative Pre-trained Transformer (GPT). Cum acest domeniu este în continuă creștere foarte multe companii au început să adopte aceste modele fiind integrate ca agenți în produsele lor.

Limitările curente ale inteligenței artificiale sunt cele de nivel hardware, dar cu siguranță acest domeniu va continua să crească mult întrucât începe să fie integrat în toate lucrurile care ne înconjoară, de la motoarele de căutare pe internet și până la aplicațiile uzuale de pe calculatoare, toate aceste aplicații au integrați agenți și algoritmi de AI.

3. Descrierea problemei și prezentarea soluției

3.1. Implementari similare

Din punct de vedere al similarității rezultatului proiectul *PurgoMalum* profanity filter [1] este un exemplu oarecum bun întrucât este un serviciu ce filtrează mail-urile ce conțin text obscen. *PurgoMalum* analizează mail-urile sub formă șirurilor de caractere, iar la găsirea unui cuvânt obscen acesta cenzurează întregul cuvânt cu caractere '*'.

Un alt exemplu ar fi filtrul integrat din *Office 365* care în mod similar cu *PurgoMalum* cenzurează sau blochează mail-urile ce conțin unul sau mai multe cuvinte dintr-un set. Diferență majoră cu această soluție este că lista de cuvinte poate fi modificată de utilizator după preferințele acestuia.

Pentru partea de atașamente o soluție de automatizare a mail-urilor este oferită tot de Microsoft prin platforma *Microsoft Flow* [2] ce permite utilizatorului să își creeze propriile automatizări pentru mail-uri urmărind textul cât și atașamentele acestora.

3.2. Arhitectura lucrării

Lucrarea a fost gândită pe un model de tipul SaaS (Software as a Service) deoarece am vrut ca acest filtru să fie independent de extensiile browser sau de integrarea completă într-o casuță de mail. Modelul ce va fi prezentat poate fi folosit pe mai multe servere de mail, nu doar pe unul singur, astfel oferind o generalizare bună.

Din punct de vedere al interacțiunii dintre utilizator și aplicație găzduirea se face pe un server web prin care fiecare utilizator va fi nevoit să își facă un cont care va fi legat de mail-ul pe care vrea să se facă filtrarea respectivă, ulterior putând din platforma să pornească sau să oprească acest filtru. Platforma web a fost realizată cu HTML, Tailwind CSS, React, Vite, Python și Nginx.

În partea de backend a platformei exista un LLM și un CNN care se ocupă de clasificarea mail-urilor, ulterior printr-un pipeline aceste mail-uri fiind șterse dacă este cazul. Pentru a ține evidența utilizatorilor platformei folosim și o bază de date cu un singur tabel în MongoDB.

3.3. Interacțiunea dintre utilizator și aplicație (Front end)

Interacțiunea dintre utilizator și aplicația web este una foarte rapidă, ușoară și intuitivă.

Pagina web este foarte simplificată, această conținând doar trei rute, acestea fiind:

- Ruta către pagina de log in
- Ruta către pagina de sign up
- Ruta către dashboard-ul utilizatorului unde poate activa sau dezactiva filtrul

Astfel, pentru ca un utilizator să își creeze contul va trebui să își facă un cont cu mail-ul pe care vrea să se facă filtrarea, parolă pentru cont și parola pentru aplicațiile de mail (Password app). Password app este modul prin care serverele de mail permit aplicațiilor externe să se conecteze la conturi și să modifice mail-uri (să șteargă, să scrie, să salveze, etc.) cât și să modifice o parte din setările contului respectiv. Fiecare server de mail are modul său propriu prin care generează aceste parole de aplicație, care ulterior vor fi folosite prin framework-ul imaplib pentru conectarea la serverele de mail.

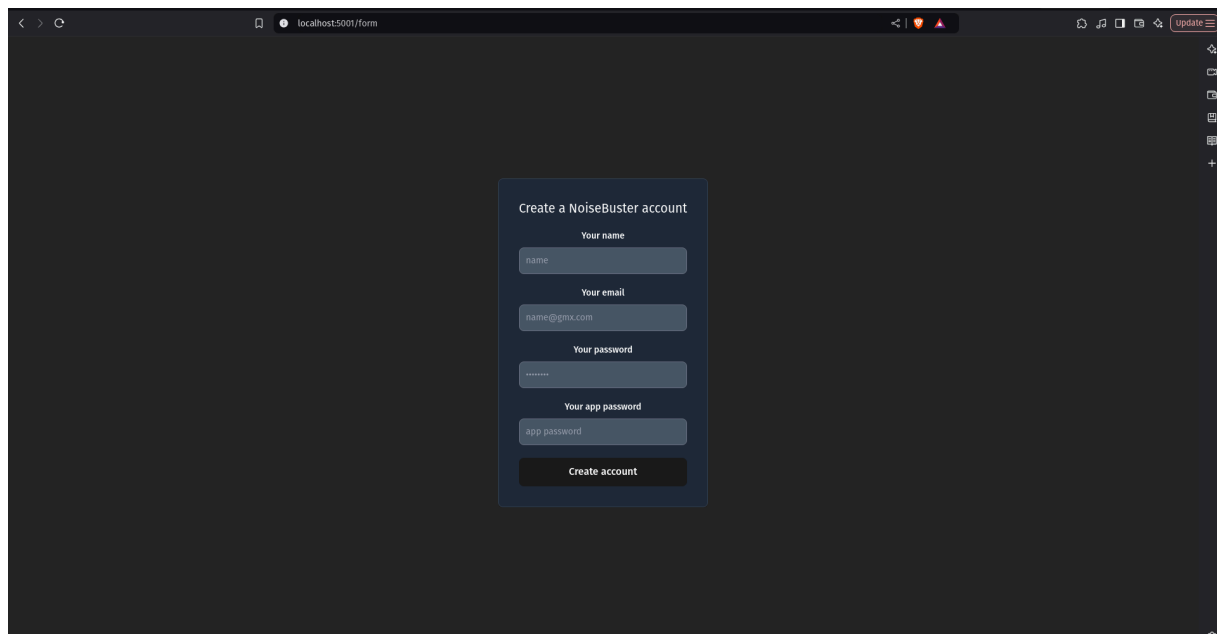
A screenshot of a web browser displaying a sign-up form titled "Create a NoiseBuster account". The form is centered on a dark background. It contains five input fields: "Your name" (placeholder: name), "Your email" (placeholder: name@gmail.com), "Your password" (placeholder: password), "Your app password" (placeholder: app password), and a "Create account" button at the bottom. The browser's address bar shows "localhost:5001/form".

Fig 1. Pagina de sign up a platformei

Pe pagina de log in utilizatorul va fi nevoit să își introducă email-ul și parola, iar ulterior dacă credențialele sunt corelate cu cele din baza de date acesta va fi dus la o altă rută unde va avea

opțiunea de a activa sau de a dezactiva acest filtru. În cazul în care credențialele nu sunt corecte o eroare va fi afișată de către browser-ul web.

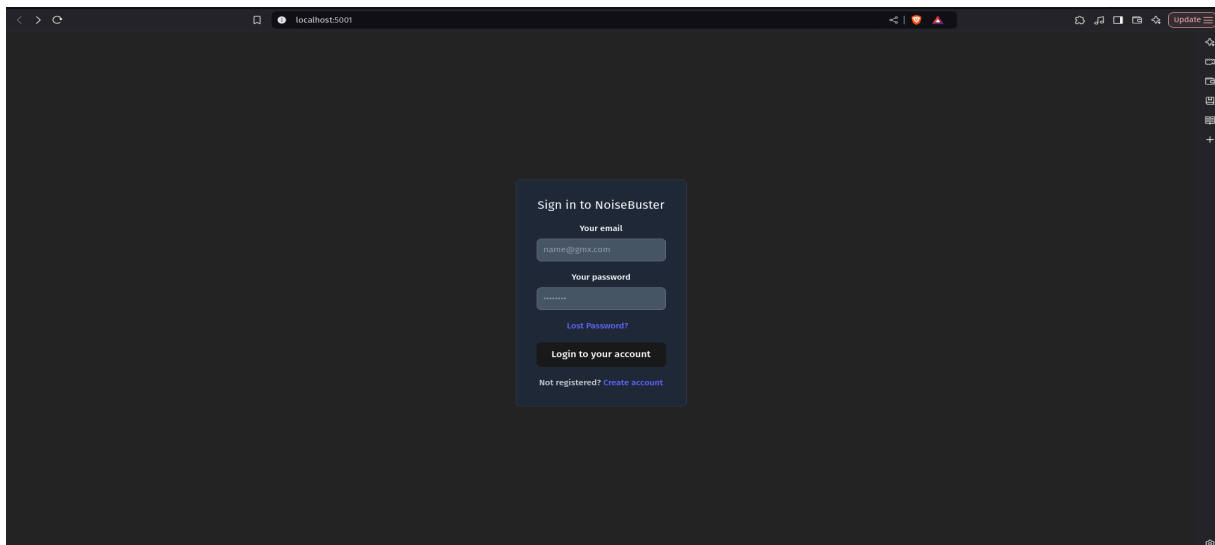


Fig 2. Pagina de log in a platformei

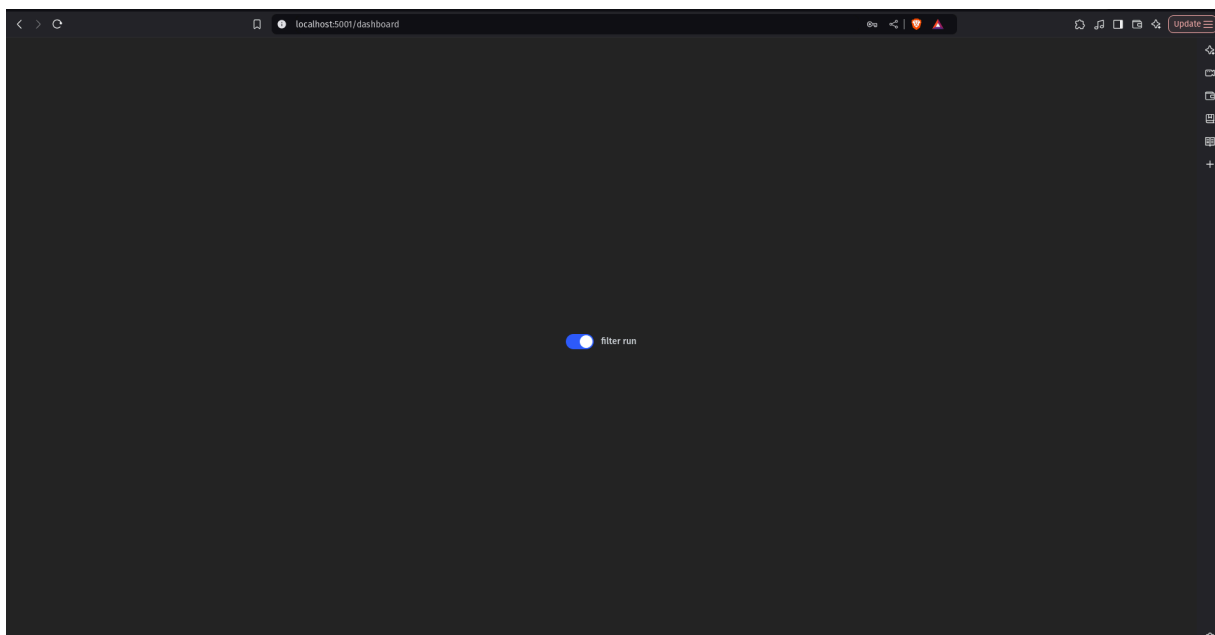
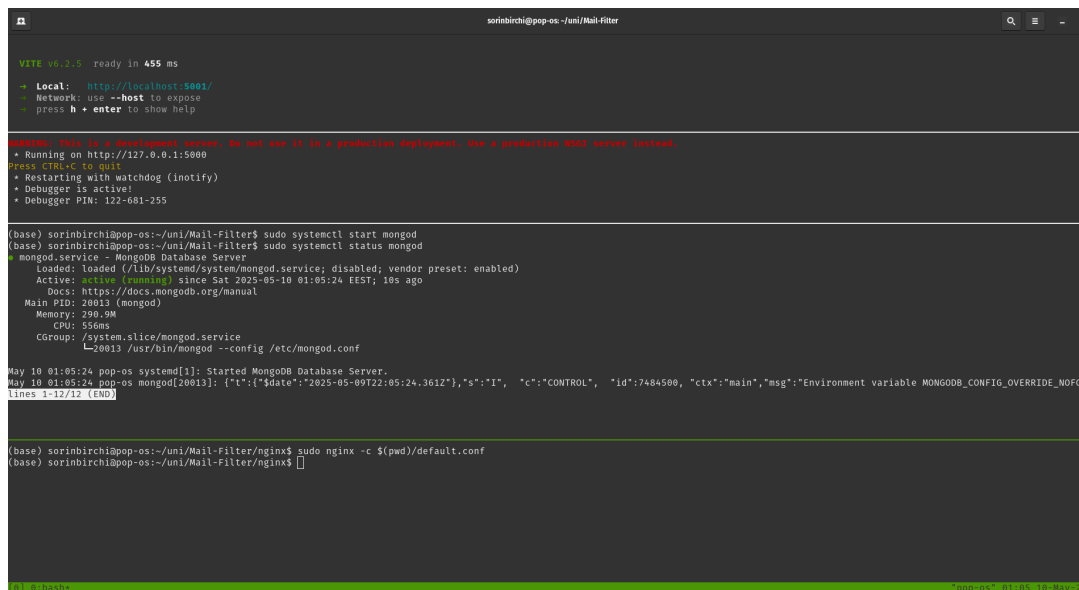


Fig 3. Pagina pentru activare/dezactivare filtru

3.4. Interactiunea dintre Front end si Back end

Interactiunea dintre front end si back end este facuta cu ajutorul modulului react-router-dom din React, care creeaza local, la rulara server-ului de front end un dom cu toate rutele site-ului, un alt rol important in comunicarea dintre front end si back end fiind jucat de Nginx.

Aplicatia are doua servere care trebuie rulate in paralel, server-ul de front end care este realizat cu Vite si server-ul de back end care este realizat cu modulul Flask din python. Nginx face legatura dintre cele doua servere care ruleaza pe porturi diferite si trimite cereri ca un reverse proxy de la front end la back end(care apoi rezolva cererile si tot prin intermediul Nginx le trimite inapoi catre front end). Am adoptat aceasta metoda in primul rand pentru securitatea sporita pe care o ofera inca de la inceput Flask-ul si totodata pentru construirea intregului back end intr-un singur limbaj, python avand aplicabilitate in toate lucrurile pe care mi le-am propus la inceput.



```
sorinbirci@pop-os: ~/uni/Mail-Filter
VITE v6.2.5 ready in 455 ms
  Local: http://localhost:5001/
  Network: use --host to expose
  press h + enter to show help

WARNING: This is a development server. Do not use it in a production deployment. Use a production Web Server instead.
  • Running on http://127.0.0.1:5000
  Press CTRL-C to quit
  • Restarting with watchdog (inotify)
  • Debugger is active!
  • Debugger PIN: 122-681-255

(base) sorinbirci@pop-os:~/uni/Mail-Filter$ sudo systemctl start mongod
(base) sorinbirci@pop-os:~/uni/Mail-Filter$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-05-10 01:05:24 EEST; 10s ago
     Docs: https://docs.mongodb.org/manual
   Main PID: 20013 (mongod)
    Memory: 290.9M
       CPU: 550ms
   CGroup: /system.slice/mongod.service
           └─20013 /usr/bin/mongod --config /etc/mongod.conf

May 10 01:05:24 pop-os systemd[1]: Started MongoDB Database Server.
May 10 01:05:24 pop-os mongod[20013]: {"t":{"$date":"2025-05-09T22:05:24.361Z"},"s":"I",  "c":"CONTROL",  "id":7484500, "ctx":"main","msg":"Environment variable MONGODB_CONFIG_OVERRIDE_NOFO
lines 1-12/12 (END)

(base) sorinbirci@pop-os:~/uni/Mail-Filter/nginx$ sudo nginx -c $(pwd)/default.conf
(base) sorinbirci@pop-os:~/uni/Mail-Filter/nginx$
```

Fig 4. Interactiunea dintre Vite, Flask, Nginx si pornirea MongoDB

3.4.1 Conectarea la server-ul de mail

În conectarea către serverul de mail, am utilizat protocolul IMAP (Internet Message Access Protocol) prin modulul **imaplib** din Python. Pentru acest lucru, am încercat o mulțime de adrese de e-mail, cum ar fi Proton, Gmail, Yahoo, Outlook și GMX. În momentul de față, aplicația utilizează doar e-mailuri de tip GMX, ulterior putând fi adăugate și celelalte căsuțe de mail. Singurul lucru necesar, în momentul de față, pentru a ne putea conecta la un alt server în afara de GMX, ar fi parola de aplicație a contului respectiv și alegerea corectă a serverului IMAP din documentația modulului.

După conectarea la căsuța respectivă de mail, funcția ce realizează conectarea ne va întoarce un obiect de tip **IMAP4_SSL**, pe care îl putem folosi pentru a naviga prin căsuța de mail a utilizatorului și pentru a modifica e-mailurile, după ce le clasificăm cu modelele de inteligență artificială.

3.5. Clasificarea textului utilizand LLM-uri

Modul de utilizare al Large Language Models este unul complex, ce necesită foarte multe date pentru a obține o antrenare cu rezultate bune. Am ales să folosesc aceste LLM-uri pentru a putea avea un model care să fie „conștient” de contextul e-mailului pe care îl va analiza, astfel încât, chiar dacă un e-mail nu ar conține explicit un cuvânt obscen, dar ar transmite o amenințare prin mesajul exprimat, modelul să recunoască acest aspect și să îl clasifice ca atare.

Modul în care un LLM vede o propoziție și învață pe baza ei este următorul:

- În prima fază, la primirea de input (text în limbaj natural), modelul nostru nu ar putea face nimic cu acest input neprelucrat, așa că realizează o tokenizare. Această tokenizare constă în atribuirea unei liste de numere reale fiecărui cuvânt din propoziție, astfel creând mai multe tokene. Așadar, un token reprezintă lista de numere reale ce definește cuvântul respectiv.
- Acum că LLM-ul nostru are aceste tokene, va putea să plaseze, într-un spațiu vectorial, fiecare cuvânt din inputul nostru, în funcție de lista de numere reale prin care este descris.
- După stabilirea locațiilor vectorilor, urmează pasul de atenție, care va face modelul să fie „conștient” de contextul propoziției noastre.

Pentru această parte de înțelegere, am cules mai multe informații de la *3Blue1Brown* [3], care este o echipă de oameni ce creează videoclipuri, cât și postări legate de domeniul inteligenței artificiale și nu numai.

Ideea inițială pentru acest task a fost să utilizez tehnicile de transfer learning și să fac fine-tune pe un model de la llama. Modelul, fiind deja preantrenat cu destul de multe date, putea genera bucăți de text consistente. Totuși, task-ul pe care l-am adoptat presupunea ca

modelul să primească conținutul e-mailului și să returneze un singur cuvânt care să reflecte un label.

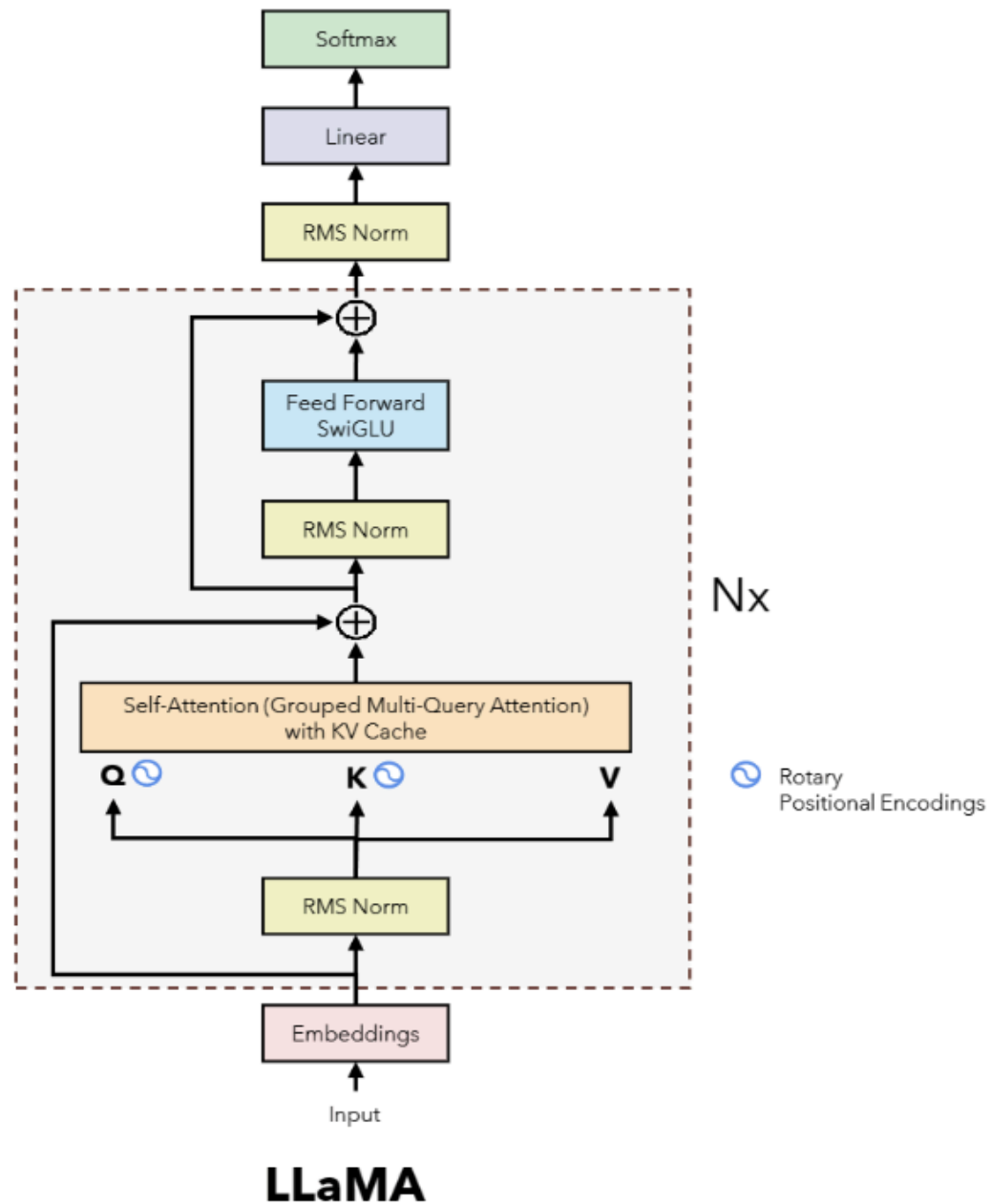


Fig 5. Arhitectura modelului llama

Fine tuning-ul a fost făcut utilizând framework-ul unsloth [4] din Python ce reușește să accelereze procesul de transfer learning pe plăcile de bază nvidia pentru ca timpul de reantrenare să fie mult mai mic față de fine tuning-ul tradițional.

Baza de date utilizată pentru acest proiect este numită ro-offense și se găsește pe hugging face la link-ul următor [5].

După încercarea de a face fine tune pe acest model am observat faptul că nu avem destule date astfel încât să putem realiza acest procedeu într-un mod eficient, acest lucru rezultând în răspunsuri fără sens ale modelului nostru.

Cum am încercat să soluționez această problemă?:

- Adăugarea de date artificiale și crearea unei noi baze de date cu aproximativ 50.000 de prompt-uri pentru LLM. Ideea a fost destul de bună, astfel încât acuratețea modelului a crescut într-un anumit procent, dar tot rămânând foarte slabă. Din cauza output-urilor limitate pe care încercam să i le arătăm modelului, asta aducea de la sine nevoia unei baze de date și mai mari, întrucât matricea de atenție nu avea valori destul de bune pentru a determina legătura între token-uri.
- Rezolvarea la care am ajuns a fost rularea locală a unui container de docker cu ollama, folosind modelul Phi-4 al celor de la *Microsoft* [6]. Prin crearea unui prompt bine stabilit care a fost folosit la generarea fiecărui răspuns pentru mail-uri, modelul a performat așa cum ar fi trebuit, alegând label-urile corecte.

3.6. Clasificarea fișierelor audio

Pentru clasificarea fișierelor audio am ales să mă folosesc de modul de clasificare al textelor din email-uri și atașamente. Convertirea inițială a fișierelor audio în fișiere text a fost realizată cu ajutorul modelului *whisper de la OpenAI* [7].

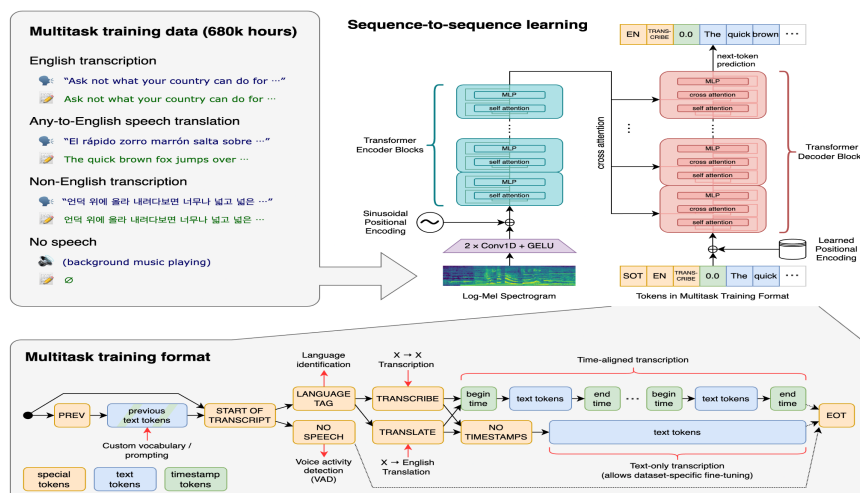


Fig 6. Arhitectura modelului whisper

Whisper funcționează în următorul mod:

- Semnalul audio este convertit într-o spectrogramă de tip Log-Mel
- Encoder-ul procesează prin transformare spectrograma
- Decoder-ul generează textul token cu token din datele din encoder
- Prin modelul de multitask training se fac mai multe acțiuni în paralel, aceste acțiuni fiind definite de token-urile de input care sunt de trei tipuri: special tokens, text tokens și timestamp tokens

După ce fișierul audio este modificat de whisper și vom primi versiunea text a acestuia, pur și simplu îl dăm ca input la modelul Phi-4 folosind prompt-ul deja definit la task-ul de clasificare a textului.

3.7. Clasificarea fișierelor foto

Clasificarea imaginilor este un task destul de răspândit, de cele mai multe ori acesta fiind abordat cu rețele neurale convoluționale (CNN) sau cu vision transformers (ViT), care sunt mai complecși și care pot înregistra mai multe feature-uri decât CNN-urile, dar necesită o cantitate mult mai mare de date și timpul antrenării lor este unul mare.

În implementarea abordată de mine am folosit o rețea convoluțională utilizând ca și clasificator funcția de activare softmax. Pentru a nu risca să întâlnim evenimentul de overfitting și pentru a avea o generalizare mai bună, înainte de a face clasificarea efectivă am creat o rețea neurală ce extrage feature-urile importante din imagini utilizând convoluția. Ulterior după ce avem destule feature-uri despre imaginile date putem utiliza clasificatorul pentru a vedea cărei clase face parte imaginea curentă

Acest task poate fi văzut și ca o problemă de detecție a anomaliilor întrucât baza de date folosită [8] are mult mai multe poze normale decât orice alt tip de poze. Încă o observație importantă este faptul că orice tip de imagine poate fi clasificat prin acest model, indiferent de extensia acesteia.

3.8. Clasificarea fișierelor video

Clasificarea fișierelor video reprezintă îmbinarea clasificării foto și celei text, întrucât fișierul video pe care îl vom primi ca și input va trebui descompus în fișier audio și mai multe poze. Fișierul audio va fi convertit în fișier text, iar aici ne vom folosi de LLM. Pentru a face ca programul să ruleze mai rapid am ales să convertim fiecare secunda din video într-o poză. Ulterior, verificarea pozelor se va face paralelizat utilizând CNN-ul.

4.0. Prezentarea aplicației

Aplicația a fost realizată utilizând mai multe framework-uri populare din Python, câteva modele de AI ale companiilor lider pe piață, tehnologii web cum ar fi HTML, Tailwind CSS și React, MongoDB, Nginx, Docker și Kubernetes.

4.1. Framework-uri utilizate

- **Numpy** - pentru calcule numerice
- **Unsloth** - pentru accelerarea fine tuning-ului LLM-urilor
- **PyTorch** - pentru antrenarea CNN-ului
- **Keras** - pentru funcțiile matematice utilizate pentru ML
- **Whisper** - pentru convertirea din speech to text
- **Phi-4** - pentru clasificarea text
- **LLaMA-3** - pentru clasificarea text
- **Transformers** - clasa din hugging face pentru transformare
- **Datasets** - clasa din hugging face pentru a încărca baze de date de pe site
- **Tailwind CSS** - framework peste CSS
- **React** - framework peste React
- **MongoDB** - baza de date distribuită
- **Vite** - tool pentru crearea și managerierea serverelor web
- **Nginx** - tool pentru reverse proxy
- **Docker** - pentru containerizare
- **Kubernetes** - pentru orchestrarea containerelor
-

4.2. Descriere a modului de funcționare

1. Se realizează logarea/înregistrarea și pornirea filtrului de către utilizator
2. Programul verifică în timp real mail-urile primite în căsuța sa de mail din momentul în care filtrul este activ pentru mail-ul respectiv

3. Pentru fiecare mail în parte începe verificarea în următoarea ordine: text, fișier audio, poze, video. În cazul în care am găsit că mail-ul respectiv este tulburător acesta va fi șters și procesul va continua. În cazul în care mail-ul trece de filtre fără a fi șters, procesul de verificare continuă cu celelalte mail-uri care vor fi primite.

5. Concluzii

Componentele descrise sunt funcționale și reușesc să comunice unele cu altele astfel încât să poată aplica filtrele respective pe mail-uri și să facă acest lucru pentru limba romana. Cu toate acestea, din cauza hardware-ului procesul de clasificare este destul de încet și în unele cazuri apar și răspunsurile fals pozitive ale modelului. Din observațiile făcute aceste fals positive apar de obicei în texte care sunt greu de descifrat pentru LLM din cauza greșelilor de scriere sau a anumitor regionalisme. Totodată un alt lucru ce influențează apariția fals pozitivelor este și numărul mic de cadre pe secundă din video-uri pentru că având doar 1 imagine per secundă se pierde multă informație.

Schimbări pe viitor:

- O primă schimbare ar fi cea legata de LLM. Consider că un LLM construit și antrenat exclusiv pe seturi în limba romana ar da rezultate mai bune și mai apropiate de cele reale.
- Adunarea mai multor texte în limba romană conținând neologisme, arhaisme și regionalisme ar putea crea o bază de date solidă pentru analiza limbajului în limba romană.
- Creșterea numărului de poze ce sunt extrase din video ar crește acuratețea clasificării.
- Utilizarea unui ViT ar fi o idee mai bună pentru că ar avea o acuratețe și mai mare și totodată ar putea fi antrenat utilizând unsloth, ceea ce l-ar face să poată fi și mai ușor de antrenat și totodată mai rapid decât CNN-ul actual.

6. Bibliografie

- [1]. Purgomalum profanity filter Link: <https://www.purgomalum.com/>
- [2]. Microsoft Flow
Link: <https://www.microsoft.com/en-us/power-platform/products/power-automate>
- [3]. 3Blue1Brown Link: <https://www.3blue1brown.com/>
- [4]. Unsloth Link: <https://docs.unsloth.ai/>
- [5]. ro-offense Link: <https://huggingface.co/datasets/readerbench/ro-offense>
- [6]. Marah Abdin Jyoti Aneja Harkirat Behl S'ebastien Bubeck Ronen Eldan Suriya Gunasekar Michael Harrison Russell J. Hewett Mojan Javaheripi Piero Kauffmann James R. Lee Yin Tat Lee Yanzhi Li Weishung Liu Caio C. T. Mendes Anh Nguyen Eric Price Gustavo de Rosa Olli Saarikivi Adil Salim Shital Shah Xin Wang Rachel Ward Yue Wu Dingli Yu Cyril Zhang Yi Zhang, "Phi-4 Technical Report"
Link: <https://www.microsoft.com/en-us/research/wp-content/uploads/2024/12/P4TechReport.pdf>
- [7]. OpenAi, Whisper Link: <https://github.com/openai/whisper>
- [8]. Sanskar Hasija, UCF Crime Dataset
Link: <https://www.kaggle.com/datasets/odins0n/ucf-crime-dataset>
- [9]. Andrew Ng, Machine Learning Specialization
Link: <https://www.coursera.org/specializations/machine-learning-introduction/>