

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ROMÂNĂ

LUCRARE DE LICENȚĂ

**O abordare euristică pentru detectarea
știrilor false bazată pe tehnici de
instruire automată**

Conducător științific
Lect. Dr. Ioan-Gabriel MIRCEA

Absolvent
Răzvan-Nicolae ȚÎMBOIU

2021

ABSTRACT

The world keeps changing. As technology evolves, it transforms the way we communicate. Everything happens online right now. New data is generated every day, increasing the volume of information shared on the web. This leads to a great question... How can we filter out deceiving content from the information we consume?

This paper aims to contribute to the creation of a partially or fully automated alternative for the verification of information distributed on social networks in Romania, mainly focusing on detecting news with misleading content.

Throughout the first chapter, we described why fake news represents a problem for Romanian society. The second chapter introduces some NLP concepts so that the reader is familiarized with concepts that will be used in future chapters. The third chapter presents related work and a quick overview of the situation we are currently in. The fourth chapter introduces our proposed approach for detecting fake news and describes the whole data gathering process, along with training details for the chosen models. The fifth chapter presents an app that is built based on our models, to make it easier to verify news and gather more data. The sixth chapter presents the results we obtained with various models, and the last chapter concludes the paper and paves the way for future work.

Our main contributions are as follows:

- **We collected over 50,000 news articles that we cleaned, labeled, and used to train five classification models.**
- **We have introduced a new class for satire articles.**
- **We tried different architectures for the proposed models, evaluated their performance, and discussed the results produced by each architecture.**
- **We have designed and implemented an application that facilitates the process of collecting and labeling news articles so that in the future we can build larger and better quality data sets.**
- **We introduced a CNN-LSTM hybrid model. It gets much better results than models based on LSTM or GRU and trains faster.**

Results show that CNN models have the best performance when classifying based on the content of the article and GRU models perform best when classifying based on the headline of an article.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Cuprins

1	Introducere	1
1.1	Prezentarea problemei	2
1.2	Motivație și relevanță	3
2	Procesarea de Limbaj Natural	5
2.1	Cum funcționează ?	6
2.2	Idei care au schimbat PLN de-a lungul timpului	7
2.2.1	Învățarea automată și extragerea de caracteristici	7
2.2.2	Învățarea profundă	10
2.2.3	Modele complexe	16
3	Domeniul problemei	17
3.1	Abordări înrudite	17
3.2	Analiza situației curente	21
4	Soluția propusă - Prezentare generală	22
4.1	Construirea setului de date	23
4.2	Alegerea modelului potrivit	26
4.2.1	Modele propuse	26
4.2.2	Experimente	27
4.3	Integrarea într-un API	31
5	Aplicația propusă	35
5.1	Arhitectura aplicației	35
5.1.1	Comportament	35
5.1.2	Structură	36
5.1.3	Legătura dintre comportament și structură	37
5.2	Implementarea aplicației	43
5.2.1	Diagrama de clase	43
5.2.2	Baza de date	44
5.2.3	Tehnologiile folosite	45

5.3	Manual de utilizare	46
6	Rezultate obținute	51
7	Concluzii și dezvoltări ulterioare	55
	Bibliografie	57

Capitolul 1

Introducere

Scopul acestei lucrări este de a contribui la crearea unei alternative parțial sau complet automatizate pentru verificarea informațiilor distribuite pe rețelele de socializare din România, cu preponderență axându-ne pe detectarea de știri cu conținut înșelător. *Contribuțiile personale se regăsesc la finalul lucrării.*

În continuarea acestui capitol am descris problema reprezentată de știrile false și motivația pentru care am ales acest subiect ca temă a lucrării.

În *capitolul 2*, cititorul va fi introdus în domeniul procesării de limbaj natural pentru a-l ajuta să înțeleagă conceptele prezentate în capitolele ulterioare. Tot aici am descris și câteva idei care au schimbat acest domeniu de-a lungul timpului.

În *capitolul 3* am prezentat diferite abordări propuse de alți cercetători pentru a detecta știrile false. De asemenea, am descris situația curentă și am propus o abordare proprie pentru detectarea de știri false. Această abordare presupune crearea unui sistem care ar putea deosebi știrile false de știri reale sau de satiră.

În *capitolul 4* am propus cinci modele iar apoi am descris în detaliu modul de colectare și preprocesare a setului de date folosit la antrenarea acestor modele. Pe lângă asta, am descris modul de funcționare al unui API unde a fost integrat modelul cu cele mai bune rezultate.

În *capitolul 5* am descris proiectarea și implementarea unei aplicații ce folosește API-ul prezentat anterior. Aplicația are ca scop ajutarea unui utilizator să verifice conținutul unui articol de știri. Aceasta oferă și funcționalități pentru a colecta articole de știri trimise de utilizatori. Acest lucru permite ca pe viitor să putem dezvolta modele cu performanțe mai bune, având mai multe date.

În *capitolul 6* am prezentat rezultatele obținute în urma experimentelor alături de o comparație a modelelor pe care am experimentat. Modelele bazate pe CNN obțin cele mai bune rezultate pentru clasificare pe baza articolului complet iar modelele bazate pe GRU obțin cele mai bune rezultate în cazul clasificării doar pe baza prezentării articolului (vezi fig. 4.3).

În *capitolul 7* se regăsesc concluziile lucrării și idei pentru dezvoltări ulterioare.

1.1 Prezentarea problemei

Lumea este într-o continuă evoluție. Avansurile tehnologice din ultimii ani au dus la schimbări majore în modul în care comunicăm. Zilnic sunt generate date noi, lucru ce duce la o creștere constantă a volumului de informație din mediul online.

Odată cu apariția platformelor de social media s-a facilitat accesul la informații. Zeci de milioane de utilizatori noi apar în fiecare an pe rețelele de socializare [17], aceștia crescând și mai rapid producția de date. În [20] este prezentat faptul că în fiecare minut:

- Utilizatorii **Snapchat** distribuie peste 527.760 poze.
- Peste 120 de liber profesioniști își crează un cont pe **LinkedIn**.
- În jur de 4.146.600 clipuri video sunt vizualizate pe platforma **Youtube**.
- 456.000 de tweet-uri sunt trimise pe **Twitter**.
- Utilizatorii **Instagram** postează 46.740 poze.

Având acces la un volum așa mare de date în orice moment, printre întrebările care ne trec prin minte sunt următoarele; Cum putem filtra conținutul negativ din informația pe care o consumăm și cum ne putem asigura că sursa informațiilor consumate este una de încredere.

Prin **conținut negativ** se înțeleg diferite tipuri de conținut cu scopul de a dezinforma precum sunt știrile false, postările de tip clickbait, propaganda politică ș.a.m.d. Distribuirea unui astfel de conținut are ca scop crearea de panică în rândul oamenilor, punerea într-o lumină proastă a unei persoane publice sau a unui grup de persoane, instigarea la revolte și violență sau alte astfel de urmări nefavorabile.

Problema cu care ne confruntăm o reprezintă distribuirea de informații false în mediul online. Pentru a demonstra importanța soluționării acestei probleme voi prezenta câteva efecte negative rezultate în urma consumului de informații derutante.

- În [6] este prezentat faptul că peste 800 de decese au avut loc din cauza știrilor false legate de noul coronavirus. Tot în acest reportaj se spune că au avut loc peste 5800 de spitalizări din aceleași motive. O mare parte din decese au avut loc în urma consumului de alcool metilic care ar fi fost propus ca tratament pentru Covid-19. Zvonuri distribuite online au dus la atacuri ale mafiei în India și otrăviri în masă în Iran.
- Tot în [6] scrie că inginerii de telecomunicații au fost amenințați și atacați, iar antenele de telefonie au fost aprinse în Marea Britanie și în alte țări din cauza teoriilor conspiraționiste care au fost incubate și amplificate online.

- În [30] sunt descrise rezultatele unui sondaj din care reiese că în jur de 28% din americani cred că Bill Gates dorește să folosească vaccinurile împotriva Covid-19 pentru a implanta cipuri în oameni iar în jur de 43% cred că virusul este produsul unor laboratoare din China de unde acesta ar fi evadat.

Chiar dacă exemplele prezentate sunt din SUA, nici în România situația nu stă mult mai bine [8]. Cele mai afectate persoane sunt vârstnicii care cad ușor pradă teoriilor conspiraționiste și informațiilor false deoarece de multe ori e mai ușor să înțelegi informația falsă prezentată într-un mod mai colorat, decât echivalentul adevărat al acesteia.

În rândul tinerilor situația stă puțin mai bine. În [11] ne este prezentat faptul că tinerii de vârsta liceului au de-a face cu știrile false săptămânal. Cu toate astea, peste trei sferturi dintre ei le pot distinge cu ușurință de știrile cu informații reale. Acest lucru ne arată că nu e pierdută încă lupta împotriva dezinformării.

Guvernul se implică și el activ în lupta împotriva dezinformării, lucru ce denotă gravitatea situației și necesitatea rezolvării acestei probleme. În țară se încearcă combaterea știrilor false prin campanii de conștientizare derulate de Serviciul Român de Informații și prin platforma Inforadar¹ (un portal de știri propus de Ministerul Apărării Naționale care are ca scop descoperirea de știri false și eliminarea surselor care le propagă în mediul online românesc).

Pe lângă cele menționate, în țară există și o rețea de fact-checking² unde un grup de oameni verifică minuțios afirmațiile unor oameni politici. O astfel de abordare nu e plauzibilă deoarece e aproape imposibil să prelucrezi toate știrile scrise zilnic fără a consuma extrem de multe resurse.

În urma celor prezentate, putem susține cu tărie că dezinformarea din mediul online reprezintă o problemă serioasă, nu doar la nivel global, ci și local, la nivel de țară. Trebuie deci să încercăm să găsim o rezolvare plauzibilă pentru această problemă sau macar un mod în care îi putem ameliora efectele negative deoarece numărul de victime afectate de știri și informații false crește de la o zi la alta.

1.2 Motivație și relevanță

Am ales această temă pentru lucrarea de licență deoarece este un subiect de actualitate care mă afectează personal și afectează în același timp întreaga societate. Prin intermediul acestei lucrări doresc să cresc nivelul de conștientizare legat de problema informațiilor false distribuite în mediul online.

Un alt factor care m-a motivat să aleg acest subiect este popularitatea acestuia în literatura de specialitate. Conform graficului prezentat în Fig. 1.1 pe parcursul

¹<https://inforadar.mapn.ro/>

²<https://factual.ro/>

ultimilor cinci ani a crescut semnificativ numărul de publicații legate de *fake news detection* (detectare de știri false) așadar am zis să contribui și eu la acest val de schimbări.

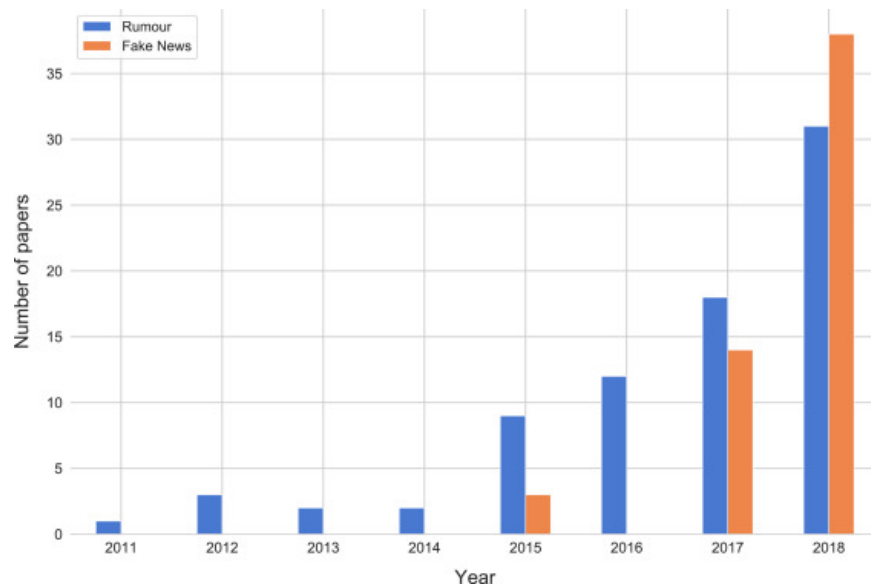


Figura 1.1: Număr de lucrări publicate pe anumite subiecte. [6]

Pe lângă motivele personale, ar mai fi și motivele etice pentru care ar trebui să luptăm împotriva dezinformării. Acestea sunt cum urmează [7]:

- *Fiecare dintre noi merită să știe adevărul* - Oricine ar trebui să fie capabil să ia o hotărâre singur atâta timp cât ar avea în față fapte și informații reale.
- *Informațiile false distrug credibilitatea surselor* - Dacă argumentele cuiva se bazează pe informații false, va fi mult mai dificil pentru oameni să considere opinia acelei persoane în viitor.
- *Informațiile false pot dăuna sănătății* - Furnizorii de sfaturi medicale false, cum ar fi *Mercola.com* și *NaturalNews.com* (sau pe plan local *TuBio.ro*), ajută la perpetuarea miturilor precum acela că HIV și SIDA nu sunt legate, sau că vaccinurile provoacă autism. Aceste site-uri sunt foarte vizitate, iar informațiile medicale false prezentate de acestea pot duce la probleme grave de sănătate sau chiar deces.

Nu în ultimul rând, propria fascinație legată de sistemele bazate pe procesare de limbaj natural m-a motivat să aprofundez acest domeniu și să înțeleg modul de lucru al unor astfel de sisteme cu scop final de a dezvolta propriul produs pentru combaterea dezinformării.

Capitolul 2

Procesarea de Limbaj Natural

Procesarea de Limbaj Natural este o știință interdisciplinară ce combină Inteligența Artificială (AI) și lingvistica computațională astfel încât computerele și oamenii să poată relaționa fără probleme.

Câteva exemple de aplicații care folosesc PLN sunt :

- **Sistemele de corectare a mesajelor** (Autocorrect) și **sistemele cu sugestii de completare** (Autocomplete) frecvent întâlnite în tastaturile integrate în majoritatea telefoanelor mai noi.
- **Serviciul de traducere Google Translate** prin intermediul căruia poți traduce din și în peste o sută de limbi cu precizie din ce în ce mai bună.
- **Asistenții precum Siri, Cortana sau Asistentul Google**, fiecare dintre ei dispunând de module pentru înțelegerea vorbirii (Speech-to-Text) și sintetizarea vorbirii (Text-to-Speech).
- **Sistemele pentru filtrarea mailurilor** în spam sau non-spam incorporate în cadrul platformelor cu servicii de poștă electronică.

Pentru a discuta cu oamenii, un program trebuie să înțeleagă sintaxa (gramatica), semantica (sensul cuvântului) și morfologia. Numărul de reguli de urmărit poate părea copleșitor și explică de ce încercările anterioare din domeniul NLP au dus inițial la rezultate dezamăgitoare.

Cu toate acestea domeniul NLP a evoluat încet încet trecând de la modele bazate pe reguli la modele bazate pe **pattern recognition** (învățarea tiparelor). Acest lucru a permis calculatoarelor să înțeleagă conținut nestructurat, utilizând AI și învățarea automată pentru a face deducții și a da context limbajului, la fel ca și creierul uman.

În ultimii ani, arhitecturile și algoritmi din *Deep Learning* (învățare în profunzime) au făcut progrese impresionante în domenii precum recunoașterea imaginilor și procesarea vorbirii. Aplicarea lor în PLN a fost mai puțin impresionantă la

început, dar ulterior au dovedit că pot aduce contribuții semnificative, oferind rezultate *de ultimă generație* (en: **SOTA** - State Of The Art) pentru unele sarcini din domeniu.

Recunoașterea de entități, etichetarea unor părți din vorbire sau analiza sentimentelor sunt câteva dintre problemele în care modelele cu rețele neuronale au depășit abordările tradiționale bazate pe reguli. Progresul în traducerea automată este probabil cel mai remarcabil dintre toate.

2.1 Cum funcționează ?

Un sistem pentru PLN este compus din mai multe module înlănțuite în cadrul unui *pipeline* (conductă). Datele trec în ordine prin fiecare modul urmând ca la ieșirea din sistem să avem un răspuns legat de problema pe care o rezolvă sistemul. Un pipeline ar putea avea următoarele module:

- **Modul Preprocesare:** Textul este curățat și pregătit pentru modulul următor.
- **Modul Extragere Caracteristici:** Se extrag caracteristici din text care apoi sunt trimise mai departe.
- **Modul Clasificare:** Pe baza caracteristicilor, clasifică textele primite în funcție de sarcina asignată sistemului.

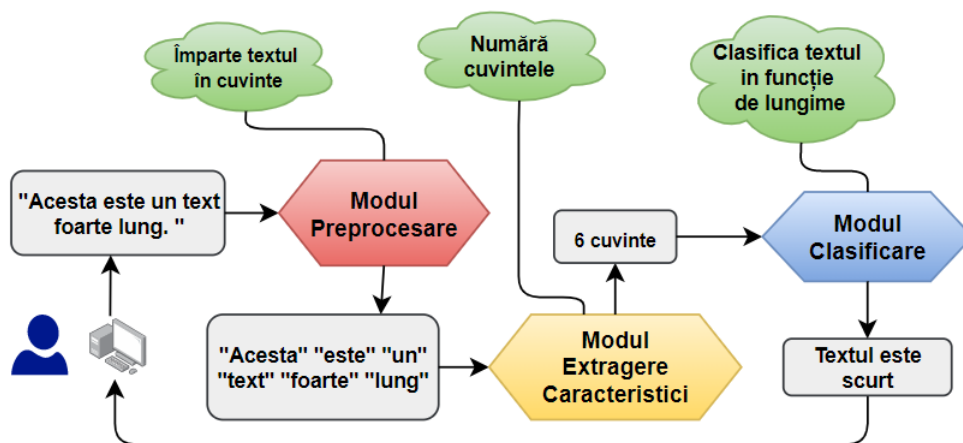


Figura 2.1: Clasificare de text în funcție de lungime.

În Fig 2.1 se poate observa o schiță ce reprezintă un *workflow* (mod de lucru) pentru un sistem bazat pe PLN care este folosit pentru clasificare de text pe bază de lungime. Modulul de preprocesare curăță și împarte textul, cel de extragere produce caracteristici (numărul de cuvinte în acest caz) iar cel de clasificare clasifică textul pe baza caracteristicilor primite de la modulul de extragere.

2.2 Idei care au schimbat PLN de-a lungul timpului

În cadrul acestui subcapitol voi prezenta cateva idei care au schimbat domeniul PLN de-a lungul timpului și care ne-ar putea fi de folos în dezvoltarea unui sistem pentru clasificarea de știri. Chiar dacă modelele prezentate au și alte cazuri de utilizare, mă voi concentra doar pe cele relevante pentru subiectului lucrării.

2.2.1 Învățarea automată și extragerea de caracteristici

Învățarea automată este o ramură a inteligenței artificiale și a informaticii care se concentrează pe utilizarea datelor și algoritmilor pentru a imita modul în care oamenii învață, îmbunătățind treptat calitatea rezultatelor.

Învățarea automată clasică sau "non-profundă" depinde mai mult de intervenția umană pentru a învăța. Experții umani determină setul de caracteristici pentru a înțelege diferențele dintre datele introduse, necesitând de obicei date mai structurate pentru a învăța. După extragerea caracteristicilor, acestea sunt folosite pentru antrenarea unor modele de clasificare.

Cateva astfel de modele de clasificare sunt:

- **Regresia Logistică**
- **Mașina cu suport vectorial**
- **Arborii de decizie și Naive Bayes**

Pentru extragerea de caracteristici din documente text, de-a lungul timpului s-au folosit mai multe metode. Dintre cele foarte cunoscute fac parte și următoarele:

Bag of Words

Unul dintre cele mai simple tipuri de modele de extragere a caracteristicilor se numește *Bag of Words* (sac de cuvinte). Numele Bag of Words se referă la faptul că acest model nu ține cont de ordinea cuvintelor. Ne putem imagina că fiecare cuvânt este pus într-un sac, în care ordinea cuvintelor se pierde.

Deși există câteva variații diferite ale acestui model, cea mai obișnuită metodă este să numeri pur și simplu numărul de apariții a fiecărui cuvânt în cadrul unui document și să reții într-un vector acel număr (numit în continuare *vector de numărare*). În acest fel, frecvențele termenilor rămân intacte, deși gramatica și ordinea se pierd.

O altă abordare este de a avea în schimb un vector binar care ține evidența dacă un cuvânt există sau nu în cadrul unui document. Un dezavantaj pentru această abordare este faptul că pierde inclusiv multiplicitatea cuvintelor, pe lângă ordine și gramatică.

TF-IDF

O metodă care s-a dovedit a fi simplă și eficientă pentru extragerea caracteristicilor este *Term Frequency-Inverse Document Frequency*. TF-IDF este o tehnică ce poate fi utilizată pentru a determina relevanța termenilor în documente. Prin folosirea acestei tehnici, putem determina ce termeni dintr-un document sunt cei mai reprezentativi pentru acel document.

Această metodă poate fi privită ca o formă de Bag of Words deoarece nu ia în considerare gramatica sau ordinea. Metoda constă în trei pași:

1. **Calculăm frecvența termenilor.** (TF - Term Frequency)
2. **Calculăm frecvența inversă a documentului.** (IDF - Inverse Document Frequency)
3. **Înmulțim TF și IDF pentru a obține relevanța pentru un termen.**

O variantă a TF funcționează în primul rând calculând de câte ori apare un termen într-un document, la fel ca și în cazul vectorului de numărare. Raționamentul aici este că cuvintele care apar frecvent într-un document sunt probabil mai importante decât cuvintele ce nu apar frecvent. Rezultatul este apoi normalizat prin împărțirea la numărul de cuvinte din întregul document.

$$TF(\alpha, \beta) = \frac{\text{Frecvența Cuvantului } \alpha \text{ în Documentul } \beta}{\text{Numărul de Cuvinte din Documentul } \beta} \quad (2.1)$$

Această normalizare se face pentru a preveni o interferență produsă de documente mai lungi, obținând frecvența la care apare termenul și nu doar numărul total de apariții ale acestuia.

Pentru a calcula partea IDF a formulei, o variantă este să luăm numărul total de documente și să îl împărțim la numărul de documente în care apare termenul. Rezultatul va fi apoi logaritmicizat. Partea IDF a formulei acționează ca o pondere, dând mai multă importanță termenilor relevanți și mai puțină celor care nu contează așa mult.

$$IDF(\alpha) = \log_2 \frac{\text{Numărul de Documente}}{\text{Numărul de Documente care Contin } \alpha} \quad (2.2)$$

În final avem:

$$\text{Relevanța Pentru Cuvantul } \alpha \text{ în Documentul } \beta = TF(\alpha, \beta) * IDF(\alpha) \quad (2.3)$$

Word Embedding

Când se utilizează modele pentru *word embedding* (încorporare de cuvinte) precum *Word2Vec* [21] [22] cuvintele sunt reprezentate în cadrul unui spațiu vectorial. În mod ideal, o metodă de încorporare reprezintă cuvintele în așa fel încât două cuvinte care sunt sinonime să aibă reprezentări vectoriale similare.

Pe lângă relațiile sintactice, pot fi păstrate și alte relații lingvistice între diferite cuvinte. De exemplu, atunci când se utilizează aceste reprezentări în spațiu vectorial pentru cuvinte, operațiile vectoriale "*Rege - Bărbat + Femeie*" dau un rezultat care este foarte asemănător cu reprezentarea din spațiul vectorial pentru cuvântul "*Regină*". (vezi Fig. 2.2).

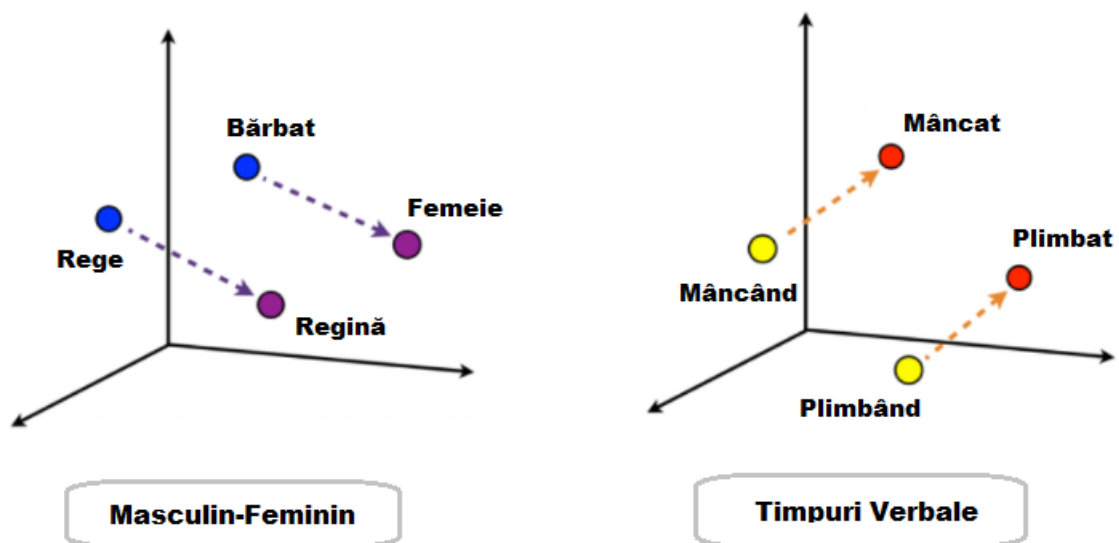


Figura 2.2: Relații reprezentate în spațiul vectorial. [3]

Un alt model de încorporare a cuvintelor utilizat în mod frecvent este *GloVe* (**G**lobal **V**ectors). *GloVe* este un model de învățare statistic nesupravegheat care se folosește de matricea de coincidență spre a genera reprezentări în spațiul vectorial pentru cuvinte. Matricea de coincidență se construiește calculând cât de des apar împreună perechi de cuvinte într-o fereastră contextuală specificată.

Până acum au fost prezentate doar modele care nu țin cont de sensul cuvintelor în momentul în care le reprezintă în spațiul vectorial. Folosind GloVe spre exemplu, cuvântul "broasca" va avea exact aceeași reprezentare și în cadrul secvenței "broasca stă pe malul lacului" dar și în "broasca ușii este stricată", chiar dacă nu se referă la aceeași entitate. Urmăreză un capitol legat de modele avansate care reușesc să diferențieze sensul cuvintelor în funcție de context.

2.2.2 Învățarea profundă

Învățarea profundă (en: *Deep Learning*) face parte dintr-o familie de metode de învățare automată bazate pe rețele neuronale artificiale.

Modul în care învățarea profundă și învățarea automată diferă este în modul în care învață fiecare algoritm. Învățarea profundă automatizează o mare parte din procesul de extragere de caracteristici, eliminând o parte din intervenția manuală. Acest lucru permite utilizarea unor seturi de date mai mari.

Învățarea automată "profundă" poate determina automat setul de caracteristici care disting diferitele categorii de date între ele. Spre deosebire de învățarea automată, nu necesită intervenția umană pentru procesarea datelor, permițându-ne să scalăm învățarea automată în moduri mai interesante.

"Profundzimea" din învățarea profundă se referă doar la profunzimea straturilor dintr-o rețea neuronală. În continuare voi descrie succint trei tipuri de rețele folosite în sisteme bazate pe învățare profundă în contextul procesării de limbaj natural.

ANN

Rețelele neuronale artificiale (en: ANN - Artificial Neural Network) sunt rețele compuse din mai multe straturi de noduri. Fiecare rețea neuronală are un strat de intrare, unul sau mai multe straturi ascunse și ultimul strat de ieșire, vezi Fig. 2.3.

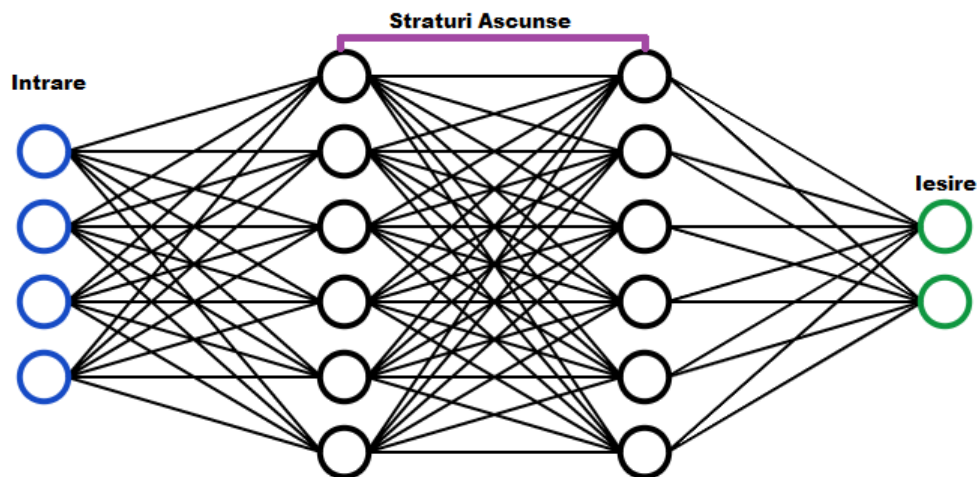


Figura 2.3: Structura unei rețele neuronale simple.

Fiecare nod, sau neuron artificial, se conectează la un altul și are o pondere și un prag asociate precum este schițat în Fig. 2.4. Dacă valoarea de la ieșirea oricărui nod este peste valoarea pragului specificat, acel nod este activat, trimițând date către următorul strat al rețelei. În caz contrar, nu sunt transmise date mai departe de către acel neuron.

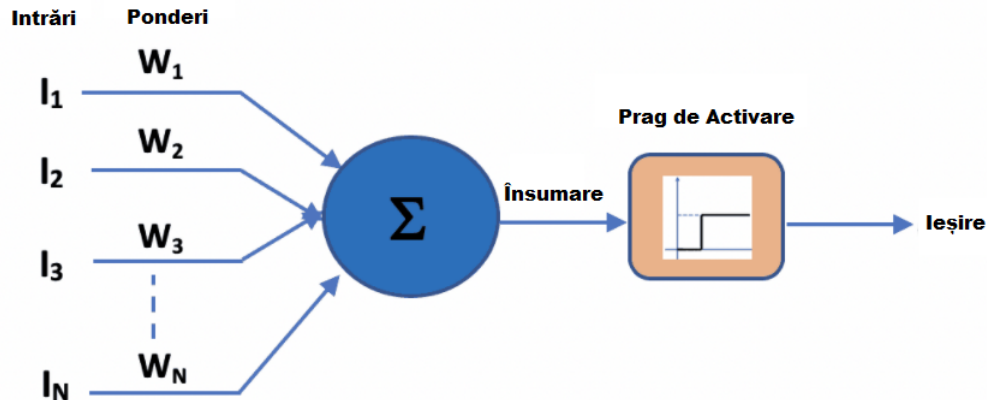


Figura 2.4: Activarea unui nod.

Antrenarea unei rețele neuronale poate fi descrisă în 3 pași:

1. **Forward Pass** - Caracteristicile sunt trecute prin rețea cu scopul de a obține o predicție.
2. **Se calculează eroarea de predicție** - folosind un loss function (funcție de pierdere).
3. **Backpropagation** - Eroarea se propagă înapoi spre stratul de intrare, schimbând ponderile neuronilor direct proporțional cu influența acestora în producerea predicției, în cazul în care predicția nu este foarte bună.

CNN

Chiar dacă **rețelele neuronale convolutive** (en: CNN - Convolutional Neural Network) sunt folosite în mare parte în domeniul *Computer Vision* (Viziune Asistată de Calculator), am decis să le prezint deoarece au obținut rezultate foarte bune și în domeniul PLN. Intuițiile din spatele CNN-urilor sunt oarecum mai ușor de înțeles pentru cazul de utilizare în domeniul Computer Vision, așa că voi începe de acolo și apoi mă voi îndrepta spre PLN.

CNN-urile [10] sunt în esență doar câteva straturi de convoluții cu funcții de activare neliniare precum *ReLU* sau *tanh* aplicate rezultatelor. Într-o rețea neuronală tradițională (ANN), conectăm fiecare neuron de intrare la fiecare neuron de ieșire din stratul următor. În CNN nu facem asta dar, în schimb, folosim convoluții peste stratul de intrare pentru a calcula ieșirea. Acest lucru duce la conexiuni locale, în care fiecare regiune a intrării este conectată la un neuron din ieșire. Fiecare strat aplică filtre diferite, de obicei sute sau mii, și combină rezultatele acestora.

În timpul fazei de antrenament, un CNN învață automat valorile filtrelor sale pe baza sarcinii asigurate. De exemplu, în clasificarea imaginilor, un CNN poate învăța să detecteze marginile pixelilor în primul strat, apoi să utilizeze marginile pentru a detecta forme simple în cel de-al doilea strat și mai apoi să utilizeze aceste forme pentru a forja caracteristici de nivel superior, cum ar fi formele unei fețe în straturile superioare. Ultimul strat este apoi un clasificator care folosește aceste caracteristici de nivel înalt pentru a face o predicție.

Pe lângă convoluții, straturile de pooling sunt un alt element constitutiv al unui CNN. Funcția lor este de a reduce progresiv dimensiunea spațială a reprezentării pentru a reduce cantitatea de parametri din rețea.

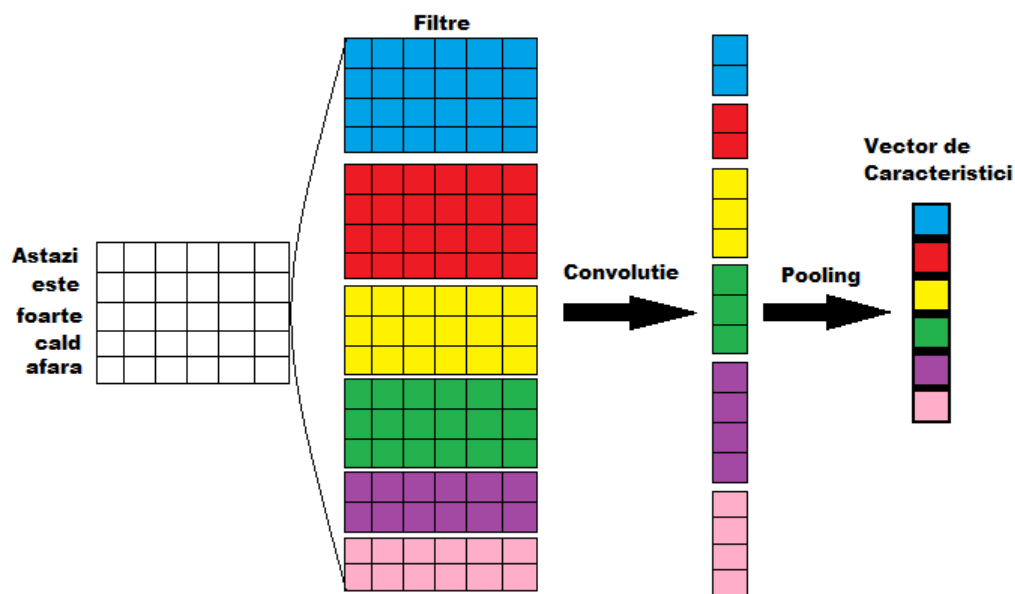


Figura 2.5: Structura CNN în contextul PLN.

În loc de pixeli de imagine, intrarea în cazul sarcinilor legate de PLN e formată din propoziții sau documente reprezentate sub forma de matrice. Fiecare rând al matricei este un vector care reprezintă un cuvânt.

De obicei, acești vectori sunt încorporări de cuvinte rezultate în urma aplicării Word2vec sau GloVe, dar ar putea fi, de asemenea, vectori one-hot care indexează cuvântul într-un vocabular. Pentru o propoziție de 5 cuvinte folosind o încorporare 6-dimensională, am avea ca intrare o matrice de 5×6 . Aceasta ar reprezenta o "imagine" în contextul PLN. (Fig. 2.5)

În viziunea asistată de calculator, filtrele alunecă peste segmente din imagine, dar în PLN folosim de obicei filtre care alunecă pe rânduri întregi ale matricei (cuvinte). Astfel, "lățimea" filtrelor noastre este de obicei aceeași cu lățimea matricei de intrare. Înălțimea sau dimensiunea regiunii poate varia, dar filtre peste 2-5 cuvinte sunt frecvent întâlnite. (Fig. 2.5)

RNN

O **rețea neuronală recurentă** (en: RNN - Recurrent Neural Network) este un tip de rețea neuronală artificială care utilizează date secvențiale sau date din intervale de timp. Acest algoritm de învățare profundă este utilizat în mod frecvent pentru probleme ordinale sau temporale, cum ar fi traducerea limbajului, recunoașterea vorbirii și multe altele. La fel ca rețelele neuronale simple (ANN) și rețelele neuronale convoluționale (CNN), rețelele neuronale recurente utilizează date pentru a învăța tipare.

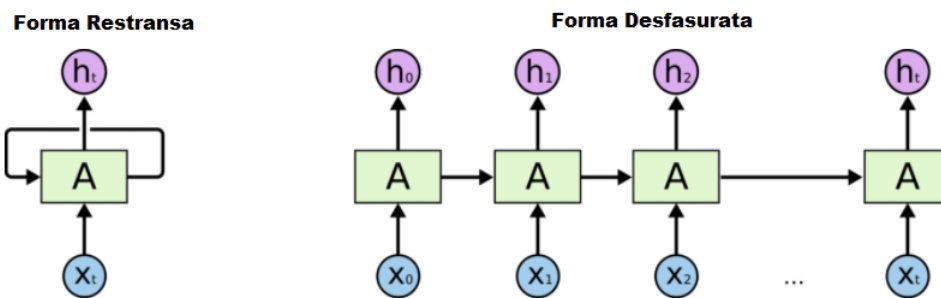


Figura 2.6: Reprezentarea grafică a unui RNN. [23]

RNN-urile se disting prin "*memoria*" lor, deoarece iau informații de la intrările anterioare pentru a influența intrarea și ieșirea curentă. În timp ce rețelele neuronale tradiționale presupun că intrările și ieșirile sunt independente una de cealaltă, ieșirea rețelelor neuronale recurente depinde de elementele anterioare din secvență.

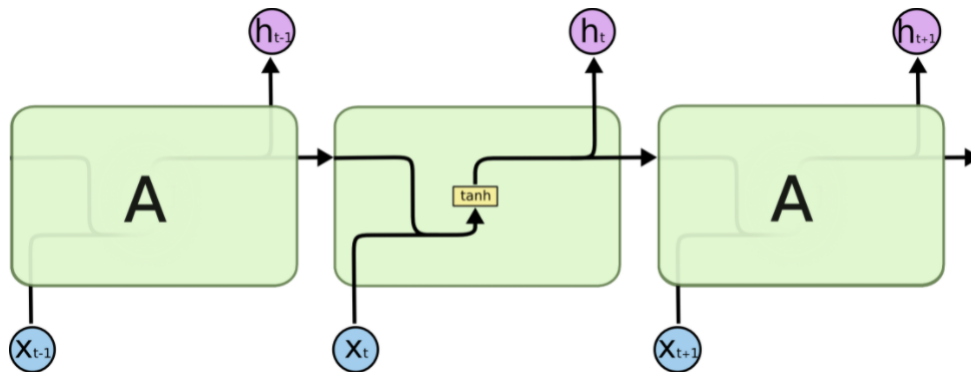


Figura 2.7: Structura unei celule RNN [23]

Un bonus teoretic adus de RNN este ideea că ar putea fi capabile să conecteze informațiile anterioare la sarcina actuală. Problema apare însă în momentul în care este vorba de secvențe foarte lungi. În acel caz, din cauza construcției, RNN-urile nu mai fac față la volumul de informație și încep să piardă informațiile care se află la distanța mai mare față de poziția curentă. Această problemă se numește problema memoriei scurte și este rezolvată în versiuni mai noi de RNN pe care le voi prezenta în continuare.

LSTM

Long Short Term Memory [14] este o variantă mai specială de RNN care reușește să învețe dependențe pe perioade mai lungi. Această arhitectură lucrează extraordinar de bine pe multe tipuri de probleme și este acum utilizată la scară largă în domenii precum procesarea de semnale sau PLN.

LSTM-urile sunt construite pentru a evita problema memoriei scurte din RNN-uri. Acestea pot să își amintească cu ușurință informații întâlnite cu mult timp în urmă așadar au o putere de învățare mult mai mare.

LSTM-urile au, similar cu RNN, structură asemănătoare unui lanț deosebire fiind însă modulul care se repetă. Acesta are o structură diferită (vezi Fig. 2.8). În loc să existe un singur strat de rețea neuronală, există patru care interacționează astfel încât să selecteze informațiile importante și să le elimine pe cele de care nu mai e nevoie.

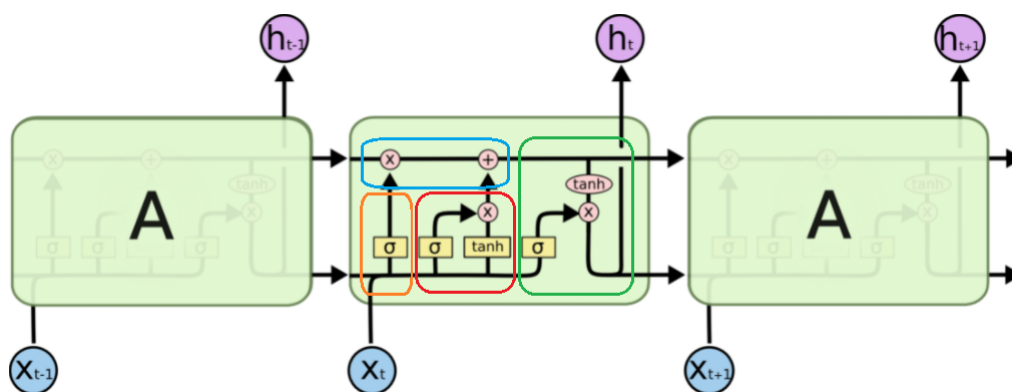


Figura 2.8: Structura unei celule LSTM [23]

Cheia LSTM-urilor o reprezintă starea celulei, bara ce trece prin partea de sus a Fig. 2.8, marcată cu albastru. Starea celulei este cam ca o bandă transportoare. Se întinde pe întregul lanț, cu doar câteva interacțiuni minore. Este foarte ușor ca informațiile să curgă de-a lungul acesteia neschimbate. LSTM are capacitatea de a elimina sau de a adăuga informații la starea celulei prin intermediul unor porți. Un LSTM are trei astfel de porți și anume:

- **Poarta de uitare** (*Forget Gate*) - Prin intermediul acestei porți, se vor alege informațiile care să fie scoase din starea celulei. Marcat cu portocaliu.
- **Poarta de intrare** (*Input Gate*) - Prin intermediul acestei porți se vor alege relațiile care să fie încărcate în starea celulei. Marcat cu roșu.
- **Poarta de ieșire** (*Output Gate*) - Prin intermediul acestei porți se selectează din starea celulei anumite caracteristici ce vor fi trimise spre următoarea celulă din rețea. Marcat cu verde.

GRU

O variantă de LSTM este **Gated Recurrent Unit** [5]. Acest model combină porțile de uitare și de intrare într-o singură poartă de actualizare.

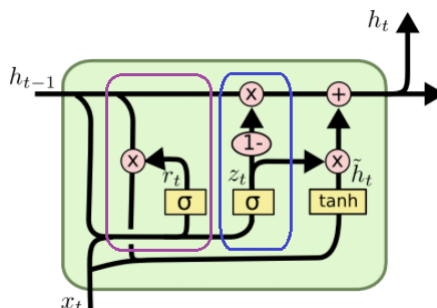


Figura 2.9: Structura unei celule GRU [23]

Diferența cheie dintre GRU și LSTM este legată de structura unei celule și complexitatea modelului (Fig. 2.9). GRU are doar două porți care sunt **poarta de resetare** (*mov*) și **poarta de actualizare** (*bleumarin*). Dacă setul de date este mic, atunci GRU este preferat deoarece este un model mai simplu iar în cazul seturilor de date mult mai mari, se poate merge pe varianta cu un model ce folosește LSTM.

BI-LSTM

Un *LSTM bidirecțional* sau *BI-LSTM* este o versiune de RNN compusă din două LSTM-uri:

- **Înainte:** prelucrează intrarea de la stânga la dreapta.
- **Înapoi:** prelucrează intrarea de la dreapta la stânga.

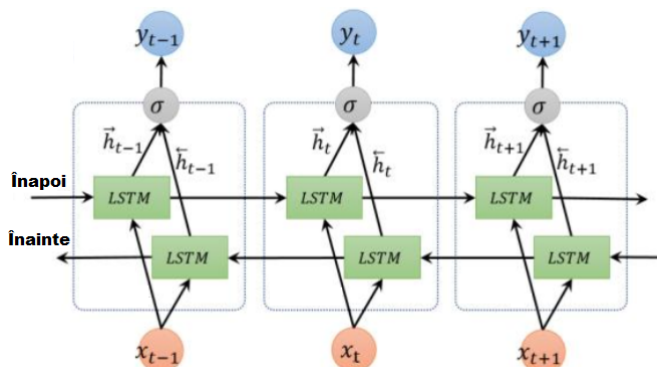


Figura 2.10: Structura unui BI-LSTM [9]

BI-LSTM crește eficient cantitatea de informații disponibile rețelei, optimizând contextul disponibil algoritmului (știind pentru un cuvânt anume ce se află înainte sau după acesta).

2.2.3 Modele complexe

ELMo

Embeddings from Language Models [26] este un model de incorporare care reprezintă cuvintele în spațiul vectorial. Acest model presupune o reprezentare contextualizată a cuvintelor care modelează atât caracteristici complexe ale utilizării cuvintelor (sintaxă și semantică) dar și modul în care aceste utilizări variază în contextele lingvistice (pentru a modela polisemia). Acest model folosește straturi de BI-LSTM pentru a extrage caracteristici.

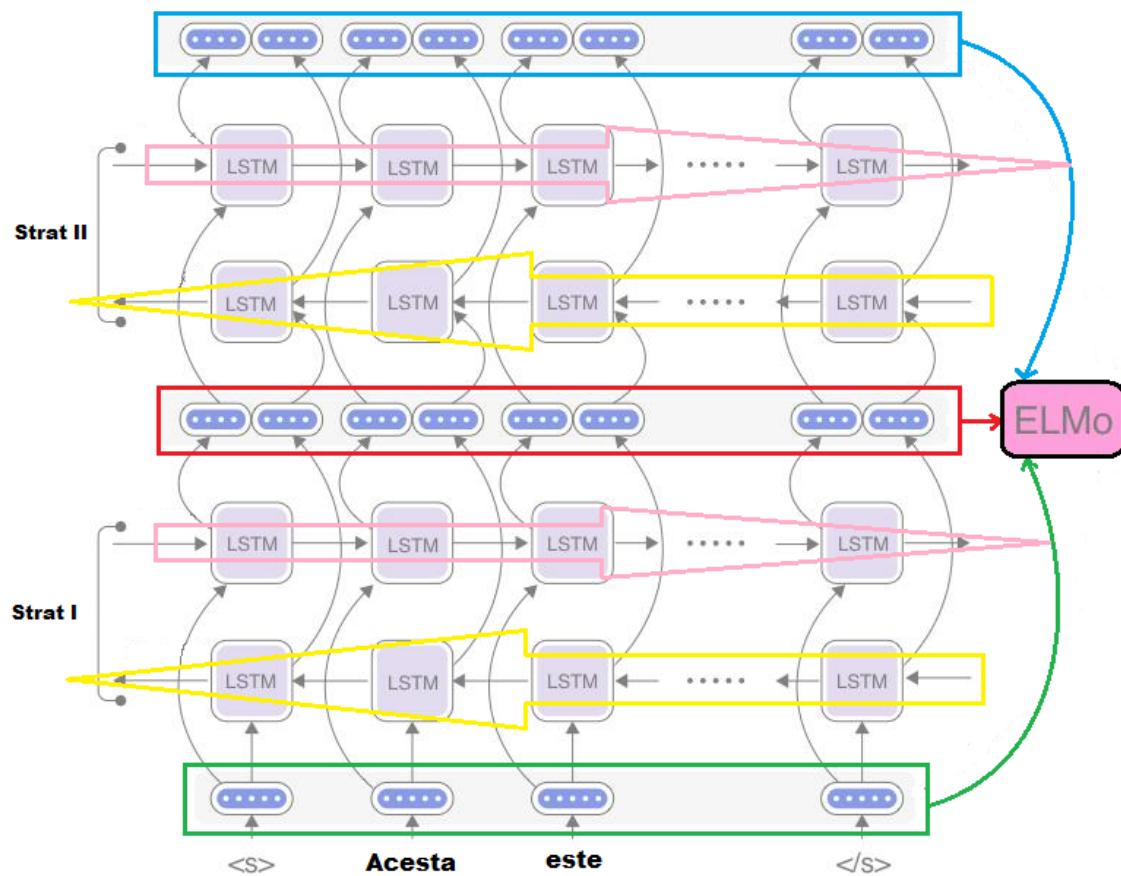


Figura 2.11: ELMo cu două straturi [13]

ELMo poate produce reprezentări diferite pentru același cuvânt deoarece are capacitatea de a recunoaște dacă cuvântul este folosit în contexte diferite. Această abilitate unică înseamnă, în esență, că încorporările ELMo au mai multe informații disponibile și, prin urmare, performanța modelelor de clasificare va crește probabil dacă le folosim.

ELMo reprezintă unul dintre cele mai mari progrese în NLP, deoarece a fost în esență primul model care a adus contextualizarea în centrul atenției, oferind performanțe mai bune într-o mulțime de sarcini.

Capitolul 3

Domeniul problemei

În cadrul acestui capitol voi prezenta o serie de lucrări în care este abordată problema detectării de știri false. Pe final vom discuta situația curentă și ce posibilități de dezvoltare avem, în funcție de soluțiile deja existente.

3.1 Abordări înrudite

- *Agrawal și colab.* [1] propun un model hibrid bazat pe învățarea profundă pentru a detecta știrile false folosind Blockchain. **DeHiDe** (introdus în lucrare) este un framework bazat pe blockchain folosit pentru schimbul legitim de știri prin eliminarea știrilor false. Acesta combină beneficiul blockchain-ului cu un model inteligent de învățare profundă pentru a consolida robustețea și acuratețea sistemului. Autorii compară metoda propusă cu metodele de ultimă generație existente și se așteaptă ca DeHiDe să depășească abordările de ultimă generație în ceea ce privește serviciile, caracteristicile și performanța.
- *Botnevik și colab.* [2] propun **BRENDA**, o extensie de browser care automatizează întregul proces de *claim-checking* (verificarea valorii de adevăr a unor afirmații). În culise, BRENDA folosește o arhitectură de rețea neuronală profundă antrenată pentru a identifica automat afirmațiile demne de verificat. Tot BRENDA clasifică afirmațiile și prezintă dovezi pe baza carora s-a făcut clasificarea.
- *Gautam și colab.* [12] propun **SGG**, un sistem de detectare a știrilor false bazat pe instrumente de parafrizare (*Spinbot*), verificare gramaticală (*Grammarly*) și încorporare de cuvinte (*GloVe*). Folosind aceste instrumente, au reușit să extragă caracteristici noi care ar putea oferi rezultate de ultimă generație pe setul de date *AMT Fake News* și rezultate comparabile pe setul de date *Celebrity* atunci când sunt combinate cu alte caracteristici esențiale.

- *Karimi și colab.* [16] au propus o tehnică care utilizează CNN pentru a extrage caracteristici dintr-un text și apoi aplică LSTM pe aceste caracteristici pentru a capta dependența temporală în întregul text. Precizia realizată în timpul experimentelor efectuate în lucrare nu merita menționată.
- *Rashkin și colab.* [27] au făcut o comparație între limbajul folosit în știrile reale respectiv în știrile false pentru a găsi caracteristici lingvistice specifice știrilor false. Acestia au construit două modele mai simple (Maximum Entropy și Naive Bayes) și un LSTM, acesta obținând cele mai bune rezultate.
- *Jwa și colab.* [15] introduc metode automate de detectare a știrilor false, bazate pe date (*data-driven*). Aceștia se folosesc de BERT pentru a detecta știrile false, analizând relația dintre titlu și conținutul știrilor. Pentru a îmbunătăți în continuare performanța, sunt adunate date de știri suplimentare și utilizate pentru pre-antrena modelul. Cercetătorii determină că natura contextuală a modelului BERT îl face perfect pentru sarcina de clasificare de știri, acesta îmbunătățind scorul F1 cu 0.14 față de modele anterioare.
- *Kai și colab.* [32] au propus o tehnică care exploatează relația dintre editori, știri și utilizatori pentru a detecta știrile false. Pentru modelarea acestei relații au creat un framework denumit **TriFN** care folosește un clasificator liniar. Ei atribuie fiecărui utilizator un scor de credibilitate pe baza comportamentului din mediul online și merg pe ideea ca un utilizator cu scor de credibilitate scăzut e mai probabil să distribuie știri false. Clasificatorul lor depășește modelele de clasificare standard precum Naive Bayes, Arborii de Decizie și Random Forest (ansambluri de arbori de decizie).
- *Kula și colab.* [18] prezintă o arhitectură hibridă care conectează BERT cu RNN pentru a detecta știrile false.
- *Paschalides și colab.* [24] introduc **Check-It**, un sistem care combină, într-un mod inteligent, o varietate de semnale într-un pipeline pentru identificarea știrilor false. Check-It este dezvoltat ca un plugin pentru browser web cu obiectivul de detectare eficientă și în timp util a știrilor false, respectând și confidențialitatea utilizatorului. Rezultatele experimentale arată că Check-It este capabil să depășească metodele de ultimă generație. Pe un set de date, format din 9 milioane de articole etichetate drept false și reale, Check-It obține o precizie de clasificare care depășește 99%.
- *Perez-Rosas și colab.* [25] se concentrează pe identificarea fără implicare umană a conținutului fals în știrile online. Contribuția lor este multiplă. În primul

rând, introduc două seturi de date noi pentru detectarea știrilor false, acoperind șapte domenii de știri diferite. În lucrare este descris în detaliu și procesul de colectare, adnotare și validare a datelor. În al doilea rând, desfășoară un set de experimente pentru a construi clasificatoare mai precise.

- *Ruchansky și colab.* [28] au propus un model numit **CSI** care clasifică un articol pe baza a trei caracteristici: conținutul articolului, răspunsul utilizatorului care primește articolul și utilizatorii care îl promovează. CSI este compus din trei module: *Capture*, *Score* și *Integrate*. Primul modul se bazează pe răspuns și text; folosește un RNN pentru a surprinde tiparul temporal al activității utilizatorului pe un anumit articol. Al doilea modul învață caracteristica sursă pe baza comportamentului utilizatorilor, iar cele două sunt integrate cu al treilea modul pentru a clasifica un articol ca fiind fals sau nu. Rezultatele prezentate arată că CSI obține o precizie mai mare decât celelalte tehnici comparate în lucrare.
- *Tanik și colab.* [29] propun două modele eficiente bazate pe învățarea profundă pentru rezolvarea problemelor de detectare a știrilor false în cadrul mai multor domenii. Ei își evaluează tehnicile pe cele două seturi de date lansate recent, și anume *FakeNews AMT* și *Celebrity*. Sistemele propuse oferă performanțe încurajatoare, depășind sistemele bazate pe *feature engineering* (extragere manuală de caracteristici) cu o marjă semnificativă.
- *Siering și colab.* [33] au abordat subiectul detectării de înșelătorii pe platformele de crowdfunding. Aceștia extrag caracteristici legate de limbajul și conținutul diferitelor tipuri de comunicare de pe aceste platforme. Ei examinează atât comunicațiile statice, cât și cele dinamice. Comunicarea dinamică se realizează în timp real, adică este similară comunicării față în față și comunicarea statică se efectuează cu întârziere. În abordarea lor au folosit diferite clasificatoare, obținând cele mai bune rezultate cu un model tip mașină cu suport vectorial.
- *Singhania și colab.* [34] au propus **3HAN**, o rețea ierarhică cu mecanism de atenție. 3HAN are trei niveluri: unul pentru cuvinte, unul pentru propoziții și ultimul pentru titlu. Articolele sunt procesate ierarhic de jos în sus. 3HAN acordă o importanță diferită părților unui articol, datorită celor trei niveluri de atenție din cadrul acestuia. Modelul obține o acuratețe de 96.77% pe un set foarte mare de date. Diferența între 3HAN și alte modele este faptul că în cazul 3HAN se pot reprezenta grafic stările ascunse din cadrul celor trei mecanisme de atenție pentru a marca anumite secvențe din articol pe care modelul le-a considerat importante în luarea deciziei.

- *Vaibhav și colab.* [35] propun un model de tip rețea neuronală sub forma de graf pentru a clasifica articolele de știri captând interacțiunea frazelor din document. Prin experimente, au arătat că metoda lor obține o precizie de ultimă generație pe seturile de date existente.
- *Volkova și colab.* [36] au propus modele predictive pentru a clasifica 130 de mii de postări de știri ca suspecte sau verificate și pentru a prezice patru subtipuri de știri suspecte - satiră, fraudă, clickbait și propagandă. În cadrul lucrării este demonstrat faptul că modelele bazate pe rețele neuronale au rezultate mult mai bune decât modelele lexicale.
- *Wang și colab.* [37] au propus un model hibrid cu CNN-uri care combină meta-caracteristici și text pentru a determina veridicitatea unui afirmații. În acest articol este propus și datasetul **LIAR**. Modelul hibrid obține rezultate mult mai bune decât alte modele încercate pe LIAR (SVM, Log Reg, CNN, Bi-LSTM).
- *Xu și colab.* [38] prezintă **FaNDs**, un sistem care detectează eficient știrile false. Sistemul se bazează pe mai multe concepte utilizate în unele lucrări anterioare, dar într-un context diferit. Există două concepte principale: un graf de inconsistență și un flux de energie. Graful de inconsistență conține știri sub formă de noduri și opinii incoerente între ele pe post de muchii. Fluxul de energie atribuie fiecărui nod o energie inițială și apoi o anumită energie se propagă de-a lungul muchiilor până converge distribuția energiei pe toate nodurile. Nodurile cu energie ridicată sunt candidații pentru a fi știri false. Autorii prezintă și rezultate conform cărora sistemul FaNDs este mult mai sensibil la știri false decât alte modele.
- *Yang și colab.* [19] au propus un model pentru detectarea știrilor false pe social media prin clasificarea căilor de propagare ale știrilor. Mai întâi este modelată calea de propagare a fiecărei știri ca o serie de timp (*time-series*) multivariată în care fiecare tuplu reprezintă caracteristicile unui utilizator implicat în procesul de răspândire al știrilor false la un moment dat. După asta, se folosesc de un clasificator compus din CNN și RNN pentru a captura variațiile caracteristicilor pe durata întregului proces de distribuire. Rezultate experimentale pe trei seturi de date demonstrează că modelul propus poate detecta știri false cu precizie 85% pe Twitter respectiv 92% pe Sina Weibo.
- *Zhang și colab.* [41] introduc un nou model de inferență automată a credibilității știrilor false, și anume **FAKEDETECTOR**. Pe baza unui set de caracteristici explicite și latente extrase din informațiile textuale, FAKEDETECTOR construiește un model de rețea difuză pentru a învăța simultan reprezentările

articolelor de știri, ale creatorilor și ale subiectelor. S-au făcut experimente ample pe un set de date de știri false din lumea reală pentru a compara FAKE-DETECTOR cu mai multe modele de ultimă generație, iar rezultatele experimentale au demonstrat eficiența modelului propus.

3.2 Analiza situației curente

Privind în ansamblu ideea de clasificare de știri observăm faptul că există extrem de multe abordări cu rezultate foarte bune așadar situația pare destul de promițătoare.

O primă problemă însă este ca majoritatea modelelor propuse au fost antrenate pe texte scrise în limba engleză. Chiar dacă modelele ar trebui să fie independente de limba folosită, acest lucru nu este garantat așadar nu putem să luăm un model antrenat pentru limba engleză și să îl folosim pentru română deoarece este foarte probabil să nu meargă și să fie doar o pierdere de timp.

Pentru a crește șansele de succes, ar trebui să dezvoltăm propriile modele de clasificare (inspirate din direcțiile existente) și să le antrenăm pe text scris în limba română. Pentru asta am avea nevoie de un set de date cu articole de știri în română.

O altă problemă o reprezintă faptul că un astfel de set de date nu prea există. O posibilitate ar fi să creem propriul set de date cu știri în limba română iar apoi să îl etichetăm în funcție de modul de clasificare. Acest lucru ar dura destul de mult dar din păcate nu există o altă alternativă dacă dorim să obținem performanțe.

Un alt factor care ne îngreunează situația este faptul că nu am găsit alte modele de clasificare de știri antrenate pentru articole în limba română așadar va fi destul de greu să compar rezultatele obținute de mine cu alte rezultate care nu există. Pentru a scăpa de această problemă, în loc să antrenez un singur model, voi antrena mai multe urmând ca apoi să compar performanțele lor între ele.

Totuși, spre norocul nostru, în literatura de specialitate există modele precum BERT și ELMo antrenate și pentru limba română. Dacă timpul îmi permite, le voi încerca și pe acelea pe setul de date pe care îl voi colecta urmând să observ dacă îmbunătățesc situația sau nu.

Un ultim lucru pe care doresc să îl menționez este faptul că mă voi axa pe modele secvențiale bazate pe rețele neuronale profunde deoarece le consider ca fiind cea mai bună potrivire în contextul acestei probleme. Am decis să nu folosesc modele precum *Bag of Words* sau *TF-IDF* deoarece le consider învechite (chiar dacă probabil ar oferi o bază de plecare destul de bună).

Capitolul 4

Soluția propusă - Prezentare generală

Vă propun un sistem bazat pe modele inteligente pentru a automatiza procesul de clasificare al articolelor de știri. Pașii pentru dezvoltarea unui astfel de sistem sunt prezentați în Fig. 4.1 și sunt cum urmează:

1. **Construirea setului de date** (*mov*)
2. **Alegerea modelului potrivit** (*portocaliu*)
3. **Integrarea într-un API** (*turcoaz*)

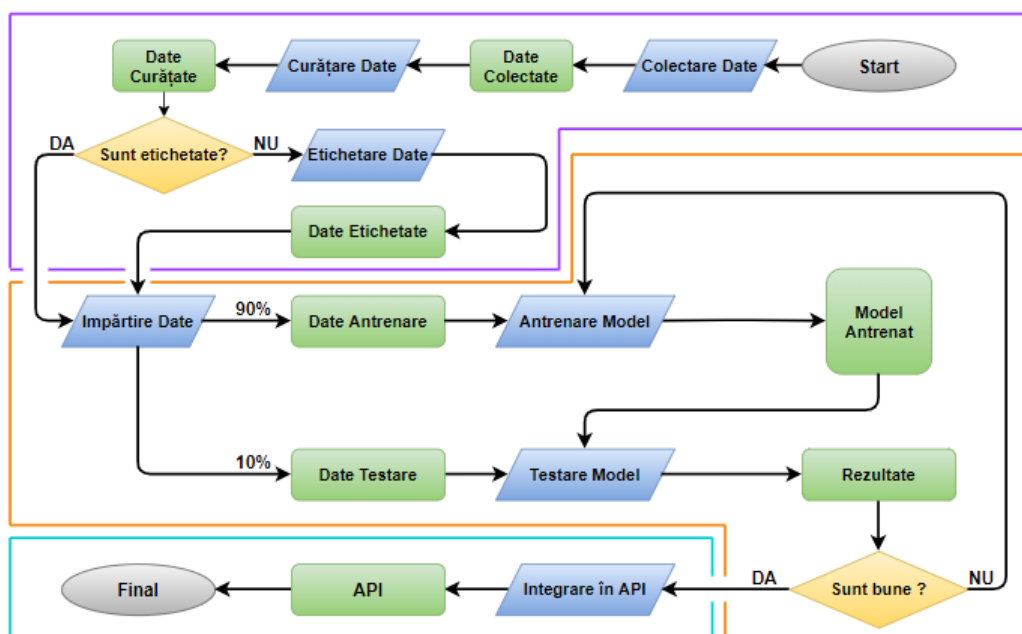


Figura 4.1: Flowchart pentru dezvoltarea soluției propuse.

Pentru a înțelege în detaliu modul de funcționare al unui astfel de sistem și ce presupune exact fiecare etapă de dezvoltare, vă propun să mă însoțiți în următoarele subcapitole.

4.1 Construirea setului de date

Colectarea datelor

După cum am menționat și în 3.2, nu am găsit un set de date cu articole de știri în limba română așadar am hotărât să construiesc propriul set de date. Acest lucru a implicat folosirea unor tehnici de *web-scraping* (extragere de informații din pagini web) pe diferite site-uri care publică știri.

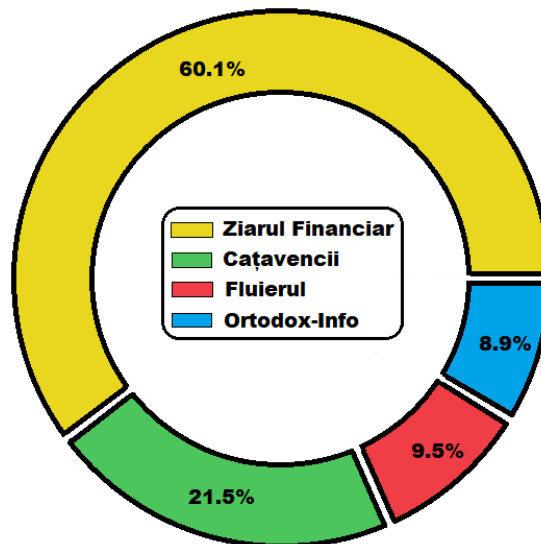


Figura 4.2: Distribuția surselor de știri din setul de date.

În total în cadrul setului de date am colectat aproximativ 50.000 articole de știri provenite din patru surse de știri. Distribuția surselor se poate observa în Fig. 4.2. Perioadele în care au fost publicate știrile se pot observa în tabela 4.1. Știrile au fost colectate și etichetate în decursul a trei săptămâni în luna aprilie. Câteva din domeniile de care aparțin știrile sunt: *Economie, Politică, Religie, Educație și Sănătate*.

Idx.	Sursă	Număr articole	Perioada publicării
1	Ziarul Financiar	30.036	Aprilie 2020 - Mai 2021
2	Cațavencii	10.742	Iunie 2011 - Mai 2021
3	Fluierul	4.741	Ianuarie 2015 - Mai 2021
4	Ortodox-Info	4.435	Februarie 2009 - Mai 2021

Tabela 4.1: Perioadele în care au fost publicate articolele extrase.

Pe lângă articole am colectat și scurta prezentare a acestora (vezi Fig. 4.3). Motivul pentru care am colectat și prezentarea este că doresc să dezvolt modele care să clasifice articolele de știri doar pe baza acestei prezentări. Astfel de modele ar necesita mai puțină putere de calcul pentru antrenare și *ar putea obține rezultate chiar foarte bune* (presupunere pe care o vom testa ulterior).



Figura 4.3: Exemplu de prezentare a unui articol de știri. [4]

Curățarea datelor

După colectarea datelor a urmat o etapă de curățare. Această etapă a fost necesară deoarece *scraperul* (unealta folosită pentru colectarea datelor) a extras și alte lucruri pe lângă textul știrilor. În cadrul acestei etape fiecare articol a fost trecut prin mai multe filtre de curățare pentru a-l face lizibil. Filtrele folosite au avut următoarele întrebări:

- Eliminarea tag-urilor HTML.
- Eliminarea URL-urilor.
- Eliminarea spațiilor albe.

Pe lângă acestea, au fost eliminate intrările care fie nu aveau conținut (în cazul știrilor cu clipuri video sau poze), fie au cauzat probleme. După această eliminare am trecut de la 58.000 (numărul de articole extrase initial) la 49.954 (numărul de articole valide rămase).

Etichetarea datelor

În elaborarea modelului am decis să trec de la ideea de **clasificare binară** (întâlnită în majoritatea abordărilor existente) la o **clasificare multi-class** așadar pe lângă clasele uzuale (**știre reală** / **știre falsă**) am introdus și clasa **știre satirică**. Știrile satirice se aseamănă și cu știrile reale (implică la fel de multă muncă în scriere) dar și cu știrile false (prezintă informații exagerate sau reinterpretate) așadar *au potențial destul de mare să încurce sistemul de clasificare* (motiv pentru care le-am și adăugat).

Chiar dacă în prima faza alegerea surselor de știri pe baza cărora a fost creat setul de date pare aleatoare, nu este. Sursele au fost alese special pentru a ușura procedeul de etichetare al articolelor:

Ca sursă reprezentativă pentru știrile reale am ales *Ziarul Financiar* deoarece este considerată una din sursele cele mai demne de încredere la noi în țară. [31]. Pe lângă asta, sunt următor al publicației de în jur de 3 ani și nu a existat nici un moment în care să mă îndoiesc de informațiile prezentate. Toate știrile extrase din această sursă au fost etichetate ca fiind **știri de încredere (REAL)**.

Ca sursă reprezentativă pentru știrile satirice am ales *Cațavencii*, una dintre cele mai vechi publicații de satiră din România. Și la această publicație sunt abonat și pot spune cu încredere că fiecare articol este gândit în detaliu și scris foarte bine (chiar dacă are scop umoristic și nu prezintă neapărat întâmplări reale). O altă sursă de unde am vrut să colectez știri a fost *TimesNewRoman*¹ dar din păcate scraperul nu funcționa prea bine pe site-ul lor. Toate știrile extrase de pe *Cațavencii* au fost etichetate ca fiind **știri satirice** (SATIRĂ).

Pentru a găsi surse de știri false am căutat în cele mai întunecate cotloane din spațiul online românesc printre sute de site-uri dubioase. Munca de colectare mi-a fost îngreunată și de guvernul țării care prin intermediul programului *Inforadar* a închis deja o mare parte din site-urile care promovau informații false înainte să apuc să colectez știri de pe ele. Cu toate astea am reușit să găsesc *Fluierul* și *Ortodox-Info* de unde am preluat cele mai reprezentative articole. Știrile preluate din aceste două surse au fost etichetate ca **știri îndoielnice** (FALS).

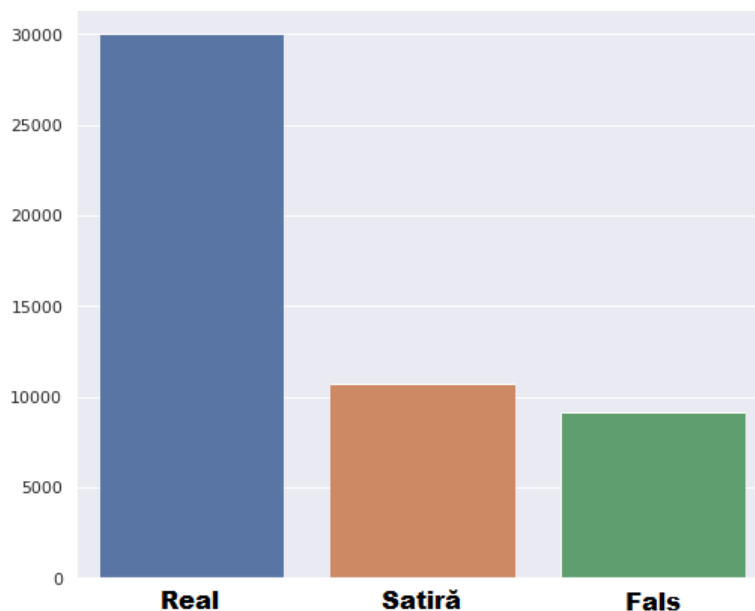


Figura 4.4: Distribuția claselor după etichetare.

Pentru a îmbunătăți calitatea setului de date și pentru a verifica etichetarea în funcție de sursă am selectat aleator **10%** din fiecare clasă spre a le verifica manual. În urma acestei verificări am determinat faptul că știrile clasificate ca fiind reale sunt într-adevăr reale, situația stând la fel de bine și în cazul satirei.

Problema însă apare în cadrul știrilor etichetate ca fiind îndoielnice printre care, din păcate, s-au mai strecurat și știri reale. Aproximez că undeva în jur de 500-1000 știri etichetate ca fiind îndoielnice sunt de fapt de încredere. Chiar dacă o parte din ele au fost eliminate, există posibilitatea să fi rămas și altele etichetate greșit.

¹<https://www.timesnewroman.ro/>

4.2 Alegerea modelului potrivit

4.2.1 Modele propuse

Acum că setul de date este pregătit, o mare parte din treabă e gata. Rămâne să alegem cel mai bun model în urma derulării unor experimente. Arhitecturile modelelor propuse sunt prezentate în Fig. 4.5.

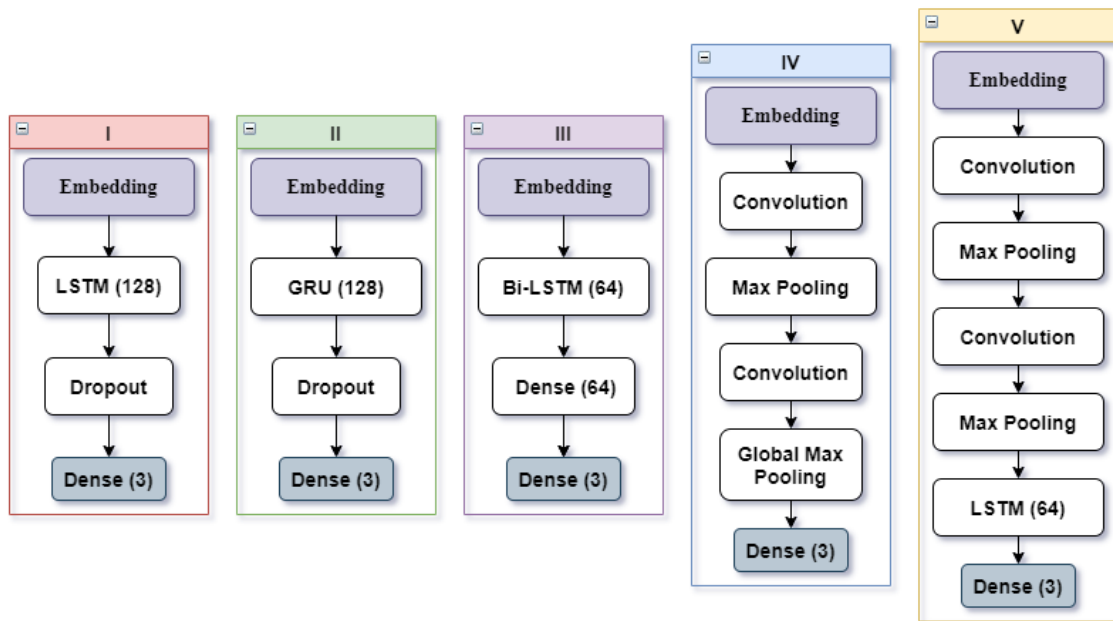


Figura 4.5: Arhitecturile modelelor propuse.

Pentru **Modelul I** am decis să fie bazat pe *LSTM* în urma unui studiu în detaliu al lucrărilor conexe, unde am observat faptul că modelele bazate pe *LSTM* sunt des întâlnite în problemele de clasificare de text și produc rezultate foarte bune.

În cazul **Modelului II** am decis să folosim *GRU* în loc de *LSTM* deoarece modelele bazate pe *GRU* se antrenează cu 30% mai repede și de multe ori produc rezultate mai bune decât un *LSTM* pentru seturi de date de dimensiuni reduse (cum e considerat și cel propus de noi) [39].

La **Modelul III** ne-am gândit că o rețea bazată pe un *Bi-LSTM* poate extrage informații și din stările anterioare dar și din cele posterioare (așadar producând mai mult context) deci ar trebui să depășească performanța unui model unidirecțional (cum sunt cele prezentate mai devreme).

În **Modelul IV** abordarea e puțin mai diferită. Am ales să folosim CNN-uri deoarece s-au dovedit ca fiind fiabile și în domeniul PLN [40]. Pe lângă asta ar fi și un motiv legat de performanță și anume faptul că rețelele CNN pot fi paralelizate (convoluțiile fiind printre operațiile fundamentale executate pe un GPU), spre deosebire de modelele secvențiale precum RNN și GRU prin care se trec reprezentările cuvintelor secvențial (unul câte unul).

În cazul **Modelului V** am vrut să combinăm puterea unui CNN cu cea a unui model secvențial, asemănător cu abordarea folosită la elaborarea modelului ELMo (chiar dacă în cazul nostru este vorba de un LSTM simplu în loc de două rețele Bi-LSTM). Acest model este experimental și nu știm dacă rezultatele vor fi mai bune sau nu.

4.2.2 Experimente

Pregătirea Datelor

Chiar dacă datele au fost curățate, mai trebuie trecute printr-o etapă de preprocesare înainte de a le introduce în model. În cadrul acestei etape sunt eliminate cheștiile care pot încurca modelul în procesul de învățare. În cadrul experimentelor au fost efectuate următoarele preprocesări (vezi și Fig. 4.6) :

- Textul a fost transformat în **litere mici**.
- **Punctuația** a fost eliminată.
- **Valorile numerice** au fost eliminate.
- **Diacriticele** au fost eliminate. *
- Au fost eliminate **cuvintele de legătură**. *

După preprocesare urmează *tokenizarea* articolelor, care presupune împărțirea în cuvinte a fiecarui articol și reprezentarea lui în formă vectorială folosind un tokenizator.

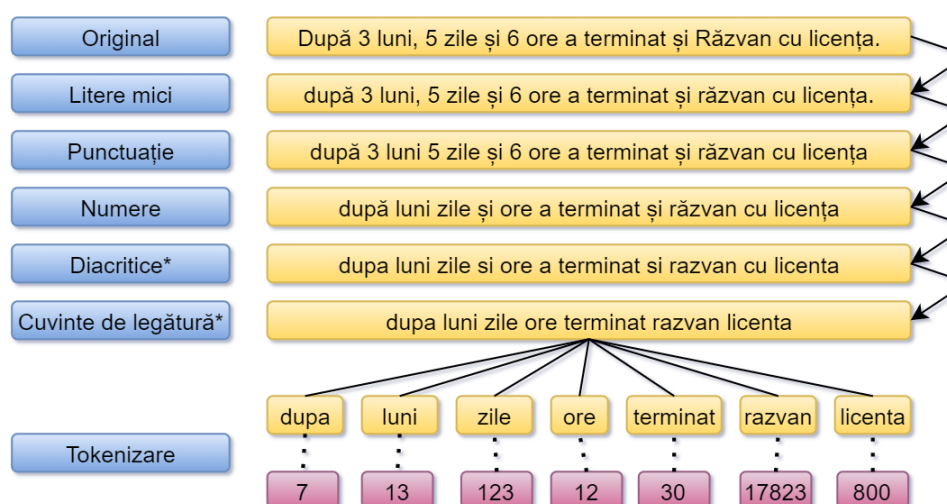


Figura 4.6: Preprocesare și tokenizare.

Tokenizatorul își construiește o reprezentare pentru fiecare cuvânt întâlnit în cadrul articolelor, ținând cont și de frecvența de apariție a cuvântului. Așadar pentru

a transforma un articol de știri în forma vectorială, ne folosim de reprezentările produse de tokenizator pentru fiecare cuvânt.

O mică problemă ce apare însă în urma tokenizării este faptul că vom avea în setul de date vectori de lungimi diferite (lungimile vectorilor variind odată cu lungimea articolelor) așadar ar trebui să setăm un prag maxim pentru lungimea vectorului.

Pentru experimentele derulate, am ales lungimea maximă ca fiind fie **512** (pentru a putea compara cu rezultate obținute de modele din familia BERT), fie **1024** (pentru a compara cu rezultate obținute de ELMo). Pentru secvențele mai lungi decât lungimea maximă excesul a fost ignorat iar pentru cele sub lungimea maximă am adăugat *padding* (valori de umplură) până s-a ajuns la lungimea necesară.

Împărțirea Datelor

În urma pregătirii datelor avem un număr de 49.954 articole, fiecare din ele fiind reprezentate în două forme, una cu lungimea de 512 și una cu lungimea de 1024.

Urmează un pas în care formăm perechi de X și y unde X este vectorul (fie el reprezentat pe 512 sau 1024) ce reprezintă un articol și y este eticheta asociată acelui articol în urma etichetării. După împerechere avem 49.954 perechi de forma (X, y) .

Odată ce avem perechile pregătite, le împartim în trei subseturi și anume:

- **Subset Antrenament** (*train*) - compus din 75% din setul de date. Este folosit pentru antrenarea modelelor.
- **Subset Validare** (*validation*) - compus din 15% din setul de date. Este folosit pentru a verifica cum se comportă modelul pe date nevazute și pentru a regla hiperparametrii modelului.
- **Subset Testare** (*test*) - compus din 10% din setul de date. Este folosit în urma obținerii celui mai bun model pentru a vedea cât de bine generalizează.

Ideea de bază la această împărțire este că dacă nu o facem, în cele mai multe cazuri, modelul se va mula pe datele noastre obținând rezultate foarte bune urmând ca ulterior să o dea în bară în momentul în care este folosit pe date din viața reală pe care nu le-a mai văzut. De menționat este faptul că împartirea datelor s-a obținut folosind un *random state* ($rs = 42$) pentru a facilita reproducerea rezultatelor.

Antrenarea modelelor

Procesul de antrenare al unei rețele neuronale este descris în mare în 2.3.2. Procesul este similar și în cazul CNN-urilor sau al modelelor secvențiale. Nu voi descrie

fiecare hiperparametru sau toate ideile folosite la antrenare deoarece se presupune că cititorul cunoaște macar în mare subiectul.

La antrenarea fiecărui model am pornit de la 0 cu un model foarte simplu urmând ca ulterior să ajung la versiunile finale prezentate în figura 4.5. De menționat sunt următoarele:

- **Optimizatorul folosit:** *Adam* cu rata de învățare între 0.0001 și 0.01 în funcție de arhitectură și rezultatele intermediare.
- **Loss function:** *Categorical Cross Entropy* fiind vorba de o problemă de clasificare multi-class cu trei clase.
- **Dimensiunea unui batch:** Am încercat valori între 16-256 în final alegând 128 ca valoare optimă.
- **Metrici urmărite:** *Loss, Acuratețe ponderată*

Cele prezentate deasupra au fost folosite la *toate modelele*. În continuare voi menționa și chestii specifice fiecărui model.

De menționat pentru **Modelul I** este faptul că am folosit 32, 64 respectiv 128 celule LSTM mergând pe versiunea cu 128 celule + dropout cu rata 0.7 pentru a obține cele mai bune rezultate. Activarea din ultimul strat este '*Softmax*'.

În cazul **Modelului II**, avem un mod de lucru asemanator cu cel din primul model, singura diferență fiind legată de folosirea de GRU în loc de LSTM și alegerea unei rate de dropout mai mici (0.4).

La **Modelul III** am ales să folosesc 64 celule Bi-LSTM deoarece ar produce un numar egal de reprezentări cu numarul de reprezentări produs de LSTM în primul model și de GRU în al doilea ($64 * 2 = 128$). Un strat dens a fost adăugat pentru a crește complexitatea modelului. Activările folosite sunt '*ReLU*' pentru penultimul strat dens și '*Softmax*' pentru ultimul strat dens.

În cazul **Modelului IV** avem convoluții unidimensionale de dimensiune 7 cu cate 32 de filtre fiecare și staturi de pooling de dimensiune 5. Funcțiile de activare pentru convoluții sunt '*ReLU*' având tot '*Softmax*' pe ultimul strat dens. În cadrul experimentelor am ales să folosesc 1,2 si 3 perechi conv + pool obținând cele mai bune rezultate cu 2 straturi de perechi. Pe langă asta am variat dimensiunea filtrelor alegând 3, 5 și 7 rămânând cu 7 ca variantă optimă.

La **Modelul V** am folosit 5 în loc de 7 pentru dimensiunea filtrelor din straturile de convoluție și 3 în loc de 5 pentru dimensiunea zonei de pooling. Aceleași variații au avut loc și în cazul acestui model.

Chiar dacă am derulat extrem de multe experimente cu diferiți hiperparametrii, nu am făcut în nici un caz grid search pentru găsirea hiperparametrilor optimi așa că nu pot garanta în nici unul din cazuri că modelul este antrenat în condiții optime.

Un lucru pe care aproape am uitat să îl menționez este legat de ponderile pentru clase pe care le-am folosit la antrenarea modelelor spre a da o importanță mai mare unor clase slab reprezentate.

Idx.	Clasa	Procentaj din Date	Ponderea Asociată
1	Real	60.13 %	0.55
2	Satiră	21.50 %	1.55
3	Fals	18.37 %	1.81

Tabela 4.2: Ponderile asociate fiecărei clase.

Majoritatea algoritmilor de învățare automată presupun că datele sunt distribuite uniform în cadrul claselor. În cazul problemelor de *class-imbalance* (dezechilibru între clase) problema este că algoritmul va fi mai înclinat spre prezicerea clasei majoritare.

Soluția pentru problema de class-imbalance este una destul de simplă și presupune calculul unei *ponderi* pentru fiecare clasă. Clasele bine reprezentate vor avea o pondere mai mică iar cele slab reprezentate, o pondere mai mare (vezi Tab. 4.2).

După calculul ponderii, aceasta va fi folosită pentru a calcula un *loss ponderat* pe baza căruia modelul va fi penalizat mult mai tare dacă clasifică greșit un articol ce face parte dintr-o clasă slab reprezentată. Formula pentru calculul ponderii unei clase este :

$$Pondere(C) = \frac{1}{Nu(C)} * \frac{Dimensiune Set de Date}{Numar Clase} \quad (4.1)$$

unde :

$$Nu(C) = Dimensiune Set de Date - Numar Elemente din Clasa C \quad (4.2)$$

O altă problemă cauzată de class-imbalance este legată de scăderea relevanței unor metrici. Acuratețea, spre exemplu, devine mai puțin relevantă deoarece având peste 60% reprezentanți dintr-o singură clasă, în cazul nostru, un model aleator care prezice doar clasa "Real" ar obține peste 60% acuratețe. Acest lucru ne obligă să folosim alte metrici precum Loss-ul sau Acuratețea ponderată.

Evaluarea modelelor

După antrenarea modelelor a urmat evaluarea lor pentru a determina dacă rezultatele se ridică la așteptările noastre. În această etapă au fost evaluate modelele pe baza metricilor și a fost construită *matricea de confuzie* (confusion matrix) pe baza căreia am determinat unde face fiecare model greșeli în predicții. Performanțele obținute vor fi descrise mai explicit în capitolul cu rezultate.

4.3 Integrarea într-un API

După derularea experimentelor, am ales modelul cu cele mai bune rezultate, model pe care l-am integrat în cadrul unui API pentru a facilita accesul la el și pentru a-i permite oricui să îl folosească în cadrul unor proiecte personale. Pe lângă asta ne-am gândit să adăugăm funcționalități pentru dezvoltarea setului de date ca pe viitor să putem produce modele cu rezultate și mai bune.

Roluri si niveluri de acces

Pentru a putea controla accesul la API am decis să introducem ideea de roluri:

- **Utilizatorul simplu** - *Nu necesită un cont.* Tot ce poate să facă este să folosească API-ul pentru a clasifica articole de știri (fie pe baza conținutului, fie pe baza scurtei prezentări). Își poate face un cont de contribuitor dacă dorește să contribuie la dezvoltarea setului de date.
- **Contribuitorul** - *Necesită un cont de contribuitor.* Poate propune articole de știri etichetate, împreună cu motivul pentru care un articol a fost etichetat într-un anumit fel. Poate aplica pentru rolul de moderator.
- **Moderatorul** - *Necesită un cont de moderator.* Poate accepta sau respinge articolele propuse de către contribuitori. În cazul în care sunt acceptate articolele, acestea sunt adăugate în setului de date prezentat în subcapitolul anterior, iar altfel sunt eliminate din sistem.
- **Administratorul** - *Necesită un cont de administrator.* Poate promova un contribuitor la rolul de moderator, în urma unui interviu avut în prealabil.

Endpointuri care nu necesită autentificare

Clasificare articol de știri	
Metoda	POST
URL	api/model/predict
Parametrii	1. headline (<i>string</i>) - scurta prezentare 2. content (<i>string</i>) - conținutul articolului 3. headline_mode (<i>bool</i>) - modul de clasificare
Răspuns	1. label (<i>string</i>) - eticheta prezisă după clasificare 2. confidence (<i>float</i>) - nivelul de încredere în predicție

Prin intermediul endpoint-ului de clasificare se poate folosi modelul integrat în API pentru a prezice clasa de apartenență a unui articol, fie doar pe baza headline-ului (*mode = True*) fie pe baza conținutului (*mode = False*).

Pe lângă clasa prezisă, este returnat și nivelul de încredere în procente unde 100% reprezintă încredere deplină și 0% reprezintă lipsa încrederii.

Logare în sistem	
Metoda	POST
URL	api/accounts/login
Parametrii	1. email (<i>string</i>) - adresa de email 2. password (<i>string</i>) - parola
Răspuns	1. token (<i>string</i>) - token Jwt pentru autentificare

Prin intermediul endpoint-ului de autentificare utilizatorii se loghează în sistem. În urma verificării credențialelor, se trimite înapoi un token *Jwt* ce va fi folosit în headerul de autentificare pentru a putea folosi endpointurile restricționate.

Înscriere în sistem	
Metoda	POST
URL	api/accounts/signup
Parametrii	1. email (<i>string</i>) - email asociat 2. username (<i>string</i>) - numele de utilizator 3. password (<i>string</i>) - parola
Răspuns	-

Prin intermediul endpoint-ului de înscriere, un utilizator fără cont își poate face un cont de contribuitor pentru a propune articole. De menționat este faptul că **doar** conturile de contribuitor pot fi create folosind acest endpoint.

Endpointuri care necesită autentificare

Propunere articol		C
Metoda	POST	
URL	api/data/add	
Parametrii	1. headline (<i>string</i>) - scurta prezentare 2. content (<i>string</i>) - conținutul articolului 3. source (<i>string</i>) - link spre sursă 4. label (<i>string</i>) - eticheta propusă 5. motivation (<i>string</i>) - motivarea alegerii	
Headere	x-access-token: <token>	
Răspuns	-	

Prin intermediul endpoint-ului de propunere de articole, contribuitorii pot propune articole de știri etichetate. Pentru a crește calitatea datelor trimise, contribuitorii

rii sunt obligați să își motiveze alegerile legate de etichetarea propusă pentru articolul trimis. Pentru a menține formatul setului de date initial, cei care contribuie sunt obligați ca pe lângă conținutul articolului să trimită și scurta prezentare a acestuia. Sursa articolului este și ea necesară deoarece va fi folosită de către moderatorii în analiza propunerii.

Propunerile primite de la contribuitori sunt sortate în ordinea datelor la care au fost trimise urmând ca moderatorii să le prelucereze *de la cea mai veche la cea mai nouă*. Am adăugat autentificare pentru acest endpoint deoarece fără aceasta, oricine ar putea trimite date greșite astfel îngreunând sarcina moderatorilor care ar trebui să respingă mare parte din propunerile primite.

Preluare propunere neevaluată		M
Metoda	GET	
URL	api/data/get-next	
Parametrii	-	
Headere	x-access-token: <token>	
Răspuns	1. source (<i>string</i>) - link spre sursă 2. motivation (<i>string</i>) - motivarea alegerii 3. label (<i>string</i>) - eticheta propusă	

Prin intermediul endpoint-ului de preluare, un moderator poate să obțină *sursa* articolului din următoarea propunere ce trebuie prelucrată. Pe lângă sursă, moderatorul primește *eticheta* propusă alături de *motivația* pentru alegerea acelei etichete. Sursa este folosită pentru a vizualiza articolul, motivația îl ajută pe moderator să aleagă dacă articolul propus merită introdus în setul de date sau nu iar semnalarea deciziei luate se face pe baza sursei.

Evaluare propunere		M
Metoda	POST	
URL	api/data/evaluate	
Parametrii	1. source (<i>string</i>) - sursa propunere 2. is_approved (<i>bool</i>) - rezultatul evaluării	
Headere	x-access-token: <token>	
Răspuns	-	

Prin intermediul endpoint-ului de evaluare, moderatorul evaluează propunerea pe baza *sursei* primite în momentul în care a preluat-o. Acesta fie acceptă propunerea (adaugă articolul și eticheta în setul de date) fie o respinge (caz în care propunerea este eliminată din sistem). De menționat este că pentru a deveni moderator trebuie obținută o promovare de la un administrator.

Promovare A	
Metoda	POST
URL	api/accounts/promote
Parametrii	1. email (<i>string</i>) - adresa de email
Headere	x-access-token: <token>
Răspuns	-

Ca administrator, pe baza unui *nume de utilizator*, se poate face promovarea de la un cont de contribuitor la un cont de moderator. Promovarea implică renunțarea la partea de propunere de articole pentru a evita cazul în care un moderator și-ar accepta propriile propuneri.

Testare endpointuri

Pentru testarea endpoint-urilor am folosit **Postman**. Fiecare endpoint a fost testat individual pentru a verifica funcționarea API-ului în condiții optime.

▼ POST	Clasificare	2 0
	Pass Clasificare pe baza de continut.	
	Pass Clasificare pe baza de prezentare.	
▼ POST	Logare	1 0
	Pass Logare cu credentiale valide.	
▼ POST	Inscriere	1 0
	Pass Inscriere utilizator nou.	
▼ POST	Propunere	1 0
	Pass Propunere cu toate detaliile.	
▼ GET	Preluare	1 0
	Pass Preluare propunere.	
▼ POST	Evaluare	2 0
	Pass Acceptare propunere.	
	Pass Respingere propunere.	
▼ POST	Promovare	1 0
	Pass Promovare contribuitor.	

Figura 4.7: Rezultatele testării endpointurilor

Detaliile legate de arhitectura și implementarea API-ului vor fi discutate în capitolul următor pentru a se vedea mai bine interacțiunea cu clienții.

Capitolul 5

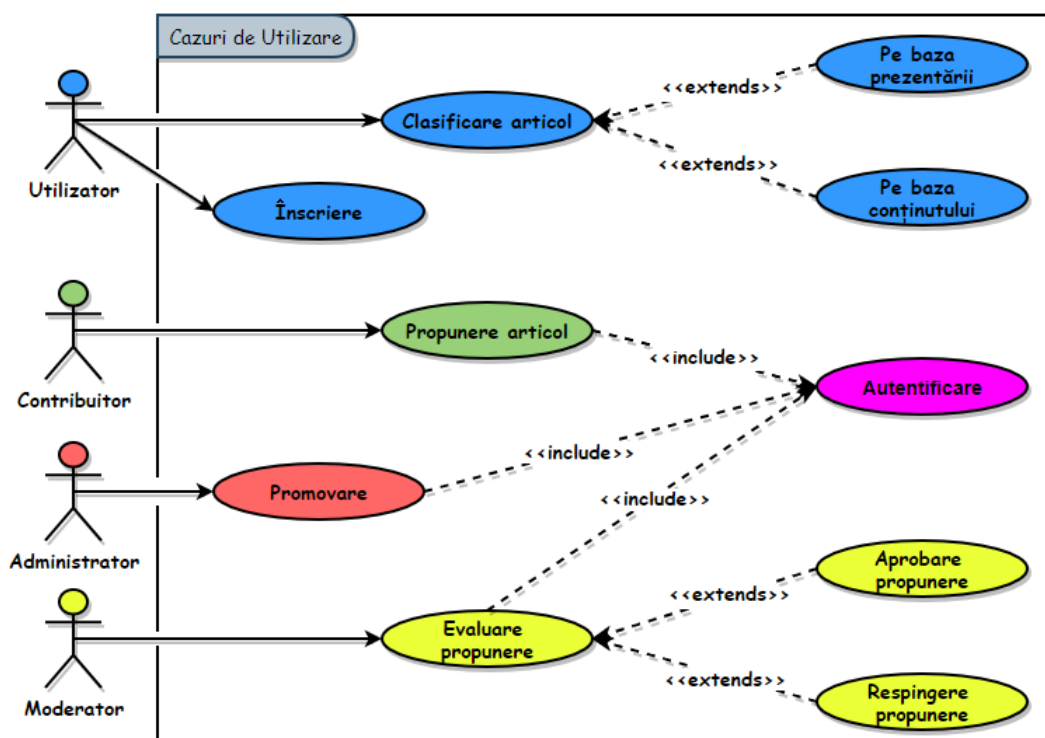
Aplicația propusă

Acum ca avem un API complet funcțional, următoarea întrebare este la ce îl putem folosi mai exact. Pentru a răspunde la această întrebare vă propun o aplicație, prin intermediul căreia vom folosi funcționalitățile oferite de API pentru a combate știrile false din mediul online românesc.

5.1 Arhitectura aplicației

5.1.1 Comportament

Comportamentul aplicației poate fi descris pe baza următoarei diagrame unde am ilustrat cazurile de utilizare.

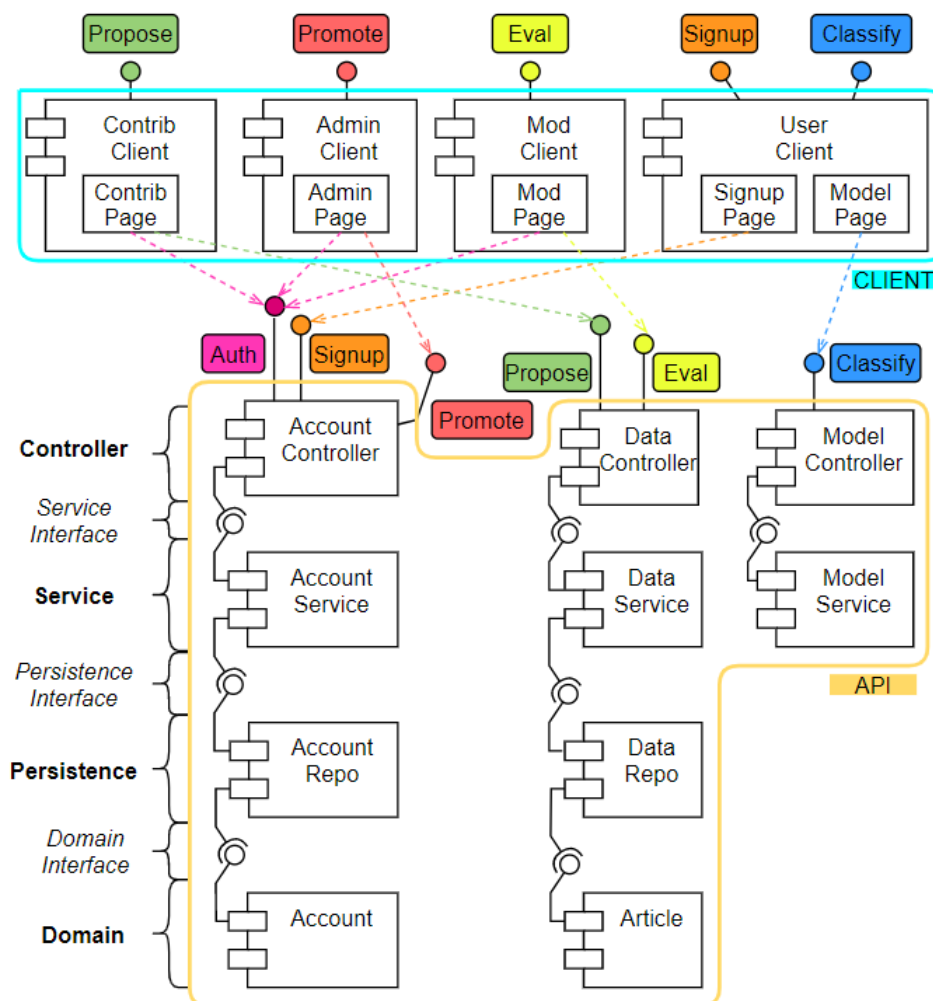


În diagramă se observă patru actori și zece cazuri de utilizare (cinci principale și cinci secundare). Legătura dintre un actor și un caz de utilizare reprezintă faptul că actorul participă la acel caz de utilizare.

Când un caz de utilizare este descris ca folosind funcționalitatea unui alt caz de utilizare, relația dintre cele două este denumită relație *'include'*. Relația *'extends'* descrie faptul că un caz de utilizare poate include comportamentul specificat în cazul de utilizare de baza (cel dinspre care indică săgeata).

5.1.2 Structură

Structura aplicației poate fi descrisă pe baza următoarei diagrame de componente.



În cadrul diagramei se poate observa folosirea unei *arhitecturi stratificate*. Prin folosirea acestui tip de arhitectură, ne-am asigurat că dependențele între componente merg într-o singură direcție (în cazul nostru de sus în jos). Interfețele folosite între layere sunt și ele utile deoarece oferă contracte pe care obiectele le pot folosi pentru a lucra împreună, fără a fi nevoie să știe nimic unul despre celalalt. Pe lângă asta, am împărțit vertical API-ul în trei segmente și anume *Account*, *Data* și *Model*.

Fiecare dintre cele trei segmente a fost proiectat pentru a se ocupa de o anumită sarcină. Ramura *Account* se ocupă de gestionarea conturilor și a operațiilor ce țin de acestea, pe ramura *Data* are loc întregul proces de colectare și evaluare a datelor iar prin intermediul ramurii *Model* se oferă acces la modelul pentru clasificare de articole antrenat în capitolul anterior.

5.1.3 Legătura dintre comportament și structură

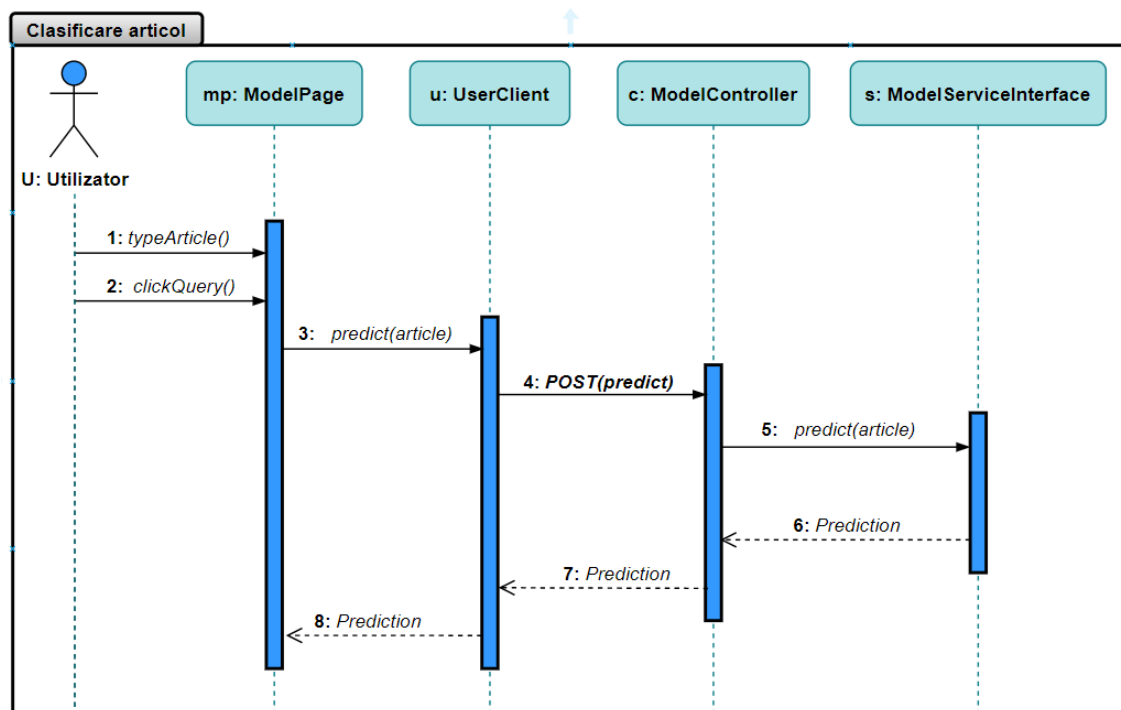
Pentru a prezenta legătura dintre comportamentul aplicației și structura acesteia voi descrie derularea fiecărui caz de utilizare, folosind diagrame de secvență. Prin intermediul acestor diagrame, vom putea observa în detaliu modul în care interacționează componentele din diagrama de componente.

Clasificare articol

Participant: *Utilizator*

Flux de evenimente:

1. Utilizatorul deschide aplicația și intră pe pagina cu modelul.
2. Completează câmpul articol cu conținutul unui articol de știri.
3. Apasă butonul pentru clasificare și observă rezultatele produse.



Utilizatorul trebuie fie să copieze conținutul unui articol, fie să îl scrie de mână. Sistemul funcționează în ambele cazuri deoarece dispune de un subsistem de curățare de text care poate elimina spațiile albe copiate folosind combinația *copy-paste*. Pentru

a avea rezultate cât mai bune, este la latitudinea utilizatorului să se asigure că ceea ce copiază reprezintă doar conținutul articolului și nu include alte corpuri textuale precum sunt comentariile, recenziile, etc.

În caz de succes, răspunsul primit de la aplicație va ilustra eticheta atribuită de model împreună cu nivelul de încredere în etichetarea propusă. Pentru cazurile de eroare, vor fi afișate mesaje de eroare intuitive pentru ca utilizatorul să își dea seama ușor ce face greșit.

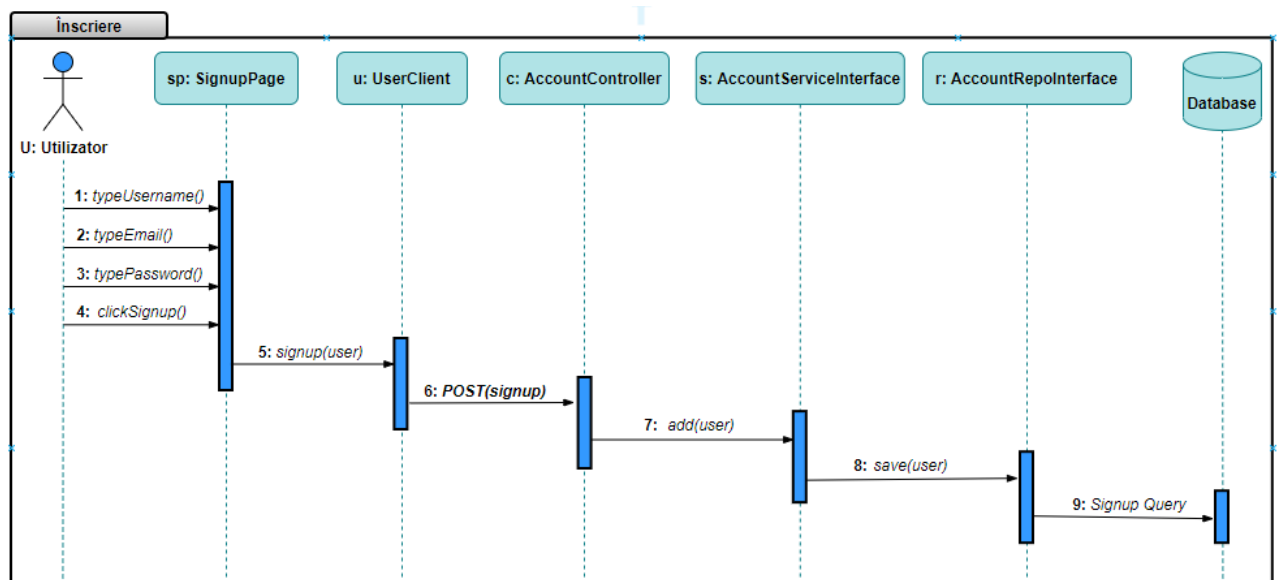
Dacă utilizatorul nu cunoaște semnificația etichetelor asociate de modelul de clasificare, acesta poate accesa meniul 'Help' unde este prezentată în detaliu semnificația fiecărei etichete și cum trebuie interpretat nivelul de încredere în etichetare.

Înscriere

Participant: *Utilizator*

Flux de evenimente:

1. *Utilizatorul deschide aplicația și intră pe pagina pentru înscrieri.*
2. *Completează câmpurile din formular (nume de utilizator, email și parolă).*
3. *Apasă butonul pentru înscriere și observă rezultatul operației pe ecran.*



Utilizatorul trebuie să se asigure că nu are un alt cont creat pe adresa de email completată în formular. În cazul în care exista deja un cont creat pe această adresă, un mesaj de eroare legat de acest lucru va fi afișat pe ecran. Chiar dacă nu este prezentat în diagramă, utilizatorul va trebui să își confirme parola aleasă pentru a preveni situația în care acesta s-ar înscrie și nu ar reține parola.

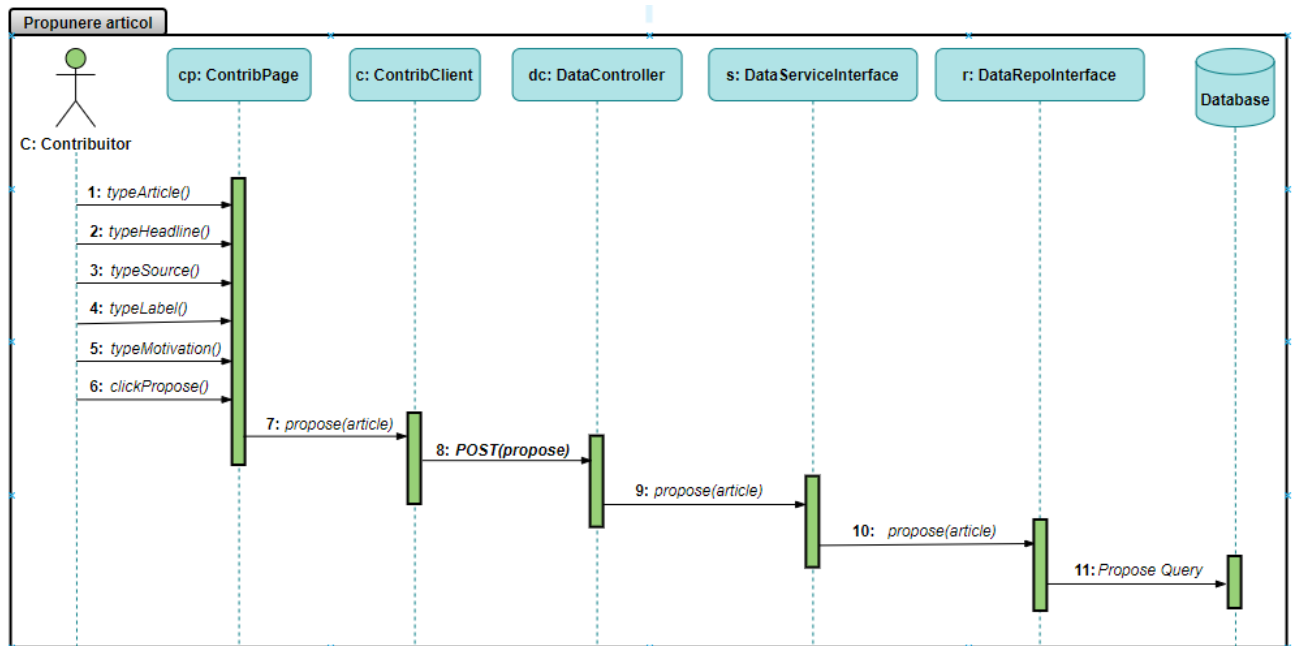
De menționat este și faptul că, după înscriere, contul creat va fi unul de contributor, permițându-i utilizatorului doar să încarce articole noi în aplicație.

Propunere articol

Participant: Contributor

Flux de evenimente:

1. Contributorul deschide aplicația și se autentifică.
2. Completează formularul cu datele legate de articolul analizat.
3. Apasă butonul pentru propunerea articolului și observă rezultatul operației pe ecran.



Pentru a evita propunerea de mai multe ori a unui articol (în speranța că așa va fi acceptat), am implementat un sistem care verifică dacă sursa propusă există deja în lista cu propunerile neevaluate. În cazul în care sursa articolului se află deja acolo, utilizatorul va fi notificat printr-un mesaj de eroare că nu poate propune etichete pentru un articol care este deja în curs de procesare.

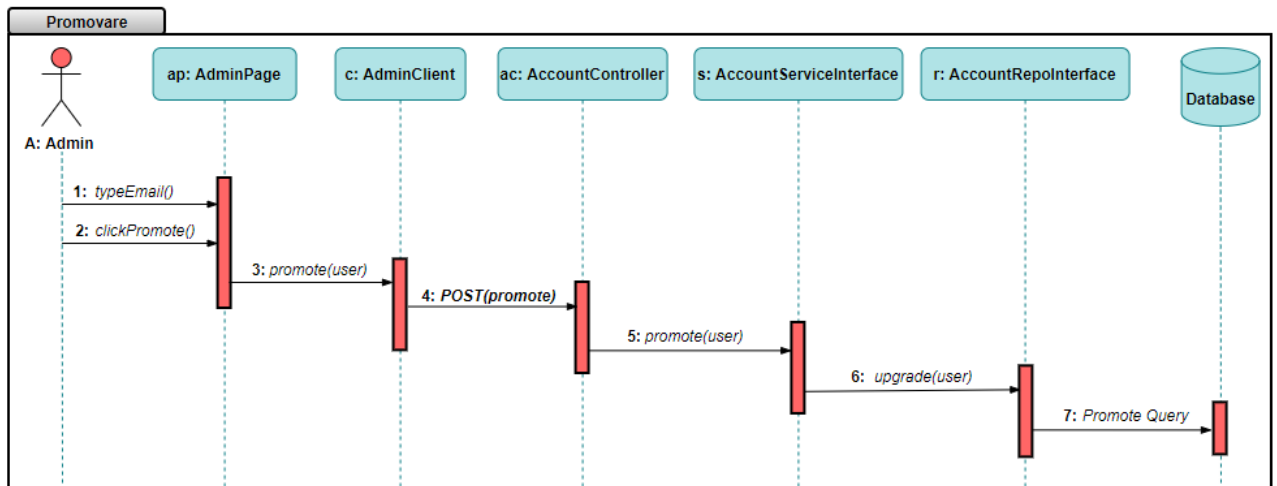
Pe lângă asta, este obligatoriu ca fiecare dintre câmpurile din formular să fie completate de contributor, în special cel legat de motivație unde ar trebui explicat în detaliu de ce a fost considerată o anumită etichetă pentru articolul trimis.

Promovare

Participant: Administrator

Flux de evenimente:

1. Administratorul deschide aplicația și se autentifică.
2. Completează email-ul contributorului ce va fi promovat.
3. Îl promovează la rolul de moderator apăsând butonul de promovare.



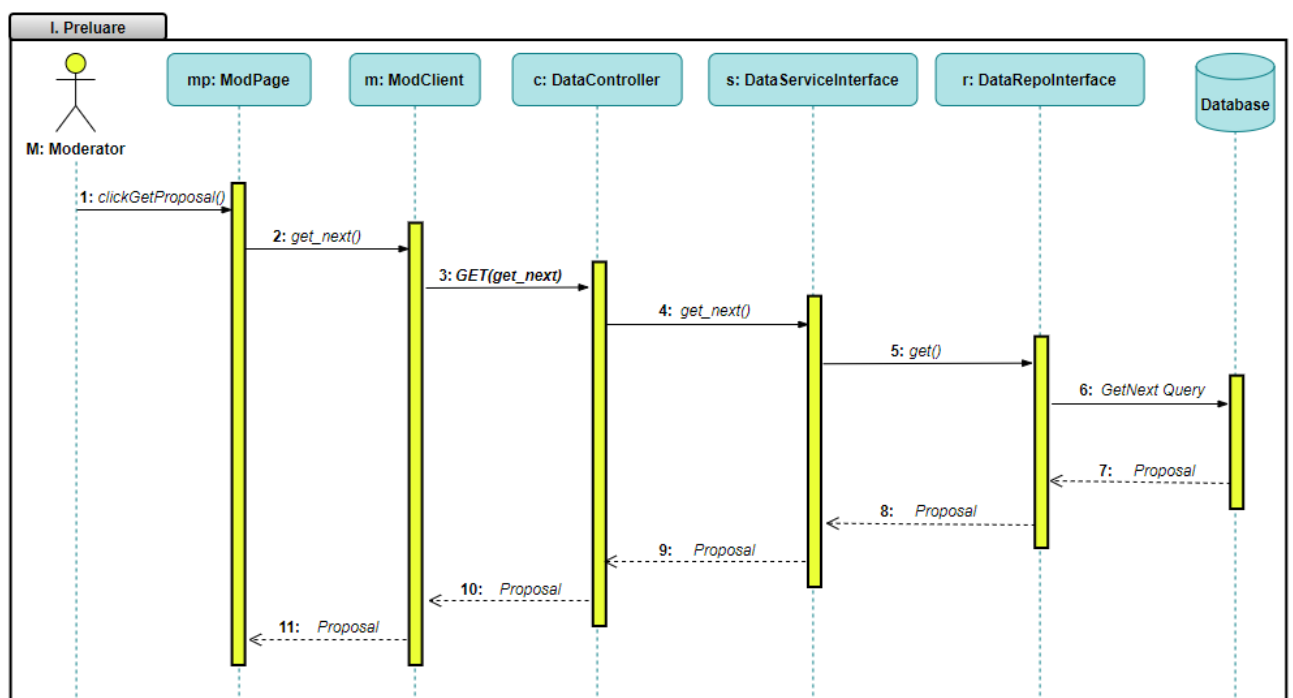
Acesta este singurul mod prin care un contribuitor poate deveni moderator. În momentul în care un contribuitor este promovat la rol de moderator pierde dreptul la a propune articole.

Evaluare propunere

Participant: *Moderator*

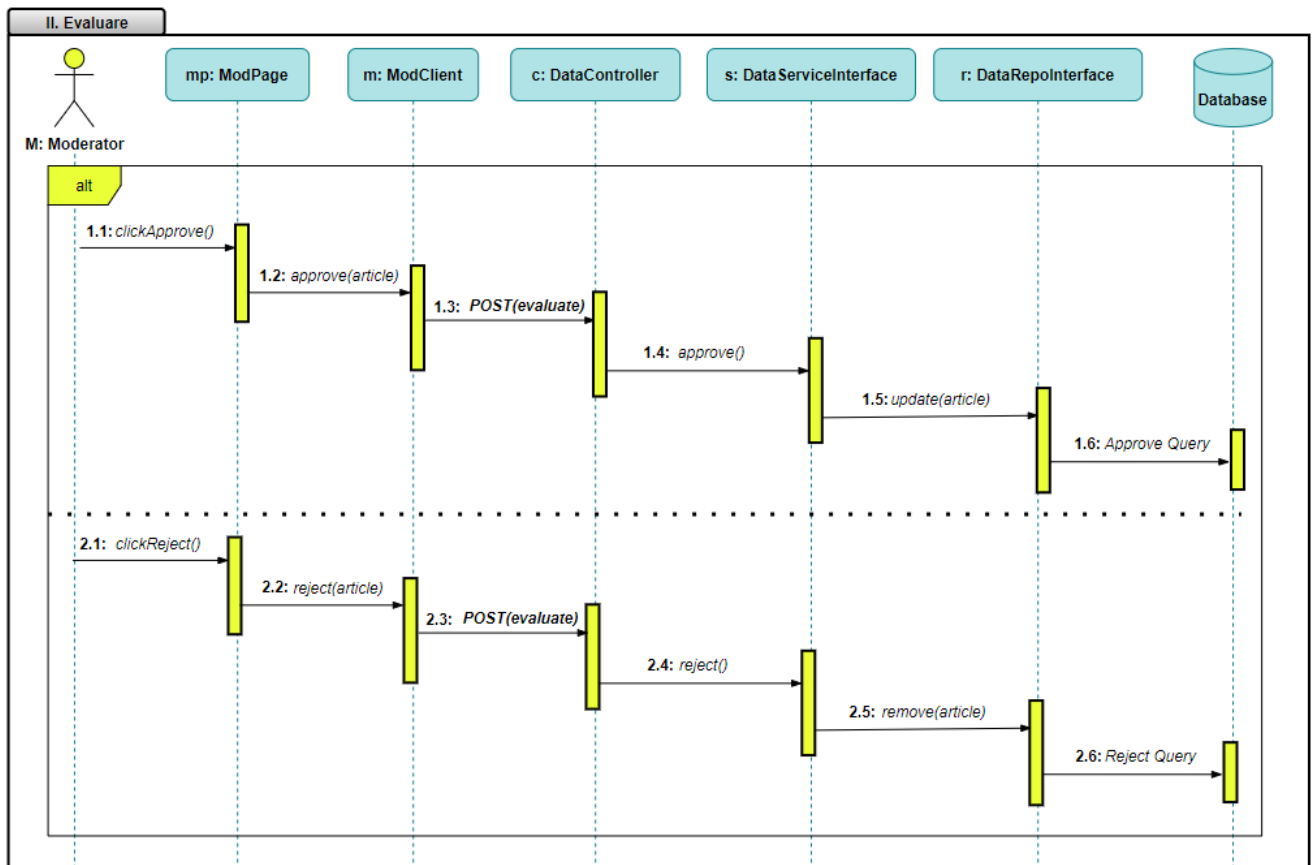
Flux de evenimente:

1. Moderatorul deschide aplicația și se autentifică.
2. Preia o propunere neevaluată.
3. Evaluează propunerea.



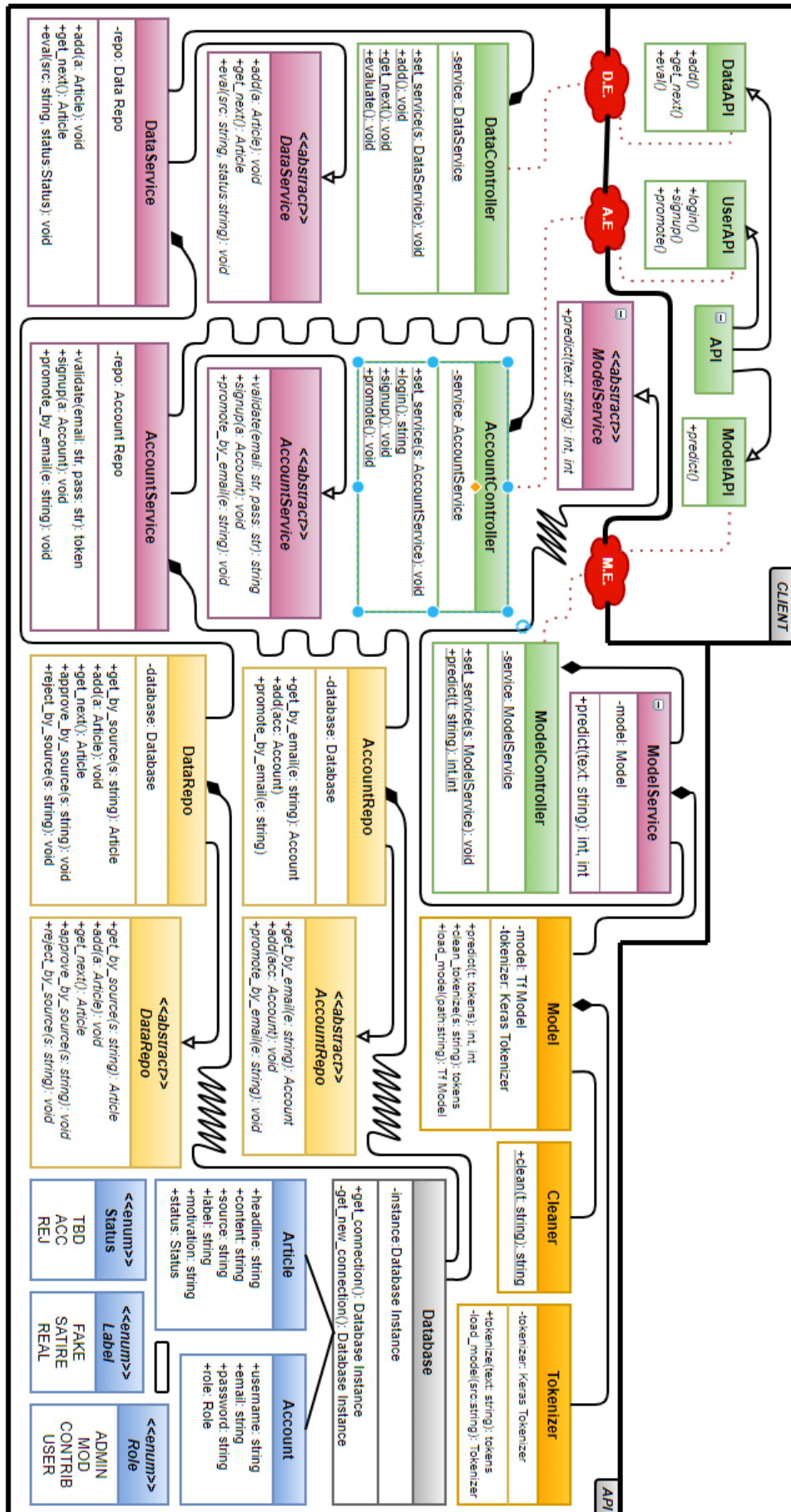
În pasul de preluare, în urma apăsării unui buton, moderatorul primește detalii legate de cea mai veche propunere neevaluată din baza de date. Detaliile primite sunt eticheta propusă și motivația iar pe lângă acestea, ii se va deschide sursa articolului într-un nou tab în browser, pentru a-i ușura treaba.

Moderatorul va respinge toate propunerile a căror sursa nu se deschide așadar contribuitorii trebuie să ofere o atenție sporită la ce sursă completează în formularul pentru propunere. Pe lângă asta, contribuitorii ar trebui să ofere o motivație coerentă, fără ambiguități sau erori logice, dacă doresc să aiba șanse mai mari ca articolul propus de ei să fie acceptat.



După preluare urmează pasul de evaluare a propunerii, moment în care moderatorul citește și analizează conținutul articolului de știri, verificând și motivația și eticheta oferită în cadrul propunerii. Dacă moderatorul o consideră o propunere bună, acesta o aprobă folosind butonul de aprobare iar altfel o respinge folosind butonul pentru respingere.

Articolele aprobate vor fi păstrate în baza de date spre a fi adăugate ulterior în setul de date iar cele respinse vor fi eliminate din sistem. Moderatorul poate fi considerat cel mai important dintre rolurile din aplicație deoarece el reprezintă filtrul ce asigură calitatea setului de date.



5.2 Implementarea aplicației

5.2.1 Diagrama de clase

În cele ce urmează, o să prezint modul în care am implementat aplicația, prin intermediul diagramei de clase de pe pagina anterioară. În diagramă, clasele au fost colorate în funcție de pachetul din care fac parte așadar avem următoarele pachete:

- **Domain** - reprezentat cu *albastru*.
- **Persistence** - reprezentat cu *galben*.
- **Service** - reprezentat cu *mov*.
- **Controller** - reprezentat cu *verde*.
- **Model** - reprezentat cu *portocaliu*.

Domain

Aici avem entitățile din domeniu și anume **Account** care reprezintă datele legate de contul unui utilizator și **Article** care reprezintă datele legate de un articol. Pe lângă entități, mai avem și obiecte de tip *enum* și anume **Status** care prezintă stările în care se poate afla un articol (*propus*, *acceptat* sau *respins*) **Label** care reprezintă etichetele pe care le poate atribui modelul unui articol (*fals*, *real* sau *satiră*) și **Role** care reprezintă diferitele tipuri de conturi (*admin*, *moderator*, *contribuitor* sau *utilizator*).

Persistence

Aici avem clasele care se ocupă cu *persistarea datelor*. Prin intermediul acestui strat am izolat stratul de servicii de sistemul de gestionare a bazelor de date folosit. În cadrul stratului avem câte un repository pentru fiecare entitate din domeniu. Un repository are în componența sa un obiect de tip **Database** care reprezintă instanța unei baze de date. Folosind aceasta instanță, fiecare repo poate executa asupra bazei de date comenzi și interogări, conform cererilor primite din service.

Service

În pachetul cu servicii avem clase care se ocupă de executarea diferitelor sarcini în funcție de funcționalitățile folosite de client. Avem service pentru gestionarea conturilor, service pentru gestionarea articolelor și un service pentru a lucra cu modelul. Service-urile ce folosesc persistență au în componență câte un repo prin intermediul căruia preiau și trimit date din și în baza de date.

Controller

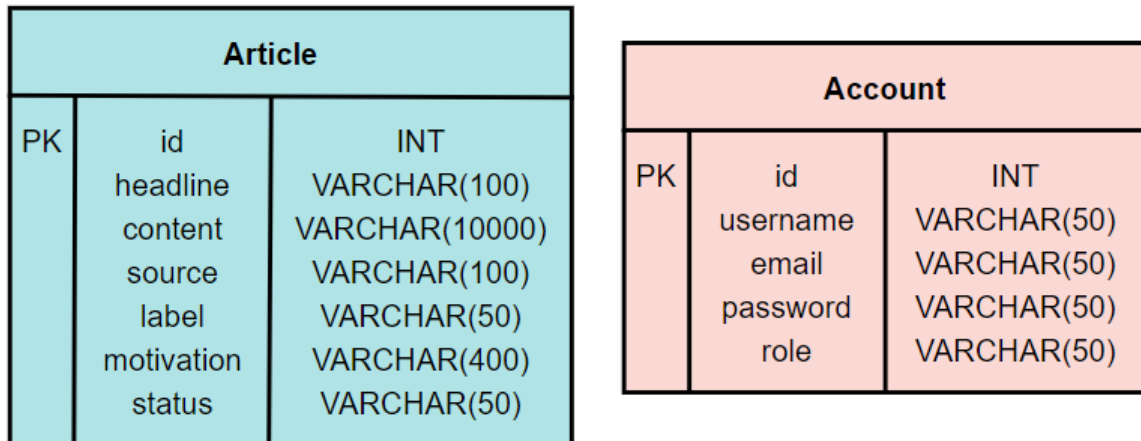
În controller avem adaptoare pentru API care în funcție de tipul cererii HTTP primite de la client, redirecționează cererea spre service-ul care se ocupă de aceasta. Tot în controller se verifică dacă utilizatorii au permisiunea să acceseze funcționalitățile ce necesită autentificare.

Model

În acest pachet avem mai multe clase utilitare folosite pentru a putea integra un model *Tensorflow* în API. Clasa **Tokenizer** se ocupă de tokenizare folosind un model de tokenizator *Keras* pe care îl încarcă dintr-un fisier. Clasa **Cleaner** se ocupă de preprocesare și curățare de date iar clasa **Model** se ocupă de clasificarea de articole folosind modelul încărcat în prealabil dintr-un fisier.

5.2.2 Baza de date

În următorul fragment voi prezenta structura bazei de date folosite, prin intermediul unei diagrame de bază de date, urmând ca ulterior să discut de ce am ales să o proiectez în modul prezentat în diagramă.



După cum se observă în diagramă, în baza de date propusă nu există relații între entități. Am fi putut asocia articolele din baza de date cu utilizatorul care le-a propus dar am decis să nu facem acest lucru pentru a păstra confidențialitatea legată de ce articole urmărește fiecare utilizator.

Eliminând legătura dintre propunător și articol, fiecare articol este propus anonim, astfel sperând că vom atrage mai multi contribuitori. Pe lângă beneficiul atragerii mai multor contribuitori mai există și o parte negativă legată de calitatea propunerilor. Având un sistem anonim, oricine poate să își creeze un cont și să propună orice îi trece prin cap doar cu scopul de a îngreuna sarcina moderatorilor.

5.2.3 Tehnologiile folosite

Frontend

- **Javascript** - este un limbaj de programare orientat obiect folosit în dezvoltarea de pagini web interactive.
- **Typescript** - este un superset sintactic al limbajului JavaScript, proiectat pentru dezvoltarea de aplicații de dimensiuni mai mari.
- **Ionic-React** - este un kit de dezvoltare open source pentru dezvoltarea de aplicații mobile hibride. L-am ales deoarece am dorit să dezvolt o aplicație care să funcționeze și pe mobil și în browser.
- **Axios** - este un client HTTP ce funcționează pe bază de promisiuni.

Backend

- **Python** - este un limbaj de programare cu scop general, folosit într-o gamă largă de aplicații. L-am ales deoarece oferă un set de librării foarte utile în dezvoltarea de sisteme inteligente.
- **Flask** - este un micro framework web scris în Python. L-am ales pentru simplitatea configurării și utilizării.
- **SQLite** - reprezintă o librerie prin intermediul căreia se pot crea și gestiona baze de date. Am ales SQLite deoarece aveam nevoie de o bază de date simplă și ușor de folosit.

Pe lângă tehnologiile folosite în dezvoltarea aplicației, am folosit librăriile **tensorflow**, **numpy**, **pandas**, **scikit-learn**, **seaborn** și **matplotlib** în procesul de pregătire al modelelor prezentate anterior. Librăriile tensorflow și numpy au fost folosite pentru a dezvolta componentele modelului, pandas pentru a lucra cu datele și pentru a genera statistici, scikit-learn pentru împartirea datelor și evaluarea modelelor pe baza unor metrice iar seaborn și matplotlib au fost folosite pentru a reprezenta vizual rezultatele obținute.

De menționat sunt și tehnologiile folosite pentru testarea aplicației. Pentru testarea modulelor am folosit librăria **PyUnit** iar pentru a testa interfața grafică a clientului am folosit **Puppeteer**.

Pentru dezvoltarea API-ului am folosit IDE-ul **Pycharm** iar pentru dezvoltarea clientului am folosit **Visual Studio Code**. Pentru antrenarea modelelor, am folosit platforma **Google Colab**.

5.3 Manual de utilizare

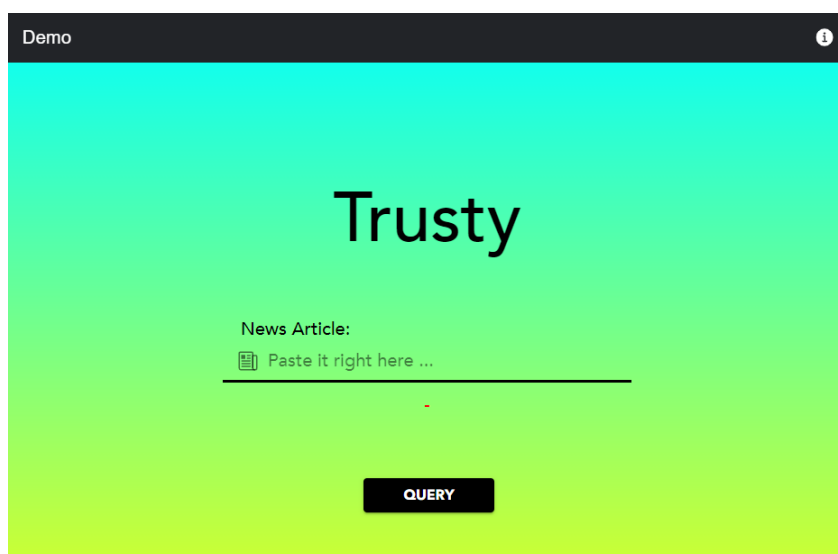
Pagina de pornire

La momentul accesării site-ului web, utilizatorul este direcționat spre pagina de pornire. Apăsând butonul **i** din dreapta sus, utilizatorul poate afla mai multe informații despre pagina pe care se află.



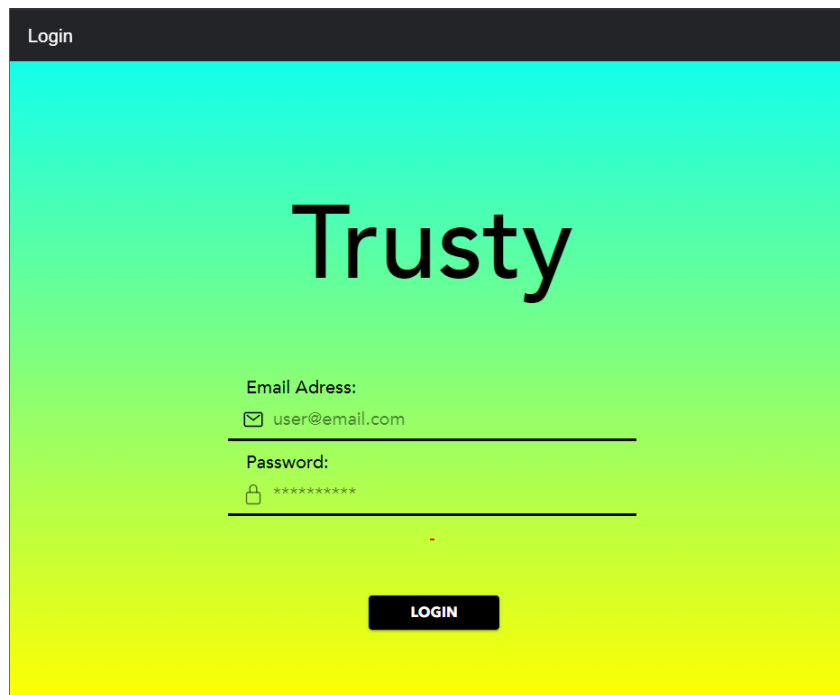
Din pagina de pornire se pot accesa diferite funcționalități ale aplicației. Dacă utilizatorul dorește să folosească modelul de clasificare, acesta va apăsa pe butonul **Try it out** și va fi redirecționat către pagina de demo. În cazul în care utilizatorul dorește să devină un contribuitor, acesta va apăsa butonul **Contribute** și va fi redirecționat spre pagina de înscriere. Dacă utilizatorul are deja cont, acesta poate apăsa pe prima iconiță din bara de sus pentru a accesa pagina de autentificare.

Pagina de demo



În pagina de demo, utilizatorul poate verifica diferite articole de stiri pe baza modelului de clasificare. Pentru a clasifica un articol acesta trebuie copiat de la sursa și lipit în caseta din cadrul paginii. Pentru a interoga modelul se va apăsa butonul **Query**. După analizarea articolului, modelul returnează un răspuns care este afișat pe ecran. Pentru informații legate de etichetele atribuite de model și despre cum ar trebui interpretat un nivel de încredere, utilizatorul poate apăsa butonul **i**.

Pagina de autentificare



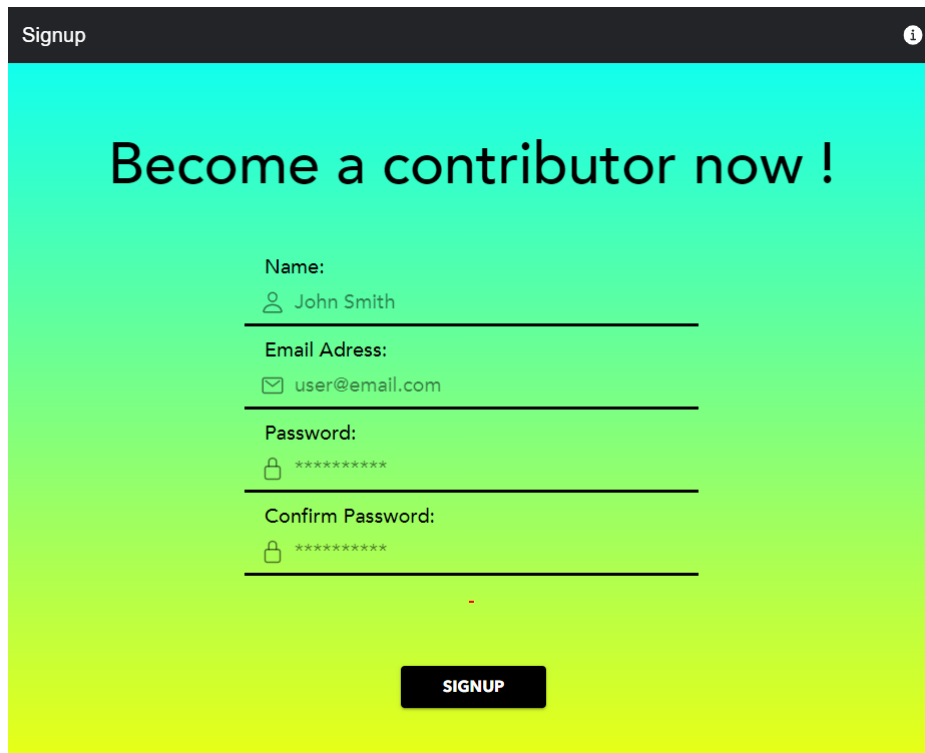
Pentru a se autentifica, utilizatorul trebuie să își introducă credențialele și să apese butonul **Login**. În cazul în care credențialele sunt valide, utilizatorul va fi redirectionat spre una din paginile admin, moderator sau contributor, în funcție de rol. În cazul în care credențialele sunt invalide, aplicația va afișa mesaje de eroare.

Pagina de înscriere

Prin intermediul paginii de înscriere, utilizatorii își pot crea un cont de contributor. Pentru a afla mai multe detalii legate de ce anume face un contributor, utilizatorii pot da click pe butonul **i**.

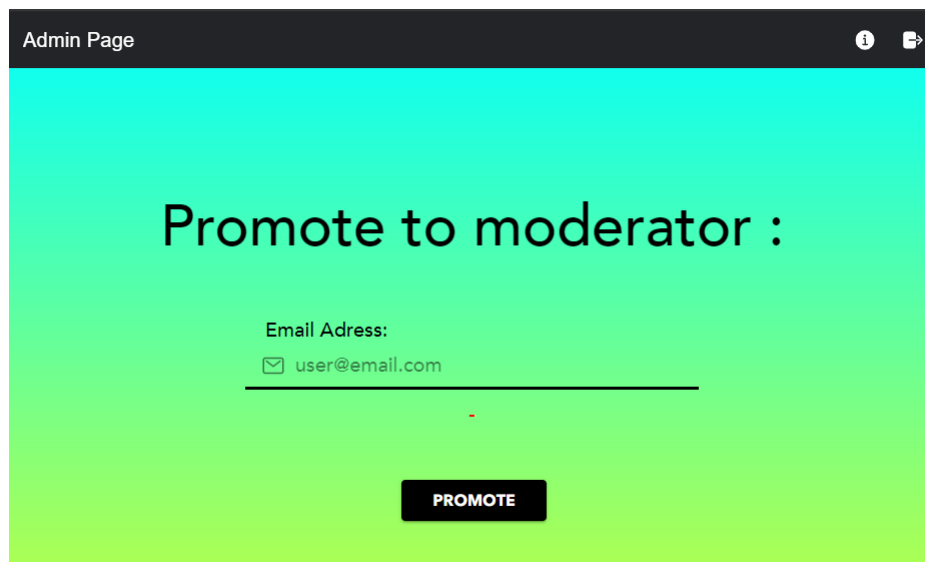
Informațiile necesare pentru a crea un cont nou sunt numele utilizatorului, o adresă de email validă și o parolă. Pentru a ne asigura de parola aleasă, utilizatorul este nevoit să o introducă de două ori. După completarea datelor, utilizatorul va da click pe butonul **Signup** pentru a se înscrie.

Dacă înscrierea are succes, utilizatorul va fi redirectionat spre pagina de pornire iar în caz contrar pe ecran vor fi afișate mesaje de eroare în funcție de situație.



The screenshot shows a web application window titled "Signup". The background is a gradient from cyan at the top to yellow at the bottom. At the top right, there is a dark header bar with the text "Signup" and an information icon. The main heading in the center is "Become a contributor now !". Below this, there are four form fields, each with a label and an icon: "Name:" with a person icon, "Email Address:" with an envelope icon, "Password:" with a lock icon, and "Confirm Password:" with a lock icon. The input fields contain "John Smith", "user@email.com", "*****", and "*****" respectively. A red asterisk is visible below the "Confirm Password" field. At the bottom center, there is a dark button with the text "SIGNUP".

Pagina de administrator



The screenshot shows a web application window titled "Admin Page". The background is a gradient from cyan at the top to yellow at the bottom. At the top right, there is a dark header bar with the text "Admin Page", an information icon, and a logout icon. The main heading in the center is "Promote to moderator :". Below this, there is one form field with the label "Email Address:" and an envelope icon. The input field contains "user@email.com". A red asterisk is visible below the input field. At the bottom center, there is a dark button with the text "PROMOTE".

După autentificare, administratorul poate folosi funcția de promovare inserând adresa de email a contribuitorului pe care dorește să îl promoveze. După introducerea adresei, se apasă butonul **Promote** pentru a promova contribuitorul. În urma apăsării butonului, pe ecran va apărea fie un mesaj de succes, fie un mesaj în care este descrisă eroarea.

Dacă administratorul dorește să iasă din aplicație, acesta trebuie să apese butonul din dreapta sus pentru a se deloga. În cazul în care dorește alte informații legate de funcționalitatea de promovare, administratorul poate apăsa butonul i.

Pagina de contributor

Contributor Page

Contribute an article !

Headline:
Article headline ...

Content:
Article content ...

Source:
www.article.com

Label:
Fake/Real/Satire

Motivation:
Why did you choose that label ?

PROPOSE

După autentificare, contributorii sunt redirecționați spre această pagină, prin intermediul căreia pot propune articole. Pentru a propune un articol, contributorii trebuie să ofere detalii legate de conținutul articolului, sursa de unde a fost extras, eticheta asociată și motivația pentru care au ales acea etichetă.

După completarea tuturor datelor necesare, se va apăsa butonul **Propose** pentru a trimite propunerea spre serverul aplicației. Dacă contributorul dorește să afle informații legate de fiecare câmp cerut în formular, poate da click pe butonul **i**.

Pagina de moderator

Moderator Page

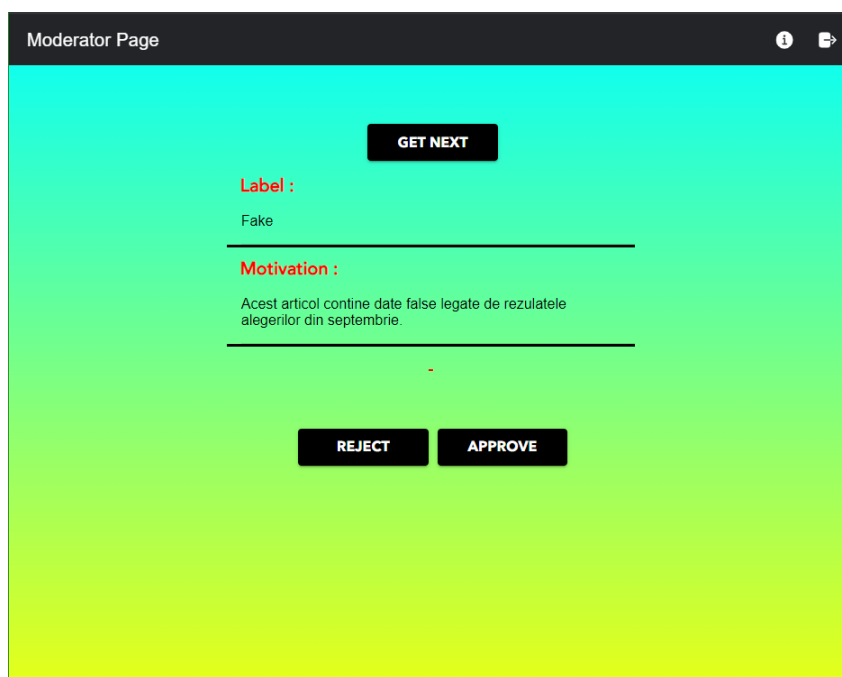
GET NEXT

Label :

Motivation :

REJECT APPROVE

După autentificare, moderatorii sunt redirecționați spre această pagină unde pot evalua propunerile venite de la contribuitori. Pentru a primi următoarea propunere de evaluat, moderatorul trebuie să apese pe **Get Next**.



După preluarea propunerii, moderatorul poate începe procesul de evaluare. Pe lângă etichetă și motivație, aplicația deschide și o pagină nouă unde moderatorul poate să vadă articolul în forma originală.

În urma unei analize în detaliu, moderatorul poate fie să aprobe propunerea sau să o respingă. Pentru aprobare apasă pe butonul **Approve** iar pentru respingere pe **Reject**. În cazul în care uită ce trebuie să ia în considerare în momentul în care evaluează o propunere, moderatorul poate da click pe butonul **i** pentru a-și reaminti pașii din procesul de evaluare. Dacă dorește să iasă din aplicație, delogarea este identică cu cea de la administrator sau cea de la contributor.

Din moment ce aplicația a fost dezvoltată folosind *Ionic*, aceasta poate fi folosită și ca aplicație de mobil. Interfața și modul de lucru sunt extrem de similare cu versiunea web așadar nu voi include și un manual de utilizare pentru versiunea mobilă.

Capitolul 6

Rezultate obținute

În cadrul acestui penultim capitol voi prezenta rezultatele obținute în urma antrenării în diferite ipostaze a modelelor prezentate în capitolul patru.

Ultime verificări

O problemă des întâlnită în dezvoltarea unui model este legată de distribuția claselor în cadrul subseturilor de antrenament, validare și testare. Spre exemplu, dacă în subsetul de testare avem doar clasa *Real*, măsurătorile obținute în urma evaluării modelului pe subsetul de testare nu vor fi relevante în contextul întregului set de date.

Pentru a ne asigura că rezultatele sunt relevante, trebuie să verificăm dacă distribuția claselor este aproximativ egală între cele trei subseturi. După cum se poate observa în fig. 6.1 clasele sunt distribuite aproximativ la fel în cadrul celor trei subseturi.

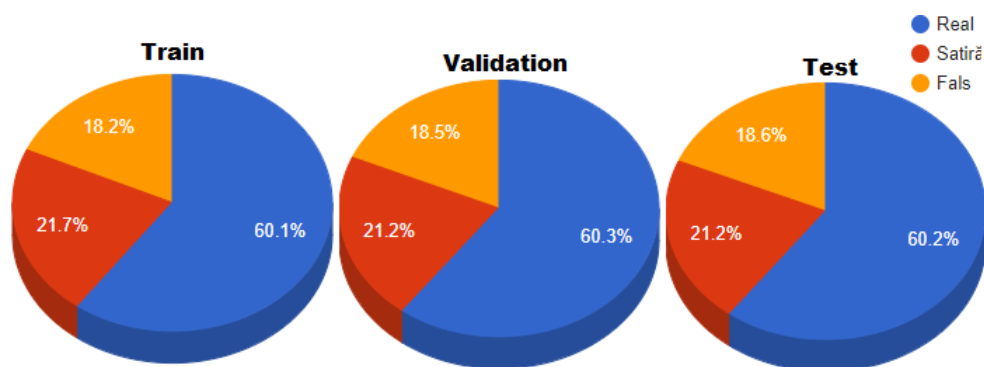


Figura 6.1: Distribuția claselor în cele trei subseturi.

De menționat este și faptul că **am evaluat modelele pe baza loss-ului**. După cum am descris anterior, în capitolul patru, acuratețea nu este foarte relevantă în evaluarea modelelor antrenate pe seturi de date cu *class-imbalance*. Chiar dacă în unele cazuri, prin continuarea antrenării a crescut acuratețea, am salvat starea modelelor în momentul în care loss-ul a fost la valoarea minimă.

Rezultate

În primul rând am decis să exclud complet eliminarea de cuvinte de legătură, propusă ca metodă de preprocesare, deoarece folosindu-o în câteva experimente a dus la rezultate destul de slabe. Pe de altă parte, eliminarea diacriticelor s-a dovedit a produce rezultate destul de interesante așadar în cele ce urmează, modelele care au fost antrenate pe text fără diacritice vor avea un caracter * după nume.

Pentru evaluarea modelelor, după cum am menționat anterior, am folosit loss-ul. Pe lângă loss am decis să folosesc și o versiune ponderată a acurateții. În tabelele ce prezintă rezultatele va exista câte o coloană pentru acuratețea obținută per clasă, urmând ca la sfârșit de tabel să fie o coloană cu acuratețea medie obținută calculând media ponderată cu ponderile din tabelul 4.2.

În următoarele două tabele sunt prezentate rezultatele obținute antrenând doar pe conținutul articolelor:

Idx.	Model	Loss	Acc. R	Acc. S	Acc. F	Acc Ponder.
1	LSTM	0.2670	93.79 %	96.96 %	72.68 %	85.27 %
2	LSTM*	0.2429	94.49 %	92.73 %	78.59 %	86.43 %
3	GRU	0.2317	94.50 %	92.79 %	78.50 %	86.42 %
4	GRU*	0.2354	95.17 %	92.94 %	77.70 %	86.20 %
5	BI-LSTM	0.2224	95.82 %	93.87 %	77.59 %	86.60 %
6	BI-LSTM*	0.2492	92.92 %	94.44 %	84.36 %	89.56 %
7	CNN	0.2026	96.09 %	94.86 %	81.17 %	88.70 %
8	CNN*	0.2386	96.82 %	94.53 %	73.50 %	85.12 %
9	CNN-LSTM	0.2479	96.77 %	97.66 %	72.19 %	85.74 %
10	CNN-LSTM*	0.2598	96.30 %	96.80 %	71.80 %	85.16 %

Tabela 6.1: Rezultate obținute pentru lungimea maximă 512.

Idx.	Model	Loss	Acc. R	Acc. S	Acc. F	Acc Ponder.
1	LSTM	0.2556	96.15 %	92.68 %	74.58 %	84.79 %
2	LSTM*	0.2455	95.91 %	97.80 %	75.23 %	87.09 %
3	GRU	0.2596	96.14 %	95.50 %	75.56 %	86.36 %
4	GRU*	0.2818	97.78 %	91.50 %	67.52 %	81.28 %
5	BI-LSTM	0.2607	93.52 %	94.67 %	81.87 %	88.58 %
6	BI-LSTM*	0.2626	95.99 %	95.12 %	77.25 %	86.97 %
7	CNN	0.2007	96.71 %	93.08 %	81.00 %	88.00 %
8	CNN*	0.2105	95.20 %	95.15 %	78.76 %	87.57 %
9	CNN-LSTM	0.2056	95.51 %	97.80 %	76.73 %	87.73 %
10	CNN-LSTM*	0.2575	97.11 %	93.40 %	72.86 %	84.41 %

Tabela 6.2: Rezultate obținute pentru lungimea maximă 1024.

Deoarece nu pot spune exact care este cel mai bun model pe caz general am decis să sortez modelele în ordinea performanțelor obținute pentru fiecare categorie și să le asignez punctaje între unu și zece, în funcție de locul pe care se clasează (zece pentru primul loc și unu pentru ultimul). La final, modelul cu cel mai mare punctaj total va fi declarat cel mai bun model pentru clasificarea de articole doar pe baza conținutului. De menționat este faptul că am ținut cont și de durata perioadei de antrenare, modelele antrenate mai rapid primind punctaj mai mare. În caz de egalitate de rezultate pentru o anumită categorie, toate modelele care erau la egalitate au primit punctaje egale.

Loc	Model	Timp	Loss 512	Loss 1024	Acc. 512	Acc. 1024	Total Puncte
1	CNN	10	10	10	9	9	48
2	BI-LSTM	8	9	3	8	10	38
3	CNN-LSTM	9	4	9	4	8	34
4	CNN*	10	6	8	1	7	32
5	LSTM*	6	5	7	7	6	31
6	GRU	7	8	4	6	4	29
7	BI-LSTM*	8	3	2	10	5	28
8	GRU*	7	7	1	5	1	21
9	LSTM	6	1	6	3	3	19
10	CNN-LSTM*	9	2	5	2	2	18

Tabela 6.3: Modelele sortate în ordinea performanței.

După cum se observă și în tabelul cu punctaje, **CNN** obține cea mai bună performanță per total, la distanță relativ mare de locurile doi și trei care sunt ocupate de **BI-LSTM** și **CNN-LSTM**. Rezultatele obținute au fost conforme cu ceea ce am presupus în 4.2.1. Pe lângă asta doresc să menționez și faptul că în cadrul experimentelor, modelele bazate pe LSTM s-au antrenat cu 30-40% mai încet decât cele bazate pe GRU, lucru ce confirmă și presupunerea legată despre timpul necesar pentru antrenare.

Pentru a motiva punctajele acordate pe timpul de antrenare, voi prezenta grafic timpii necesari pentru antrenarea fiecărui tip de model, raportați în funcție de modelul care se antrenează cel mai rapid.

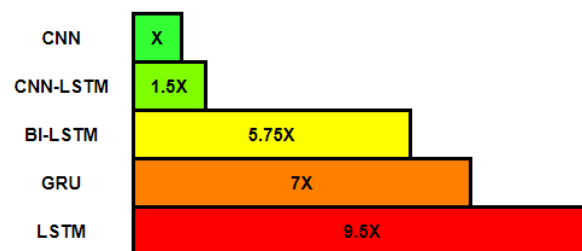


Figura 6.2: Durata antrenării modelelor în funcție de durata antrenării CNN.

În continuare voi descrie și rezultatele obținute în urma antrenării de modele care clasifică articole doar pe baza prezentării. Ca lungime maximă am ales 128 deoarece prezentarea articolului are maxim 3-4 propoziții. Rezultatele obținute sunt prezentate în cadrul tabelului următor. Pentru ordonarea modelelor după performanță am folosit aceeași abordare ca mai sus.

Idx.	Model	Loss	Acc. R	Acc. S	Acc. F	Acc Ponder.
1	LSTM	0.2424	96.21 %	89.13 %	76.17 %	84.12 %
2	LSTM*	0.2855	96.18 %	92.37 %	73.87 %	84.34 %
3	GRU	0.2299	96.13 %	89.60 %	79.09 %	85.65 %
4	GRU*	0.2469	96.70 %	92.17 %	75.48 %	85.08 %
5	BI-LSTM	0.3074	96.71 %	93.24 %	68.73 %	82.39 %
6	BI-LSTM*	0.2475	95.82 %	93.49 %	74.14 %	84.86 %
7	CNN	0.2971	95.61 %	88.83 %	72.07 %	82.03 %
8	CNN*	0.2916	96.44 %	87.56 %	69.17 %	80.30 %
9	CNN-LSTM	0.4186	96.00 %	84.04 %	56.29 %	72.88 %
10	CNN-LSTM*	0.4263	95.68 %	81.77 %	55.41 %	71.53 %

Tabela 6.4: Rezultate obținute pentru modelele antrenate doar pe baza prezentării.

Loc	Model	Țimp	Loss 128	Acc. 128	Total Puncte
1	GRU	7	10	10	27
2	GRU*	7	8	9	24
3	BI-LSTM*	8	7	8	23
4	LSTM	6	9	6	21
5	LSTM*	6	6	7	19
6	CNN	10	4	4	18
7	CNN*	10	5	3	18
8	BI-LSTM	8	3	5	16
9	CNN-LSTM	9	2	2	13
10	CNN-LSTM*	9	1	1	11

Tabela 6.5: Modelele sortate în ordinea performanței.

Din tabel reiese faptul că modelele care aveau performanțe excelente pe secvențe mai lungi funcționează mult mai rău pe secvențe scurte (a se vedea diferențe la CNN și CNN-LSTM). Cel mai bun model este **GRU** iar locurile doi și trei sunt ocupate de modele **GRU*** și **BI-LSTM*** antrenate pe text fără diacritice.

Un lucru impresionant este faptul că am putut produce rezultate comparabile cu cele de mai sus folosind doar prezentarea articolului, în loc să folosim tot articolul. Din câte am observat, modelele bazate pe GRU funcționează mult mai bine decât alte modele, pe secvențe scurte de text.

Capitolul 7

Concluzii și dezvoltări ulterioare

În acest ultim capitol, voi prezenta câteva păreri legate de rezultatele obținute folosind modelele propuse iar ulterior, voi discuta în mare despre ce posibilități de dezvoltare există pentru abordarea propusă.

Lucrarea de față a avut ca scop crearea unui sistem de verificare al articolelor de știri, pentru a limita efectele negative produse de distribuirea de știri false în mediul online. Am propus cinci modele care au fost antrenate pe un set de date colectat de mine din diferite surse de știri. După antrenare, modelele au fost evaluate și cel mai bun dintre ele a fost incorporat într-un API pe care l-am folosit într-o aplicație. Aplicația propusă este una destul de simplă și are ca scop adunarea unor volume mai mari de date pentru ca, în viitor, să putem produce modele cu rezultate și mai bune.

Personal, am considerat că rezultatele obținute au fost peste așteptări. Chiar dacă perioada de antrenare a modelelor a fost îndelungată și plină de dificultăți, sunt fericit pentru ce am reușit să realizez într-un timp atât de scurt. Cel mai mult mă bucur pentru rezultatul obținut cu GRU, clasificând articolele doar pe baza prezentării și obținând performanțe aproape la fel de bune ca în cazul clasificărilor pe baza întregului conținut.

De menționat este și faptul că, din păcate, nu am reușit să produc rezultate folosind modele contextuale complexe (precum ELMo sau BERT). În lucrările viitoare, sper să nu mai fie vorba de astfel de probleme. Chiar dacă nu am reușit să folosesc acele modele, am venit cu o propunere de model (CNN-LSTM) care a obținut rezultate promițătoare.

Totodată, consider că aș fi putut obține rezultate mult mai bune dacă aș fi avut resursele necesare pentru a derula *grid-search* spre a găsi hiperparametrii optimi pentru fiecare model. Pe lângă asta, dacă aș fi verificat individual fiecare articol din setul de date, as fi redus numărul de probleme apărute din cauza etichetărilor greșite.

În concluzie, chiar dacă mai era loc de îmbunătățiri, consider că am făcut o treabă destul de bună în dezvoltarea acestei lucrări.

Dezvoltări ulterioare

Ideile de dezvoltare s-ar împărți în două categorii, fie idei care duc la îmbunătățirea procesului de clasificare, fie idei pentru îmbunătățirea aplicației.

Pentru a îmbunătăți procesul de clasificare, s-ar putea încerca colectarea mai multor date folosind aplicația propusă de noi. Având mai multe date, modelele ar putea generaliza mai bine pe datele din viața reală.

Pe lângă asta, alte metode de preprocesare ar putea fi utilizate pentru a observa efectul lor asupra rezultatelor. Metode precum *stemming* sau *lemmatizare* ar putea duce la rezultate mult mai bune.

De asemenea, s-ar putea antrena modele din familia *BERT* sau *ELMo*, în momentul în care s-ar obține un set de date destul de cuprinzător. Conform literaturii de specialitate, aceste modele au performanțe chiar mai bune decât modelele secvențiale sau convolutive. O altă arhitectură care ar merita încercată este cea de rețea neuronală ierarhică (HAN).

Pentru a îmbunătăți aplicația, s-ar putea introduce un sistem pentru validarea automată a etichetelor pe baza sursei de proveniență. Un astfel de sistem s-ar folosi de o bază de date cu surse credibile și ar automatiza o parte din munca depusă de moderatorii.

O alternativă ar fi să se dezvolte o versiune mai *light* a acestei aplicații sub formă de extensie de browser. Această extensie ar putea folosi endpointul de clasificare pe baza prezentării, pentru a clasifica postările din newsfeedul unei rețele de socializare. În urma clasificării, postările cu conținut negativ ar putea fi ascunse pentru ca utilizatorul să nu interacționeze cu ele.

Contribuții personale

- Am introdus o clasă nouă pentru articolele de satiră.
- Am colectat peste 50.000 articole de știri pe care le-am curățat, etichetat și folosit pentru antrenarea a cinci modele de clasificare.
- Am încercat diferite arhitecturi pentru modelele propuse, le-am evaluat performanța și am discutat rezultatele produse de fiecare arhitectură în parte.
- Am proiectat și implementat o aplicație ce ușurează procesul de colectare și etichetare a articolelor de știri pentru ca pe viitor să putem construi seturi de date mai mari și de o mai bună calitate.
- Am introdus modelul CNN-LSTM. Acesta obține rezultate mult mai bune decât modelele bazate pe LSTM sau GRU și se și antrenează mai rapid.

Bibliografie

- [1] Prashansa Agrawal, Parwat Singh Anjana, and Sathya Peri. Dehide: Deep learning-based hybrid model to detect fake news using blockchain. *International Conference on Distributed Computing and Networking 2021*, Dec 2020.
- [2] Bjarte Botnevik, Eirik Sakariassen, and Vinay Setty. Brenda: Browser extension for fake news detection. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul 2020.
- [3] Eligijus Bujokas. Creating Word Embeddings: Coding the Word2Vec Algorithm in Python using Deep Learning. <https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8>. Online; accessed 15 May 2021.
- [4] Businessmagazin.ro. Cine este miliardarul european care termină dominația americană devenind cel mai bogat om din lume, după ce l-a depășit pe jeff bezos. <https://www.businessmagazin.ro/actualitate/cine-este-miliardarul-european-care-termina-dominatia-americana-20107533>. Online; accessed 1 June 2021.
- [5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.
- [6] Alistair Coleman. 'hundreds dead' because of covid-19 misinformation. <https://www.bbc.com/news/world-53755067>. Online; accessed 3 March 2021.
- [7] Clackamas College. What's wrong with fake news? <https://libguides.clackamas.edu/c.php?g=652128&p=4574289>. Online; accessed 20 February 2021.
- [8] Victor Cozmei. Sondaj european despre fake news. <https://economie.hotnews.ro/stiri-media-publicitate-22360411->

eurobarometru-sondaj-fake-news-stiri-false-romania-retele-sociale.htm. Online; accessed 11 April 2021.

- [9] Zhiyong Cui, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values. *Transportation Research Part C: Emerging Technologies*, 118:102674, sep 2020.
- [10] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Hubbard, L. D. Jackel, and D. Henderson. *Handwritten Digit Recognition with a Back-Propagation Network*, page 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.
- [11] Edupedu. Studiu: Nouă din zece liceeni cred că fenomenul fake news este o problemă în românia. <https://www.edupedu.ro/studiu-noua-din-zece-liceeni-cred-ca-fenomenul-fake-news-este-o-problema-in-romania/>. Online; accessed 11 April 2021.
- [12] Akansha Gautam and Koteswar Rao Jerripothula. Sgg: Spinbot, grammarly and glove based fake news detection, 2020.
- [13] Masato Hagiwara. Improving a sentiment analyzer using elmo — word embeddings on steroids. <http://www.realworldnlpbook.com/blog/improving-sentiment-analyzer-using-elmo.html>. Online; accessed 11 May 2021.
- [14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [15] Heejung Jwa, Dongsuk Oh, Kinam Park, Jang Kang, and Heuiseok Lim. ex-bake: Automatic fake news detection model based on bidirectional encoder representations from transformers (bert). *Applied Sciences*, 9:4062, 09 2019.
- [16] Hamid Karimi, Proteek Roy, Sari Saba-Sadiya, and Jiliang Tang. Multi-source multi-class fake news detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1546–1557, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.
- [17] Simon Kemp. Digital 2021: Global overview report. <https://datareportal.com/reports/digital-2021-global-overview-report>. Online; accessed 8 March 2021.
- [18] Sebastian Kula, Michał Choraś, and Rafał Kozik. *Application of the BERT-Based Architecture in Fake News Detection*, pages 239–249. 01 2021.

- [19] Yang Liu and Yi-Fang Wu. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. 12 2018.
- [20] Bernard Marr. How much data do we create every day? the mind-blowing stats everyone should read.
- [21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.
- [22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.
- [23] Cristopher Olah. Understanding lstm networks. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Online; accessed 10 May 2021.
- [24] Demetris Paschalides, Alexandros Kornilakis, Chrysovalantis Christodoulou, Rafael Andreou, George Pallis, Marios D. Dikaiakos, and Evangelos Markatos. Check-it: A plugin for detecting and reducing the spread of fake news and misinformation on the web, 2019.
- [25] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.
- [26] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.
- [27] Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.
- [28] Natali Ruchansky, Sungyong Seo, and Yan Liu. Csi. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, Nov 2017.
- [29] Tanik Saikh, Arkadipta De, Asif Ekbal, and Pushpak Bhattacharyya. A deep learning approach for automatic detection of fake news, 2020.

- [30] Linley Sanders. The difference between what republicans and democrats believe to be true about covid-19. <https://today.yougov.com/topics/politics/articles-reports/2020/05/26/republicans-democrats-misinformation>. Online; accessed 3 March 2021.
- [31] Justina Alexandra Sava. Most trusted news brands in romania in 2020, by trust score. <https://www.statista.com/statistics/1104247/most-trusted-news-brands-romania/>. Online; accessed 18 May 2021.
- [32] Kai Shu, Suhan Wang, and Huan Liu. Beyond news contents: The role of social context for fake news detection, 2018.
- [33] Michael Siering, Jascha-Alexander Koch, and Amit Deokar. Detecting fraudulent behavior on crowdfunding platforms: The role of linguistic and content-based cues in static and dynamic contexts. *Journal of Management Information Systems*, 33:421–455, 04 2016.
- [34] Sneha Singhania, Nigel Fernandez, and Shrisha Rao. 3han: A deep neural network for fake news detection. 11 2017.
- [35] Vaibhav Vaibhav, Raghuram Mandyam Annasamy, and Eduard Hovy. Do sentence interactions matter? leveraging sentence level representations for fake news classification, 2019.
- [36] Svitlana Volkova, Kyle Shaffer, Jin Jang, and Nathan Hodas. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. pages 647–653, 01 2017.
- [37] William Yang Wang. “liar, liar pants on fire”: A new benchmark dataset for fake news detection, 2017.
- [38] Jiawei Xu, Vladimir Zadorozhny, Danchen Zhang, and John Grant. Fands: Fake news detection system using energy flow, 2020.
- [39] Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. pages 98–101, 06 2020.
- [40] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. 02 2017.
- [41] Jiawei Zhang, Bowen Dong, and Philip S. Yu. Fakedetector: Effective fake news detection with deep diffusive neural network, 2019.