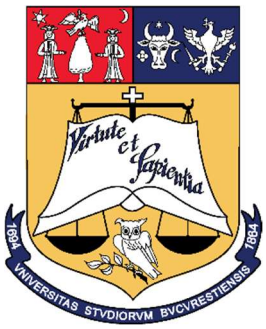

UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

FAKE-NEWS DETECTION

Absolvent
Gherghel-Buțan Mihnea

Coordonator științific
Claudia Muresan

București, iulie 2024

I Rezumat

Accesul la informații adevărate și reale reprezintă un aspect important în viața fiecărei persoane, influențând în mod decisiv opiniile și impresiile fiecărui individ în aspecte relevante precum: politică, sport, mediu înconjurător, război, sistem medical sau sistem educațional.

Având în vedere aspectele prezentate mai sus, am considerat oportună crearea unei aplicații web care să detecteze știrile false, folosind tehnici specifice Învătării automate[1] (Machine Learning), precum Random Forest[2], Support Vectore Machine[3] sau Rețele Neurale[4] și Prelucrării Limbajului Natural precum Word2vec[5], Tf-Idf[6] sau Lematizare[7].

În vederea realizării aplicatiei web au fost utilizate tehnologii actuale precum React[8] pentru construirea interfetei cu utilizatorul, NodeJs[9] pentru realizarea serverului ce se ocupă de prelucrarea cererilor transmise de interfață și interogarea bazei de date, MySQL[10] pentru baza de date ce are funcția de a reține datele necesare funcționării aplicației și Python[11] pentru realizarea serverului ce gestionează modelul care prezice dacă o știre este adevărată sau nu.

II Cuprins

III Introducere

Tipul lucrării și subdomeniul în care se încadrează tema

Prezentare generală a temei și motivația alegerii

Contribuția în realizarea lucrării și structura ei

Repere istorice și rezultate cunoscute

1. Tipul lucrării și subdomeniul în care se încadrează tema

Această lucrare de licență se axează atât pe dezvoltarea unei aplicații web utilizând tehnologii modern, cât și pe antrenarea unor modele și utilizarea unor algoritmi ce să poată prezice pe baza titlului și textului unei știri dacă aceasta este falsă sau nu. Principalele domenii explorate în această lucrare sunt:

1. **Prelucrarea limbajului natural**: utilizată pentru extragerea caracteristicilor și preprocesarea știrilor.
2. **Învățare automată**: utilă în conceperea modelelor și algoritmilor folosiți în detectarea știrilor false.
3. **Aplicație web**: realizarea unei aplicații folosind React[5], Node.Js [4] și MySQL[3] pentru a facilita accesul clienților la model.

2. Prezentarea generală a temei și motivația alegerii

Tot mai multe persoane se informează prin intermediul internetului. Un sondaj realizat de Avangarde arată că 15% dintre români se informează de pe internet, iar procentul se dublează în cazul tinerilor cu vârsta cuprinsă între 18 și 31 de ani[12]. Având în vedere acest aspect, sursele de informații online încep să aibă o importanță crucială în societatea contemporană, care se confruntă cu un numar tot mai mare de știri false, fapt ce conduce la dezinformarea maselor. Impactul dezinformării poate fii devastator având un impact devastator asupra unor domenii cheie. Astfel, consider că detectarea și distingerea știrilor reale de cele false este esențială.

Luând în considerare ideile prezentate mai sus, am considerat că este de bun augur să fac o aplicație care să detecteze știrile false utilizând tehnici specifice învățării automate și prelucrarea limbajului natural.

3. Contribuția în realizarea lucrării și structura ei

Lucrarea cuprinde două părți esențiale:

1. **Predictia știrilor**: include preluarea, analizarea, preprocesarea și împărțirea datelor, dar și antrenarea și testarea diferitelor tehnici și modele specifice învățării automate și prelucrării limbajului natural.
2. **Aplicația Web**: este împărțită în trei părți importante, fiecare realizată într-o tehnologie specifică:
    a. **Partea de frontend**: Aceasta va fi compusă din 5 pagini web:
        I. **Pagina Home**: Include o prezentare generală despre aplicație și utilitatea ei.
        II. **Pagina de Login**: Facilitează logarea utilizatorului.
        III. **Pagina de Sign up**: Facilitează înregistrarea utilizatorului.
        IV. **Pagina de introducere a titlului și textului**: Utilizatorul introduce titlul și textul unei știri și va primi ca răspuns procentul în care modelul consideră că o știre este falsă.
        V. **Pagina de introducere a URL-ului**: Utilizatorul introduce URL-ul unei știri, iar modelul va analiza dacă știrea de pe acel URL este falsă sau nu.
    b. **Partea de backend**: Aceasta este compusă la rândul ei din alte două părți:
        I. **Serverul de Node.Js** [4] : Se ocupă de cererile venite de la frontend, interogările asupra bazei de date și trimite cereri către serverul de Python[2].
        II. **Serverul Python** [2]: Are ca atribuții răspunderea la cererile venite de pe serverul de Node.js și predicția știrilor pe baza modelului.
3. **Baza de date**: Conține informații relevante despre utilizatori, tipul contului pe care îl au și numărul de interogări pe care le-au făcut.

4. Repere istorice și rezultate cunoscute

Utilizarea inteligentei artificiale în vederea detectării știrilor false se lovește și în prezent de două mari probleme: lipsa seturilor de date care ar trebui să surprindă specificul local al fiecărei regiuni sau țări și abilitatea creatorilor de știri false de a-și adapta stilul.

Cu toate acestea, de-a lungul timpului, oamenii au încercat să depășească aceste probleme și să construiască modele cât mai performante:

1. Veracity(2007) : a fost un program aparut în 2007 ce detecta știrile false pe baza unor reguli precum apariția anumitor cuvinte.
2. Fake-News Challenge[13]: în anul 2017 a avut loc o competiție bazată pe constrcția de clasificatoare pentru detectarea știrilor false utilizând modele specifice de învățare automată. Echipa câștigătoare a obtinut o acuratețe de 92%.

IV Preliminarii

Noțiuni științifice și tehnologice ce stau la baza temei

Știrile false și stirile adevarate

Problema propusă spre rezolvare și obiectivele proiectului

1. Noțiuni științifice și tehnologice ce stau la baza temei

Domeniul inteligenței artificiale a cunoscut o dezvoltare exponentială în ultimul deciniu datorită investițiilor majore venite din partea mediului academic și al marilor corporații din domeniul tehnologiei precum Google, Facebook sau Siemens. Printre principalele beneficii pe care le aduce acest domeniu se numară automatizarea proceselor unde este nevoie de intervenție umană, îmbunătățirea performației sistemelor, soluționarea problemelor complexe și simplificarea vieții cotidiene. Industriile principale ce utilizează inteligența artificială sunt: medicina, automatizare industrială, finante, transport și asistență personală. AI este un domeniu general compus din mai multe subdomenii printre care se regasesc învățare automată(machine learning) sau procesarea limbajului natural(natural language processing).

Învătarea automată reprezintă un subdomeniu al inteligenței artificiale care se concentrează pe dezvoltarea unor algoritmi și modele care permit unității de calcul să învețe pe baza unor date. Spre deosebire de programarea clasică, modelele și algoritmii de învățare automată nu au decizile programate explicit ci învață să ia decizii pe baza datelor de antrenare, dobândind capacitatea de a face predicții pe date noi. Principalele avantaje pe care le are acest subdomeniu sunt: capacitatea de a procesa un volum mare de date, de a descoperi tipare complexe în cadrul seturilor de date și de a face predicții mai precise decât omul în anumite cazuri.

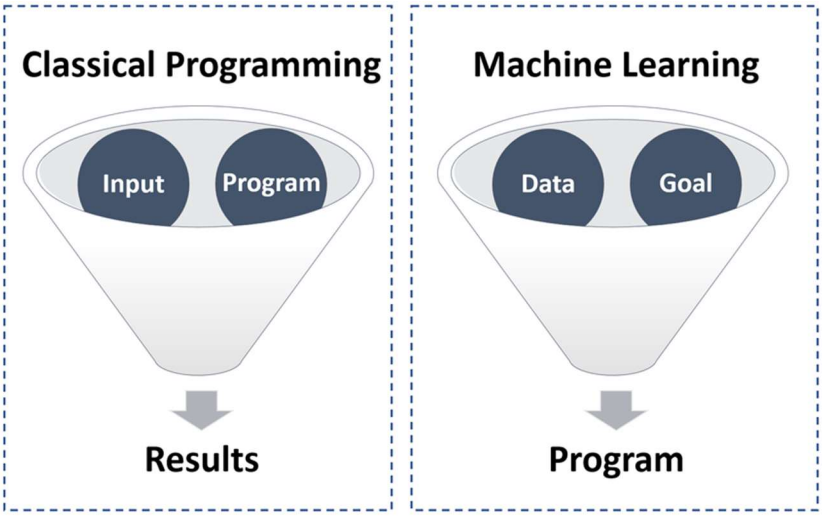

Figura IV.1 Programarea clasică vs Machine Learning

Procesarea limbajului natural este un subdomeniu al inteligenței artificiale care se concentrează pe interacțiunile dintre unitățile de calcul și limbajul uman. Scopul acestui domeniu este de a realiza modele cât mai bune ce să poată înțelege, interpeta și genera limbaj uman. Principalele domenii în care se utilizează aceste tehnologii sunt: traducerea automată, chatboti, analiza sentimentelor, motoare de căutare, medicină sau social media. Principalele etape în dezvoltarea unui proiect bazat pe procesarea limbajului natural sunt: preprocesarea textului(elimnarea semnelor de punctuație, eliminarea cuvintelor nesimnificative, reducerea cuvintelor la forma lor de bază, reprezentarea cuvintelor folosind numere), selectarea și construirea modelelor, antrenarea modelelor, evaluarea modelelor, alegerea celui mai bun model și optimizarea lui.

2. Știrile false și știrile adevarate

Știrile reprezintă principala modalitate de informare a oamenilor în domenii precum: politică, societate, tehnologie, sport, economie, societate, mediu, cultură, conținând informații actuale ce afectează viața cotidiană a unui persoane.

O știre bună și reală respectă următoarele aspecte:

- **Neutralitate și obiectivitate**: O știre reală ar trebui să fie obiectivă și să redea toate perspectivele relevante asupra subiectului. Jurnalisti ar trebui să redea o acoperire echilibrată asupra subiectului
- **Acuratețe și Verificabilitate**: O știre reală trebuie să prezinte un subiect fără să adauge informații suplimentare, cu informații precise susținute de dovezi solide. De asemenea, informațiile trebuiesc verificate în toate sursele implicate în acel subiect.
- **Context**: O știre bună ar trebui să ofere toate infomațiile și tot contextul necesar înțelegerii corecte a subiectului
- **Independență editorială și transparență**: O știre bună ar trebui sa fie lipsită de influențe externe și de presiunile persoanelor sau instituților implicate. De asemenea, este important ca sursele de informare în legătură cu respectiva știre să fie publice.
- **Corectitudinea limbajului**: O știre reale ar trebui să fie corectă din punct de vedere grmatical și să evite utilizarea unui limbaj agresiv sau discriminatoriu

O știre falsă reprezintă o înlănțuire de informații false, fabricate, distorsionate sau inexacte cu scopul de a manipula opinia publică, de a crea confuzie și conflicte în societate, de a promova o anumită agendă politică sau de a manipula pieţele financiare. Știrile false pot fi de mai multe tipuri în funcție de informațiile pe care le redau:

- **Știri fabricate complet**: Acest tip de știri relatează evenimente care nu s-au întâmplat niciodată, fiind adesea senzaționaliste
- **Știri cu informații înselătoare și distorsionate**: Acest tip de știri pornesc de la un eveniment real, dar anumite fapte sau idei sunt distorsionate, omise sau exagerate pentru a susține o anumită perspectivă
- **Știri cu contextul omis**: În acest tip de știri false, sunt omise anumite informații relevante, care pot schimba persepectiva cititorului asupra unei situații

3. Problema propusă spre rezolvare și scopul proiectului

Lucrarea de față își propune să automatizeze procesul de detectare a știrilor false prin utilizarea unor algoritmi și modele specifice învățării automate și procesării limbajului natural, cu scopul de a reduce dezinformarea. De asemenea, pentru a putea fi utilizată cu ușurință lucrarea își propune dezvoltarea unei aplicații web.

În scopul realizării scopul descris mai sus s-au realizat următorele etape:

1. Colectarea și preprocesarea datelor
2. Construirea, antrenarea și testarea unor algoritmi specifici învățării automate
3. Realizarea aplicației web:
    - Interfața cu utilizatorul
    - Serverului central care face legătura cu restul componenetelor
    - Baza de date
    - Serverul de python care conține modelul de învățare automată

IV Preluare și preoprocesarea datelor

1. Preluarea datelor

În cadrul proiectului s-au utilizat două seturi de date: unul dintre ele a fost preluat de pe platforma Kaggle[14], în timp ce, al doilea set de date a fost compus prin extragerea știrilor de pe site-ul Veridica, versiunea în engleză[15].

1.1 Preluarea datelor de pe Kaggle

Setul de date de pe platforma Kaggle conținea cinci coloane pentru fiecare exemplu:

1. id – reprezintă identificatorul unic al fiecărei știri
2. title – reprezintă titlul unei știri
3. author – reprezintă autorul respectivului articol
4. text – reprezintă textul respectivei știri
5. label – indică 1 dacă știrea este falsă și 0 altfel

Dintre cele 5 coloane ale setului de date de pe Kaggle au fost extrase id, title, text și label și redenumite: Index, Title, Text, Label, urmând să fie utilizate pentru antrenarea și optimizarea modelelor de învățare automată.

```
Codeium: Refactor | Explain | Generate Docstring | ×
def kaggleFakeNews():
    file_path = 'C:\\Users\\40741\\Documents\\LucrareLicenta\\Data\\ExcelData\\kagg
    excel_kaggle = pd.read_csv(file_path)
    columnDelete = ["id", "author"]
    excel_kaggle = excel_kaggle.drop(columns=columnDelete)
    change_name = {'title': 'Title', 'text': 'Text', 'label': 'Label'}
    excel_kaggle = excel_kaggle.rename(columns=change_name)
    excel_kaggle = excel_kaggle.loc[(excel_kaggle['Title'].apply(type) == str) &
                                    (excel_kaggle['Text'].apply(type) == str)]
    excel_kaggle.insert(0, 'Index', excel_kaggle.reset_index().index)
    excel_kaggle.to_excel('kaggleFakeNewsFinal.xlsx', index=False)


kaggleFakeNews()
```

Figura IV.1 – selectarea coloanelor necesare

1.2 Preluarea datelor de pe Veridica

Preluarea datelor a fost realizată utilizând biblioteca BeautifulSoup din Python și a inclus trei etape:

1. **Extragerea URL-urilor aferente știrilor** a fost realizată prin preluarea conținutului HTML al fiecărei pagini principale a site-ului. Pagina principală a unui site de știri conține titlurile, niște scurte rezumate și link-uri către întregile articole ale mai multor știri. Dupa preluarea conținutului HTML a paginilor au fost extrase toate hyperlink-urile care aveau atribuită clasa “stretched-link” în cazul știrilor reale, respectiv toate hyperlink-urile care au un anumit sufix în cazul știrilor false. Astfel, au fost obținute URL-urile aferente știrilor de pe Veridica

```python
def returnURLToPages(url):
    # request catre url paginii de pe care se doreste extragerea informatiei
    response = requests.get(url)
    # extragerea url-urilor din pagina
    article_links = re.findall(
        r"<a href=\"(.*?)\">(.*?)</a>", response.content.decode('utf-8')
    )
    # filtrarea url-urilor
    target_formats = ['https://www.veridica.ro/en/fake-news/',
                      'https://www.veridica.ro/en/disinformation/',
                      'https://www.veridica.ro/en/propaganda/']
    filtered_urls = []
    for article_link in article_links:
        for target_format in target_formats:
            if article_link[0].startswith(target_format):
                filtered_urls.append(article_link[0])
    return filtered_urls
```

Figura IV.2 – extragerea URL-urillor aferente știrilor false

2. **Extragerea știrilor de pe site-ul Veridica** a fost realizată utilizând URL-urile obținute la pasul anterior. A fost extras întreg conținutul paginilor aferente fiecărui URL. Utilizând metode speciale din biblioteca BeautifulSoup au fost extrase titlurile textelor, în timp ce pentru extragerea textelor au fost concatenate toate textele aferente elementelor “div” din HTML care au clasa article-content. Pasul următor a fost reprezentat de salvarea titlurilor și textelor în fișiere de tip txt

```python
def extractDataVeridica(url):
    # se trimite un request catre url
    response = requests.get(url)
    # se verifica daca requestul a fost acceptat
    if response.status_code == 200:
        # se parseaza continutul paginii de tip html
        soup = BeautifulSoup(response.text, 'html5lib')
        # se extrage titlul articolului
        title = soup.title.text.strip()
        # se extrage continutul articolului
        text_content_div = soup.find('div', class_='article-content')
        if text_content_div == None:
            return
        # se extrage textul articolului
        paragraphs = text_content_div.find_all('p')
        text_content = "\n".join([paragraph.text.strip()
                                  for paragraph in paragraphs])
        # obtinere nume fisier
        match = re.search(r"/([^/]+)$", url)
        if match:
            extracted_url = match.group(1)
            # dimensiunea fisierului e redusa la maxim 45 de caractere
            if len(extracted_url) > 45:
                extracted_url = extracted_url[:45]
            # se adauga in fisierul cu nume corespunzator titlul si continutul stirii false
            with open(f"data\\{extracted_url}.txt", 'w', encoding='utf-8') as file:
                file.write(
                    f"Title: {title}\n\nText Content:\n{text_content}")
```

Figura IV.3 – extragerea textului știrilor de pe site-ul Veridica

3. **Convertirea datelor în format excel** a fost realizată prin preluarea și citirea datelor din folderele ce conțin știrile reale și știrile false. Ulterior a fost realizat un document excel care conținea coloana Index(id unuic pentru fiecare știri prealuată de pe site-ul veridica), Title(titlul știrii). Text(textul știrii) și Label(coloană ce indică dacă știrea este reală sau nu).

```
Codeium: Refactor | Explain | Generate Docstring | ×
def createInformation():
    folder_path_fake = '..\\ExtragereDataVeridicaFakeNews\\data'
    folder_path_real = '..\\ExtragereDateVeridicaRealNews\\data'
    files_name_fake = allFileNameInFolder(folder_path_fake)
    files_name_real = allFileNameInFolder(folder_path_real)
    informationFake = constructInformation(files_name_fake, "fake", 0)
    counter = len(informationFake)
    informationReal = constructInformation(files_name_real, "real", counter)
    news = []
    for info in informationFake:
        news.append(info)
        counter += 1
    for info in informationReal:
        news.append(info)
        counter += 1
    return news


Codeium: Refactor | Explain | Generate Docstring | ×
def veridicaExcel():
    news = createInformation()
    df = pd.DataFrame(news, columns=["Index", "Title", "Text", "Label"])
    df.to_excel("veridicaData.xlsx", index=False)
```

Figura IV.4 – covertirea știrilor provenite de pe Veridica din format text în format excel

3. Informații despre setul de date

3.1. Informații despre setul de date preluat de pe Kaggle

Setul conține în total un numar de **20203** de știri dintre care **10387** sunt știri reale, în timp ce **9816** sunt false.

Setul de date de pe platforma Kaggle se axează pe știri ce prezintă evenimente care au avut loc în Statele Unite ale Americii, impactează în vreun mod acest stat(acțiuni teroriste) sau care surprind o problemă globală(Brexit) din perioada 2013-2017. Politica este principalul subiect abordat în cadrul acestui set de date, accentul fiind pus pe lupta dintre democrați și republicani, dintre Hillary Clinton și Donald Trump pentru funcția de președinte al Statelor Unite ale Amercii. Acest lucru nu este surpinzător întrucât conform unui studiu realizat de NewsGuard, în Statele Unite ale Americii au fost identificate în jur de 150 de site-uri care se ocupau cu distribuirea de informații false dintre care 60% aveau ca subiect principal politica[17]. Alte subiecte importante sunt războiul(cursa înarmărilor, anexarea Crimeei) și economia(încercarea de manipulare a piețelor financiare).

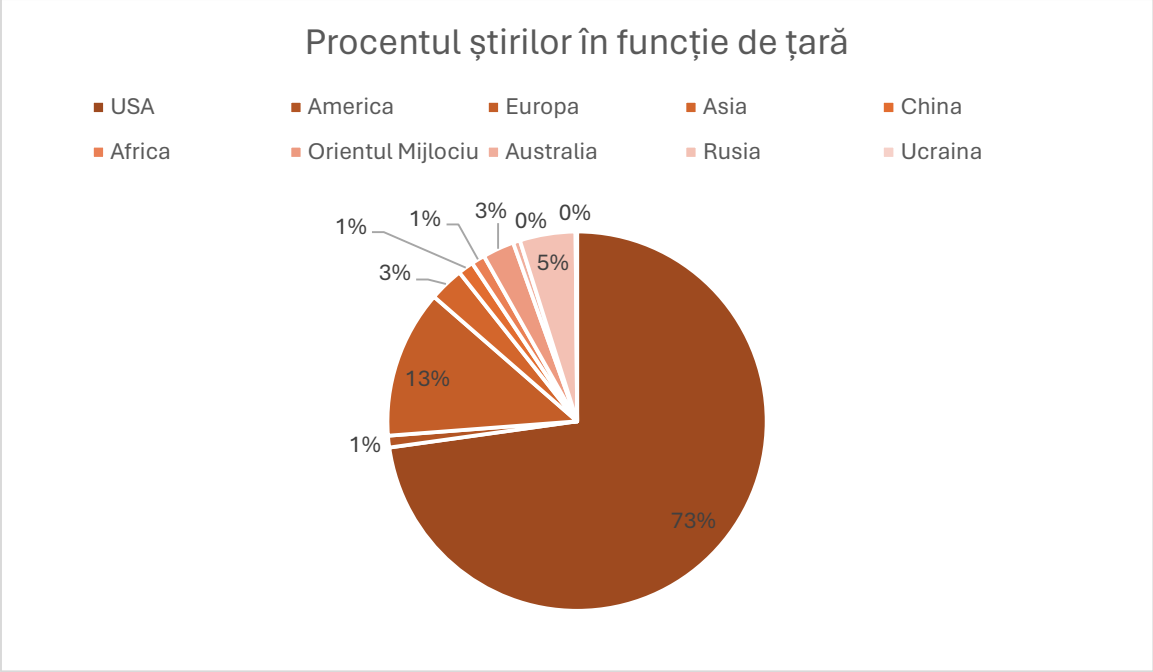

Diagrama IV.1 – Procentul știrilor din setul de date de pe Kaggle în funcție de țară

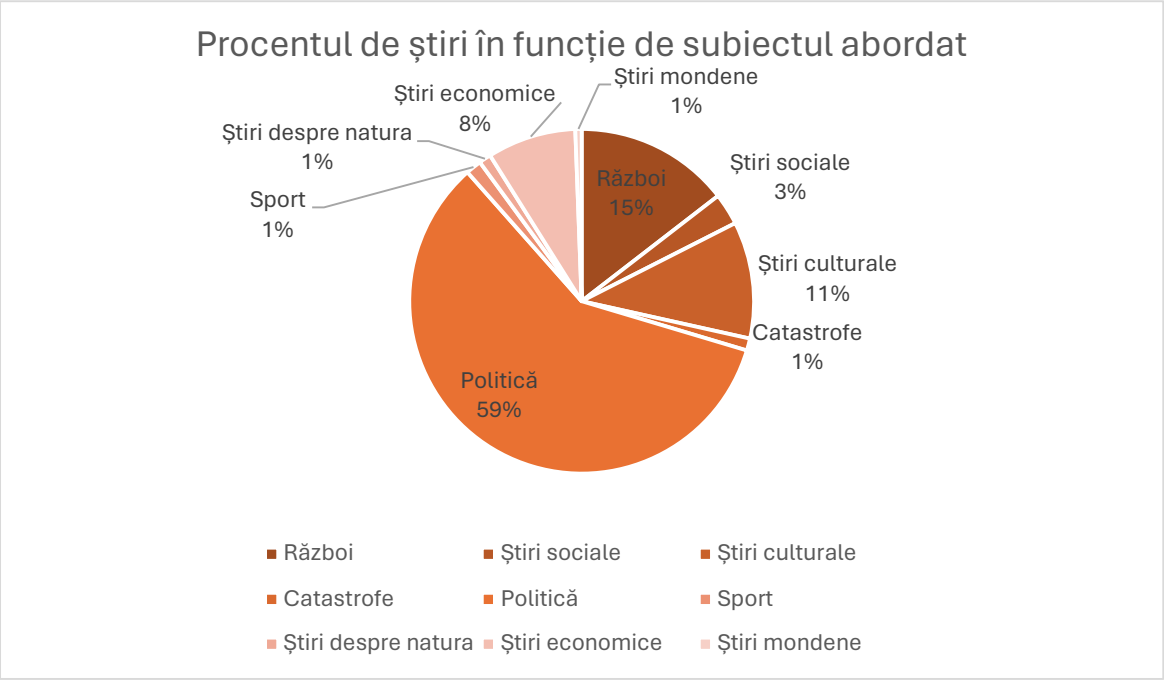

Diagrama IV.2 – Procentul de știri din setul de date de pe Kaggle în funcție de subiectul abordat

Aceste statistici au fost realizate prin determinarea unor cuvinte cheie care apar în general în știrile ce aparțin unui anumit domeniu, respectiv țari. De exemplu cuvinte cheie pentru știrile ce surpind evenimente din Statele Unite ale Americii sunt: “America” ,”america” ,”usa” ,”USA”, ”Trump” sau ”Clinton”. Au fost numarate cuvintele cheie care indică spre un anumit subiect respectiv țară. Știrea a fost asociată domeniului și țării pentru care apar

un numar maxim de cuvinte cheie în textul știrii. Pe baza acestor statistici s-a făcut ulterior împărțirea datelor în date de antrenare, testare și validare.

```
Codeium: Refactor | Explain | Generate Docstring | ×
def numberOfWords(element, pPattern):
    pattern = {}
    for key, value in pPattern.items():
        for word in value:
            if isinstance(element, str) == True:
                value = pattern.get(key, 0)+element.count(word)
                pattern[key] = value
    return pattern
Codeium: Refactor | Explain | Generate Docstring | ×
def maximValue(dictionar, default):
    maxim = 0
    result = default
    for key, value in dictionar.items():
        if value > maxim:
            maxim = value
            result = key
    return result
Codeium: Refactor | Explain | Generate Docstring | ×
def splitDataBySubjectRegion(data, indexes):
    result = createSplitDataType()
    for index, element in enumerate(data):
        countries = numberOfWords(element, countryPattern)
        country = maximValue(countries, "USA")
        subjects = numberOfWords(element, subjectPattern)
        subject = maximValue(subjects, "politics")
        result[country][subject].append(indexes[index])
    return result
```

Figura IV.5 – Metode realizate pentru a împărții datele în funcție de domeniu și țară

3.1. Informații despre setul de date preluat de pe Kaggle

Setul de date conține un total de 892 de știri dintre care 443 sunt știri false, iar 449 sunt știri reale.

Setul de date de pe site-ul Veridica, secțiunea în Engleză se axează pe știri ce prezintă evenimente care au avut loc în România, impactează în vreun mod acest stat(acțiuni militare la granița noastră) sau care surprind o problemă globală(războiul din Ucraina) din perioada 2021-2023. Războiul este principalul subiect abordat, invadarea Ucrainei de către Rusia fiind cel mai dezbătut aspect în știrile preluate. Acest lucru nu este surprinzător din doua motive. Primul motiv este reprezentat de interesul și teama românilor față de prezența războiului la granițele țării, iar cel de al doilea este reprezentat de dorința Rusiei ca prin știrile false să creeze spaimă în rândul statelor ce susțin Ucraina în cadrul conflictului. Alt subiect important este politica.

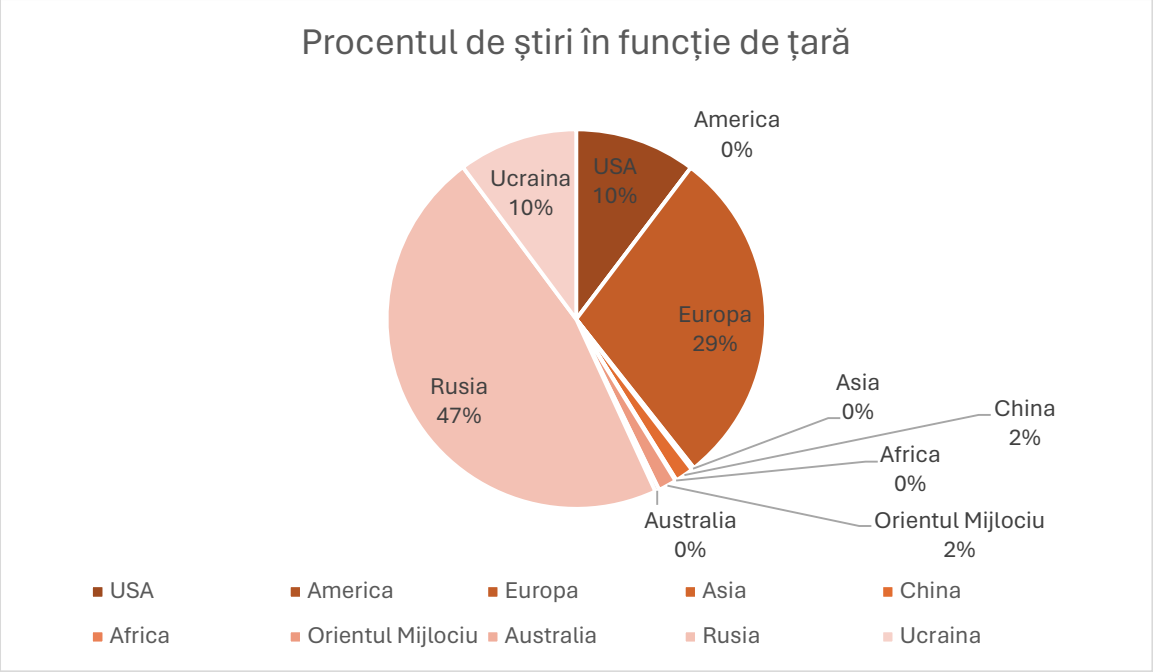

Diagrama IV.2 – Procentul de știri din setul de date de pe Kaggle în funcție de subiectul abordat

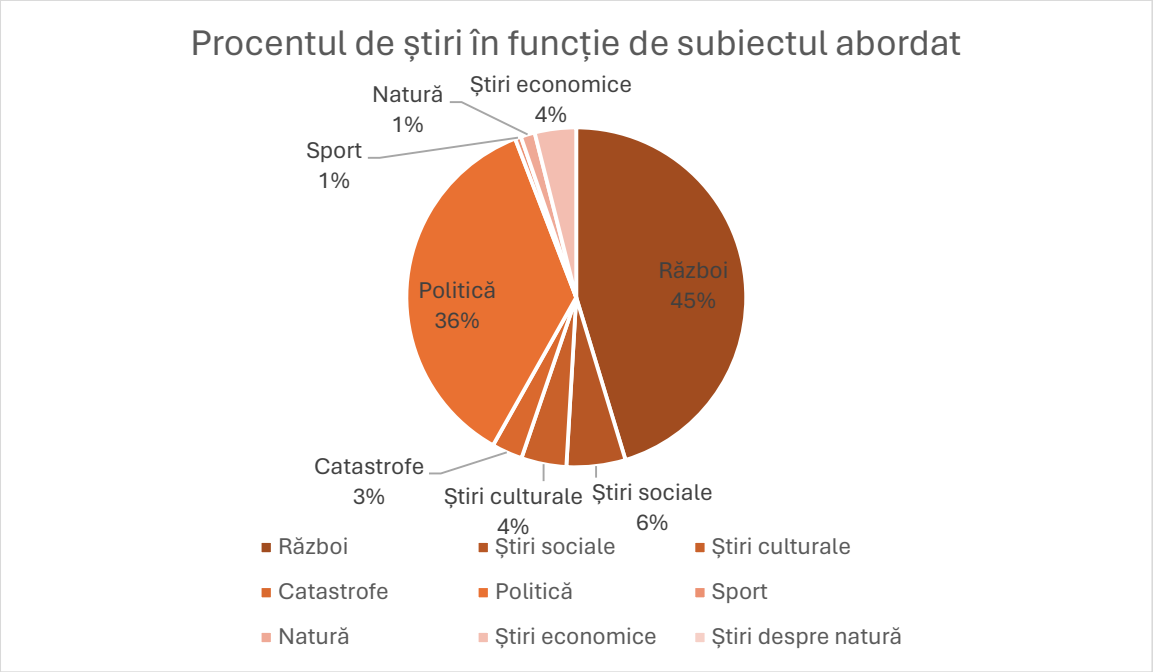

Diagrama IV.3 – Procentul știrilor din setul de date de pe Veridica în funcție de țară

Metoda de realizare a statisticilor la datele preluate de pe Veridica este identică cu cea de rezlizare a statisticilor în cazul datelor de pe Kaggle.

2.2. Corelația între tipul știri și clasa din care fac parte

O altă etapă semnificativă a proiectului a constant în descoperirea anumitor corelaților între textul știrilor și clasa din care fac parte(știri reale sau false). O primă ideea a fost aceea de a corela procentul în care apar anumite părți de vorbire cu o anumită clasă. Astfel, am plecat de la ipoteza că știrile false conțin un procent semnificativ mai mare de adverbe și adjective deoarece doresc să surpindă fapte exagerate și că sunt prezente mai multe pronume care înlocuiesc substantive deoarece informațiile prezentate nu sunt clare. Pentru a identifica procentul din fiecare parte de vorbi s-a utilizat biblioteca NTLK, bibliotecă specifică domeniului de procesare a limbajului natural.

```
def percentagesOfPartOfSpeach(text):
  parts_of_speach = {
    'NN': 0,'NNS': 0,'NNP': 0,'NNPS': 0,
    'VB': 0,'VBD': 0,'VBG': 0,'VBN': 0,'VBP': 0,'VBZ': 0,
    'JJ': 0,'JJR': 0,'JJS': 0,
    'RB': 0,'RBR': 0,'RBS': 0,'PRP': 0,'PRP$': 0,
    'IN': 0,'DT': 0,'CC': 0,'MD': 0,
    'WP': 0,'WP$': 0,'WRB': 0
  }
  tags=partOfSpeach(text)
  pos_counts=defaultdict(int)
  for word, tag in tags:
    pos_counts[tag] += 1
  total_words = len(tags)
  for pos, count in pos_counts.items():
    percentage = (count / total_words) * 100
    if pos in parts_of_speach:
      parts_of_speach[pos]=percentage
  return parts_of_speach
```

Figura IV.6 – Detectarea procentului de apariție a unei părți de vorbire

În urma analizării datelor și rezultatelor s-a ajuns la concluzia că ipoteza de plecare este falsă și că nu există corelații între frecvența părților de vorbire și clasa știrii

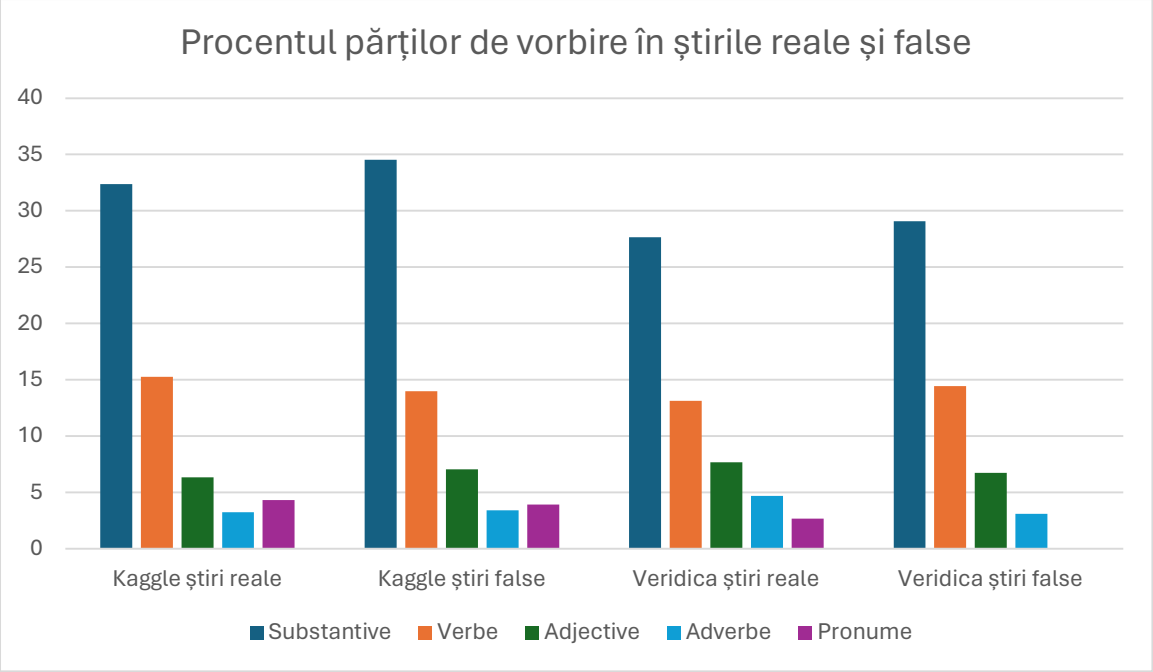

Diagrama IV.3 – Procentul părților de vorbire în știrile reale și false

3. Împărțirea setului de date

Setul de date preluat de pe platforma Kaggle a fost împărțit în trei părți:

- Train: setul de date de antrenare conține aproximativ 70% din totalul exemplelor
- Validation: setul de date de validare conține aproximativ 20% din totalul exempleror
- Test: setul de date de testare conține aproximativ 10% din totalul exemplelor

În vederea împărțirii datelor s-a ținut cont atât de subiectul central al articolului cât și de cadrul geografic de care este legat conținutul articolului. După cum este explicat în secțiunea 3.1 a lucrării datele au fost împărțite în funcție de subiect și regiune geografică. Din punct de vedere tehnic acest lucru a fost realizat prin utilizarea unui structuri de date de tip dicționar în dictionar, primul dicționar ținând cont de regiunea geografică, iar al doilea de subiect. Ulterior, știrile au fost împărțite conform procentelor prezentate mai sus în mod aleator, cu mențiunea că știrile care aparțin subiectelor cu puține exemple(știri despre natură, catastrofe, știri mondene sau sociale) au fost adunate împreună și ulterior împărțite. Acest mod de împărțire a știrilor a avut ca efect o împărțire uniformă a datelor.

```python
def splitData(data):
    train_data = []
    validation_data = []
    test_data = []
    diffrent = []
    for keyCountry, valueCountry in data.items():
        for keySubject, valueSubject in valueCountry.items():
            if keySubject in ['tragedy', 'sport', 'nature', 'entertainment', 'social']:
                diffrent += valueSubject
            else:
                train, validation, test = splitVector(valueSubject)
                train_data += train
                validation_data += validation
                test_data += test
    train, validation, test = splitVector(diffrent)
    train_data += train
    validation_data += validation
    test_data += test
    return train_data, validation_data, test_data
```

Figura IV.6 – Modul de împărțire a datelor colectate de pe Kaggle

Datele preluate de pe site-ul Veridica vor fi utilizate ca un set de test pentru a observa capacitatea de generalizare a modelelor și algoritmilor. Seturile de date preluate de pe Kaggle și cele de pe site-ul Veridica sunt distincte din punct de vedere circumstanțial în sensul în care regiunea pe care se concentrează și perioada în care se desfășoară evenimentele prezentate sunt diferite. Astfel, dacă un model reușește să prezică cu precizie atât ăe datele preluate de pe platforma Kaggle, cât și pe datele preluate de pe platforma Veridica, sunt șanse mari ca modelul să funcționeze corect și pe alte date.

4. Preprocesarea datelor

Preprocesarea este alcătuită din mai mulți pași:

1. **Alipirea titltlului și textului**: Această etapă se desfășoară înainte împărțirii setului de date în train, validation și test și constă în alipirea textului și titlului într-un singur string fiind despărțite de un spațiu.

2. **Eliminarea anumitor caractere**: Această etapă constă în eliminarea tuturor caracterelor care nu sunt alfanumerice sau $. În principiu sunt eliminate semnele de punctuație. În plus, majusculele cuvintelor sunt transformate în litere minuscule. Cele două operații sunt realizate folosind biblioteca re și operațiile pe stringuri.

```
import re

def lower_characters_and_delete_punctutation(sentence):
    words = sentence.split()
    new_words=[]
    for word in words:
      new_word = re.sub(r'[^\w\s$]', '', word).lower()
      new_words.append(new_word)
    return " ".join(new_words)
```

Figura IV.7 – Eliminarea punctuației și transformarea majusculelor

3. Eliminarea cuvintelor de oprire (stopwords): Stopwords sunt cuvinte care apar frecvent în texte dar nu ajută la întelegerea textului și nu aduc o valoare semantică. Printre cuvintele de oprire din limba engleză se numără: in, the sau an. Eliminarea cuvintelor de oprire este utilă pentru scăderea dimensiunilor datelor, eliminarea zgomotului și îmbunătățirea interpretării textului. Pentru realizarea eliminării cuvintelor de oprire am utilizat biblioteca NTLK, care oferă o listă de cuvinte de oprire pentru limba engleză.

4. Reducerea cuvintelor la forma normalizată: Această etapă constă în reducerea fiecărui cuvânt la forma lui de bază, existând două tipuri de astfel de operații: lematizare și stematizare. Scopul reducerii cuvintelor la o formă normalizată este de a oferi textelor consistență, reducerea timpului de antrenare ale modelelor, reducerea dimensionalității. În cadrul proiectului am aplicat atât lematizarea cât și stematizare pe știri.

a. Lematizarea: reprezintă procesul prin care unui cuvânt i se asociază forma lui de bază folosind cunoștiințe lingvistice și analiză contextuală. Pentru realizarea lematizării textelor am utilizat lematizatorul din WordNet din biblioteca NTLK. WordNet este un dicționar electronic și o bază de date lexicală pentru limba engleză.

```
def stopWordsLemmatizer(train, lemmatizer):
    train_final = []
    for sample in train:
        words = []
        for x in sample.split():
            if x not in stopwords.words("english"):
                words.append(lemmatizer.lemmatize(x))
        train_final.append(" ".join(words))
    return train_final

wordnet_lemmatizer = WordNetLemmatizer()
```

Figura IV.8 – Funcția utilizată pentru lematizarea cuvintelor și eliminarea cuvintelor de oprire

b. Stematizare: : reprezintă procesul prin care unui cuvânt i se asociază forma lui de bază prin elimarea sufixelor și prefixelor, folosind reguli euristice. Pentru realizarea stematizării știrilor am utilizat stemărul din biblioteca NTLK.

```
import re

def lower_characters_and_delete_punctutation(sentence):
    words = sentence.split()
    new_words=[]
    for word in words:
      new_word = re.sub(r'[^\w\s$]', '', word).lower()
      new_words.append(new_word)
    return " ".join(new_words)
```

Figura IV.9 – Funcția utilizată pentru stematizarea cuvintelor și eliminarea cuvintelor de oprire

5. Vectorizarea textului : Această etapă constă în reprezentarea textului sub forma unui vector de numere deoarece unitațile de calcul nu înteleg cuvinte ci numere. Există mai multe de vectorizări ale textului pe care le-am abordat în cadrul lucrării:

a. Tokenizarea și vectorizarea simplă: În cazul acestei metode textul este împărțit în cuvinte, fiecărui cuvânt din setul de date train i se atribuie o valoare numerică unică(tokenizatorul este antrenat) și ulterior pe baza acestei indexeri a cuvintelor din setul de date de train sunt convertite și seturile de validare și testare. Datorită dimensiunii inegale a textelor este necesară adăufarea unei completări cu 0-uri la începutul textului.

```
def tokenize_data(train,validation,test,veridica):
  tokenizer=Tokenizer(oov_token="<OOV>")
  tokenizer.fit_on_texts(train)
  with open('/content/drive/My Drive/tokenizer.pkl', 'wb') as f:
    pickle.dump(tokenizer, f)
  text_to_numbers_train=tokenizer.texts_to_sequences(train)
  text_to_numbers_validation=tokenizer.texts_to_sequences(validation)
  text_to_numbers_test=tokenizer.texts_to_sequences(test)
  text_to_numbers_veridica=tokenizer.texts_to_sequences(veridica)
  text_to_numbers_train=pad_sequences(text_to_numbers_train,padding="pre",truncating="pre",maxlen=500)
  text_to_numbers_validation=pad_sequences(text_to_numbers_validation,padding="pre",truncating="pre",maxlen=500)
  text_to_numbers_test=pad_sequences(text_to_numbers_test,padding="pre",truncating="pre",maxlen=500)
  text_to_numbers_veridica=pad_sequences(text_to_numbers_veridica,padding="pre",truncating="pre",maxlen=500)

  return text_to_numbers_train,text_to_numbers_validation,text_to_numbers_test,text_to_numbers_veridica
```

Figura IV.10 – Funcția utilizată pentru tokenizarea și vectorizarea simplă a seturilor de date

b. TF-IDF: Această tip de vectorizarea a textului este alcătuită din doi pași. În partea de antrenare a modelului este reținut pentru fiecare cuvânt logaritmul numărului total de texte supra numărul total de texte care conțin cuvântul respectiv. Această parte poartă denumirea de IDF, iar valoarea IDF pentru un cuvânt este cu atât mai mare cu cât un cuvânt apare mai rar în texte. A doua parte se axează pe numărul de apariții ale cuvântului în cadrul textului analizat. Acestă parte poartă denumirea de TF, iar valoarea TF este cu atât mai mare cu cât un cuvânt apare mai des în propoziție. Cele două valori sunt înmulțite, iar valoarea rezultată reprezintă reprezentarea cuvântului în textul respectiv. Fiecare cuvânt care apare în textul de antrenare are o valoarea asociată în reprezentarea tuturor textelor din seturile de date de antrenare, testare și validare. Pentru a implementa această vectorizare am utilizat metoda TfidfVectorizer din biblioteca Sklearn.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()

tfidf_train_stemmer= vectorizer.fit_transform(train_stemmer_stopwords)
tfidf_validation_stemmer = vectorizer.transform(validation_stemmer_stopwords)
tfidf_test_stemmer = vectorizer.transform(test_stemmer_stopwords)
tfidf_veridica_stemmer=vectorizer.transform(veridica_stemmer_stopwords)
```

Figura IV.11 – Modul de convertire a datelor de antrenare,validare, testare și Veridica utilizând metoda TF-IDF

c. Word2Vec: Acest tip de vectorizare își propune ca prin învăre automată să capteze semnificațiile semantice. Metoda își propune să înțeleagă sensul unui cuvânt pe baza cuvintelor vecine ale cuvântului dat. Astfel, Word2Vec este antrenat să prezică vecinii unui cuvânt, iar în funcție de rezultate, reprezentațiile vectoriale ale cuvântului sunt ajustate. Pentru a implementa acest tip de vectorizare a fost antrenat propriul model folosind datele de antrenare și biblioteca Gensim. Ulterior, au fost reprezentate știrile din antrenare, validare și testare folosind modelul antrenat.

```
tokenized_train = [sentence.split() for sentence in train_stemmer_stopwords]
model = Word2Vec(sentences=tokenized_train, vector_size=1, window=5, min_count=1, workers=4)

def embedding_word2vec(data_set, max_length=1000):
    result = []
    for sentence in data_set:
        embedded_sentence = []
        words = sentence.split()
        for word in words:
            if len(embedded_sentence) < max_length:
                if word in model.wv:
                    embedded_sentence.append(model.wv[word])
        while len(embedded_sentence) < max_length:
            embedded_sentence.append(np.zeros(1))
        result.append(embedded_sentence)
    return result

word2vec_train_stemmer=embedding_word2vec(train_stemmer_stopwords)
word2vec_validation_stemmer=embedding_word2vec(validation_stemmer_stopwords)
word2vec_test_stemmer=embedding_word2vec(test_stemmer_stopwords)
word2vec_veridica_stemmer=embedding_word2vec(veridica_stemmer_stopwords)
```

Figura IV.12 – Modul de convertire a datelor de antrenare,validare, testare și Veridica utilizând metoda Word2Vec

12 https://www.digi24.ro/stiri/actualitate/social/de-unde-isi-iau-romanii-informatiile-sondaj-avangarde-tv-presa-online-si-retelele-sociale-principalele-surse-de-informare-2475537

13 https://competitions.codalab.org/competitions/16843#learn_the_details-terms_and_conditions

14 https://www.kaggle.com/c/fake-news/data

15 https://www.veridica.ro/en

16 https://pypi.org/project/beautifulsoup4/

17 https://www.forbes.com/sites/toddwasserman/2021/02/19/a-much-needed-service-newsguard-cracks-down-on-fake-news/?sh=1bd47ee247f8