

**Academia de Studii Economice din București**

**Facultatea de Cibernetică, Statistică și Informatică Economică**

## **Lucrare de Licență**

**Conducător Științific:**

Prof.univ.dr. Stelian STANCU

**Absolvent**

Robert Gherghișan

**București 2023**

**Academia de Studii Economice din București**

**Facultatea de Cibernetică, Statistică și Informatică Economică**

**Analiza de tip Text Mining (TM) efectuată în contextul  
inteligenței artificiale conversaționale.**

**Conducător Științific:**

Prof.univ.dr. Stelian STANCU

**Absolvent**

Robert Gherghișan

**București 2023**

# Cuprins

Introducere .....	6
Capitolul 1. Definiții și abordări .....	8
1.1. Descoperirea cunoștințelor în baze de date ( KDD ).....	8
1.2. Data Mining.....	9
1.3. Text Mining.....	10
Capitolul 2. Preprocesarea textului si metode .....	11
2.1. Modelului spatiului vectorial (Vector space model) .....	11
2.2. Preprocesarea textului in Natural Language Processing .....	13
2.2.1. Transformarea literelor mari in litere mici .....	13
2.2.2. Tokenizare .....	13
2.2.3. Eliminarea elementelor cu semnificatie redusa.....	14
2.2.4. Stemming si lematizare .....	15
2.3. Preprocesarea lingvistica.....	15
2.3.1. Identificarea partii de vorbire .....	16
2.3.2. Parsing .....	16
2.3.3. Chunking .....	17
2.3.4. Dezambiguarea sensului cuvantului.....	17
Capitolul 3. Metode de Text Mining.....	18
3.1. Information Extraction .....	18
3.1.1. Extragerea entitatilor .....	18
3.1.2. Determinarea relatiilor intre entitati.....	20
3.1.3. Aplicarea tehnicilor de machine learning.....	20

3.2. Clasificare / Categorizare .....	22
3.2.1. Clasificatorul Bayesian Naiv .....	23
3.2.2. Clasificatorul celui mai apropiat vecin .....	24
3.2.3. Arbori decizionali .....	25
Capitolul 4. Aplicatia practica .....	26
ANEXE .....	44

## Lista figuri

Figură 1 Lista de stop words romanesti.	15
Figură 2 Exemple completare sabloane	21
Figură 3 Parte din setul de antrenare pentru convorbirile cu utilizatorul	27
Figură 4 Parte din setul de antrenare pentru realizarea de preziceri	27
Figură 5 Rezultatul procesului de tokenizare, si lematizare a datelor	29
Figură 6 Adaugarea datelor preprocesate in setul de antrenare	30
Figură 7 Realizarea straturilor	30
Figură 8 Rezultatele primelor 10 compilari	31
Figură 9 Rezumatul modelului	32
Figură 10 Pregatirea datelor pentru estimarea autorului	32
Figură 11 Realizarea modelului si primele 10 compilari	33
Figură 12 Functia curatare	34
Figură 13 Functia bow	34
Figură 14 Functia cea_mai_buna_alegere	35
Figură 15 Model 1 conversatie cu chatbot	35
Figură 16 Model 2 conversatie cu chatbot	36
Figură 17 Model 3 conversatie cu chatbot	36
Figură 18 Functia check_math	36
Figură 19 Rezultate ale functiei math	37

Figură 20 Functia bow_bac	37
Figură 21 Functia cea_mai_buna_alegere_bac	38
Figură 22 Functia cele_mai_dese_cuvinte	38
Figură 23 Frecventele celor mai folosite cuvinte in documentul despre ursii panda	39
Figură 24 Frecventele celor mai folosite cuvinte in documentul despre chatbots	40
Figură 25 Histograma posibilitatilor de aparitie a autorilor la examenul de bacalaureat	41

# Introducere

Alegerea temei a fost facuta pornind de la importanta temei – Text Mining – si de la dorinta de a patrunde in particularitatile acestei teme. Din materialul pe care l-am studiat, am constatat ca aceasta tema este foarte des abordata in viata de zi cu zi, si ca nivelul ei de importanta trebuie subliniat. Prin aceasta lucrare doresc sa aprofundez aplicabilitatea acestei teme. Acesta este motivul pentru care tot ceea ce urmeaza este o incercare de a pune in practica attributele acestei ramuri a domeniului de Data Science. Voi incerca prin intermediul aplicatiei care consta atat in parte teoretica, cat si in exemple, sa popularizez aceasta arie noua si intr-o continua dezvoltare. Domeniul de aplicabilitate al Text Mining cuprinde domenii ca tranzactii comerciale, financiare, dar merge si spre domeniu social (retele sociale).

Text Mining se referă la acea ramura a Data Mining-ului care preia informații din diferite surse, precum cărți, articole, documente, arhive, datele unor compani, conversații între utilizatori online si proiecte, și sub diferite forme, precum formate text, și care foloseste diferite metode care permit interpretarea unei cantități mari de informații cu scopul de a extrage informații complet noi și de o calitate superioară, care se ascunde la prima vedere printre rânduri. Prin realizarea unei analize de Text Mining, putem trage ipoteze și concluzii noi, pe care ulterior le putem trece prin diferite alte teste și analize pentru a realiza noi adevăruri în diferite domenii de activitate.

Data mining este și ea la rândul său o ramură a altei științe ( computer science ), care se ocupă cu descoperirea de reguli și informații, ce se regăsesc în seturi mari de date și care ulterior pot fi interpretate prin diferite metode, majoritatea fiind de tip statistic.

În realizarea unei analize de tip Text Mining stau la bază concepte precum machine learning si lingvistică computațională, lucru pe care îl are in comun cu data mining-ul. Similitudinea dintre acestease termină în momentul în care facem diferența dintre tipurile de informații pe care cele două analize le oferă și locul de unde cele doua analize își iau datele. În timp ce analiza data mining preia informație din baze de date structurate si o modeleaza astfel încât să poată fi folosită mai târziu în alte analize și pentru a trage concluzii deja existente, precum realizarea unor modele de estimare statistică, scopul analizei Text Mining este de a face descoperiri și deducții, bazându-ne pe informație cunoscută pentru a ajunge la informație necunoscută, această informatie cunoscută fiind

reprezentată de limbaj natural (*natural language*) și fiind mai apoi reprezentată sub forma unui *pattern*.

Orice analiză de Text Mining implică utilizarea anumitor proceduri pentru a transforma text din forma sa cea mai crudă, în informație utilă, care poate mai apoi fi supusă diferitelor analize, depinzând de fiecare context în parte prin folosirea de algoritmi analitici printre care se numără și *natural language processing*. Printre aceste proceduri se numără procesele de *information retrieval* și *information extraction* pentru a face rost de date, *pattern recognition* pentru identificarea de regularități în date, metode de vizualizare și de creare de predicții și algoritmi de căutare a frecvenței în distribuții de cuvinte, precum *lexical analysis*, algoritmi de clusterizare, care se refera la crearea de grupuri de date nestructurate ale caror informatii au legaturi mai stranse in interiorul grupurilor decat intre doua informatii din doua grupuri diferite si tehnologie de baze de date.

Termenul de *natural language processing* (NLP) se referă la domeniul din cadrul inteligenței artificiale care face legătura între tot ce înseamnă particularitățile limbii umane ( expresii, elemente de substrat, suprastrat si adstrat, lingvistică istorica etc. ) și un computer. Altfel spus, NLP se ocupă cu realizarea de tehnologie care poate face un computer să înțeleagă limba umană și să o organizeze.

O analiză de tip Text Mining poate fi interpretată în mai multe feluri, și depinzând de anumiți factori și contexte în care fiecare se aplică, avem mai întâi crearea procesului de extragere de informație, sau *Information extraction*, care are la bază tehnologia de *natural language processing*, întrucât, în realizarea acestuia se folosesc date nestructurate sau semi-structurate regăsite în surse scrise de oameni, apoi avem *knowledge discovery in databases*, sau KDD, care este o metodă analitică de extragere si procesare a informației utile din date structurate în baze de date, și în final, Text Mining se mai poate interpreta și ca text data mining, deoarece putem aplica algoritmi din domeniile de machine learning și din statistică.

Inteligența artificială conversațională , sau chatbot-ul , este un program de calculator care folosește *natural language processing* și inteligență artificială pentru a putea simula o conversație umană și are scopul de a fi o interfață ce poate comunica în mod eficient și clar cu clienți pentru a le oferi informații în cadrul a mai multor domenii de activitate ale firmelor care folosesc un astfel de program.

# Capitolul 1. Definiții și abordări

Înainte de a putea realiza o definiție completă a termenului „Text Mining”, trebuie să cunoaștem proveniența unor termeni mai de profunzime.

Așa cum am menționat mai sus, Text Mining este o ramură a Data Mining-ului, însă, în crearea unei astfel de analize, avem nevoie de procesul de descoperire a cunoștințelor în baze de date, sau *knowledge discovery in databases*, în totalitatea sa sau doar bucăți din el. Astfel, pentru realizarea unei definiții complete a conceptului de Text Mining, avem nevoie să definim acești termeni și câțiva din termenii adiacenți.

## 1.1. Descoperirea cunoștințelor în baze de date ( KDD )

Precum Data Mining, KDD este o ramură a computer science, dar prin intermediul căreia se extrag relațiile și informațiile ascunse din setul de date, aceste informații având caracter de utilitate, validitate și pot ajuta la descoperirea de regularități în date. De obicei datele sunt structurate în baze de date sau date în fișiere text. Acest proces urmează următorii pași (Hotho et al., n.d.):

**1. Înțelegerea și definirea scopului:** Pentru a putea începe să realizăm un astfel de proces, trebuie să cunoaștem foarte bine domeniul de activitate și motivul pentru care realizăm această lucrare.

**2. Înțelegerea și alegerea datelor:** În mod evident, procesul de descoperirea cunoștințelor folosește date, iar noi cei care modelează trebuie să știm ce reprezintă datele, că ulterior să putem alege seturi semnificative de date și să le putem folosi.

**3. Pregătirea datelor:** Acest proces se referă la ștergerea informațiilor irelevante ( numite și *noise* ), pregătirea pentru tratarea excepțiilor și găsirea informației utile, care reprezintă baza informației modelului.



**4.Transformarea datelor:** Pasul imediat următor după acesta este utilizarea algoritmilor de dată mining, iar pentru a putea face asta, datele trebuie să aibă anumite forme, pe care le transformăm la acest pas prin funcții, folosirea anumitor caracteristici ale datelor, anumitor attribute, lucruri în comun, etc.

**5. Data Mining:** La acest pas se realizează extragerea de informații cu un grad mare de utilitate, pentru a putea fi interpretate mai târziu folosind alte modele, algoritmi, sau orice alt proces necesar depinzând de fiecare analiză în parte.

**6. Identificarea de cunoștințe și interpretarea lor:** După ce au fost aplicați algoritmi de dată mining, se extrag informații, care sunt puse în contextul analizei, iar interpretările lor generează cunoștințe, care ulterior pot ajuta la luarea de decizii

## **1.2. Data Mining**

Avem deja o definiție destul de clară asupra acestui concept, însă, încă nu am discutat în detaliu despre metodele pe care acest proces le folosește.(Vijay Gaikwad, 2014)

Dată mining folosește baze de date, descris mai sus, iar pe lângă asta, folosește metode de machine learning, precum Text Mining-ul, și metode statistice.

**Machine learning** = Știință care se ocupă cu crearea de programe care permit sistemelor de calcul să învețe și să creeze conexiune pentru a putea se putea perfecționa. În realizarea unui algoritm de machine learning sunt necesare niște seturi de antrenare, care permite calculatorului să învețe având în față lui un model, pentru a putea ajunge singur la niște concluzii. A nu se confunda cu inteligență artificială. În timp ce machine learning este un tip de inteligență artificială, inteligență artificială nu este neapărat un sistem de machine learning.

**Statistică** = Știință care se ocupă cu analizarea de date reale din diferite domenii prin intermediul calcului matematic și interpretarea acestora prin diferite metode grafice.

### 1.3. Text Mining

După realizarea unei descrieri succinte despre KDD și Dată Mining, acum putem vorbi în amănunt despre Text Mining și particularitățile acestui domeniu.

Este recunoscut la nivel mondial faptul că un procent foarte mare din informațiile unei firme (majoritatea studiilor realizate ajung în jurul valorii de 80%) sunt nestructurate și constau în diferite forme de date, precum documente, email-uri și rapoarte, care sunt documente de tip text. Datorită acestui fapt, se crede că dată mining-ul ar putea decădea în fața Text Mining-ului din punct de vedere al potențialului comercial. Pe de altă parte, procesul de dată mining este unul mai ușor de realizat decât acela de Text Mining din cauza caracterului neclar al datelor pe care procesul de Text Mining le folosește în realizarea sa. (H.-C. Yang & Lee, 2005)

Text Mining utilizează un proces numit *parsing*, care folosește algoritmi de *natural language* pentru a analizășiruri de caractere și pentru a încerca să realizeze legături și înțelesuri pentru propozițiile încadrate în date. Aici putem aduce vorba de conceptul menționat mai devreme de lingvistică computațională, astfel, procesul de parsing în acest context se referă la procesul care realizează un parsing tree, în care se descriu relațiile semantice dintre cuvinte. Acest proces este descris în continuare în capitolul 2.3.2.

Text Mining se întâlnește cu diferite probleme când vine vorba de extragerea de date și în obținerea de informații noi, iar acestea pot fi rezolvate prin aplicarea informațiilor din domeniul științelor de *Natural language processing (NLP)*, *Information Retrieval (IR)* și *Information Extraction (IE)*, alături de algoritmi de dată mining, cu scopul de a putea găsi în interiorul datelor caracteristici și proceduri specifice domeniului din care sunt extrase acestea. (*Text Mining: The State of the Art and the Challenges*, 2000)

**Information retrieval** este procesul de căutare a datelor într-un document și metadatele acestor date, precum și căutarea bazelor de date și a documentelor în sine. Hearst (1999) definește IR ca fiind un proces din care reies documente care răspund la întrebările puse, nu răspunsurile în sine. Acest proces conține algoritmi de căutare a similitudinilor și compararea cu documente existente, căutând să aibă un grad cât mai ridicat de relevanță. Această știință este una relativ îmbătrânită, comparativ cu nivelul tehnologiei actuale, care crește cu o rată exponențială, primele modele automate realizându-se în 1975 de Salton, motiv pentru care se spune că procesările clasice

ale limbajului natural au un nivel mai ridicat de complexitate și ajung mai în profunzime, comparativ cu information retrieval.

**Information extraction** se folosește pentru a scoate doar anumite informații din date de tip text și este definit de Appelt (1999) că și sarcina de a face parte din datele de intrare ale modelelor de baza pentru limbajul natural. Această este o metodă de Text Mining și va fi explicată în detaliu în capitolul 3.

## **Capitolul 2. Preprocesarea textului si metode**

Din cauza mărimii mari pe care le au documente ce sunt folosite pentru o analiză Text Mining, avem nevoie să structurăm datele astfel încât să ajungem la o formă a informației care este mai ușor de procesat ulterior. Pentru asta, marea majoritate a programelor au nevoie de niște cuvinte sau expresii cheie, numite keywords, care ajută la stabilirea, pe scurt, a subiectului unui document.(H. C. Yang & Lee, 2005)

În multe cazuri, aceste cuvinte cheie sunt alese de specialiști sau de autor, care ar trebui să analizeze textul și să aleagă cuvintele care descriu cel mai bine textul, caz în care apare un oarecare nivel de subiectivitate umană, motiv pentru care algoritmi de căutare a cuvintelor cheie s-au dezvoltat astfel încât la final să sugereze sau chiar să ofere cele mai potrivite alegeri, lucru care în mod normal ar fi un proces foarte îndelungat.

### **2.1. Modelul spațiului vectorial (Vector space model)**

Modelul spațiului vectorial (Salton, McGill, 1983) preia diferite tipuri de obiecte, precum documente text, și le encodeaza pentru a deveni elemente ale spațiului vectorial. Desi acest

model folosește un tip de date relativ simplu, fără a mai adăuga alte filtre semantice, el se dovedește a oferi rezultate analitice foarte eficiente. (Hotho et al., n.d.)

Vectorii numerici ce reies din acest model pot avea următoarea formula:

$$w(d) = (x(d, t_1), \dots, x(d, t_m))(1)$$

Fiecare document respectă această formulă, motiv pentru care acestea se pot compara folosind proprietăți de baza ale vectorilor precum suma vectorilor a și b este egală cu suma vectorilor b și a. Documentele se pot mai apoi compara și ordona folosind funcția de similaritate (Salton et al. 1994). Acest tip de reprezentare se folosește pentru a se putea găsi o metodă de codificare potrivită.

Valoarea d se referă la documentul respectiv, t înseamnă termen iar m se referă la numărul de cuvinte al unui document (cu excepția unor posibile apariții de grupări de cuvinte precum expresii sau cuvinte legate prin ”-” ), deci fiecare element din vector este un cuvânt. Există mai multe tipuri de codificări, însă cea care se bazează pe căutarea cuvintelor cu importantă cea mai mare este *term weighting*, unde o ”greutate” mare reprezintă o importantă mai ridicată a cuvântului respectiv în cadrul documentelor mai importante și nu se aplică neapărat la toate documentele folosite (Salton, Buckley 1988).(Hotho et al., n.d.)

Pentru a calcula greutatea fiecărui cuvânt, avem formulă:

$$w(d, t) = \frac{tf(d, t) * \log\left(\frac{N}{n_t}\right)}{\sqrt{\sum_{j=1}^m tf(d, t_j)^2 * \left(\log\left(\frac{N}{n_{t_j}}\right)\right)^2}}, (2)$$

unde N este numărul de documente,  $n_t$  este numărul de documente din colecție care conține termenul t, funcția tf vine de la term frequency iar  $\log(N/n_t)$  este inversă frecvenței documentului, notată idf(t).

Formulă pentru calcularea similitudinii între două documente d1 și d2 are la baza produsul dintre produsul interior al valorilor vectorilor greutăților:

$$S(d_1, d_2) = \sum_{k=1}^m w(d_1, t_k) * w(d_2, t_k). \quad (3)$$

Din acest punct există mai multe formule matematice ce se pot aplica, precum distanță euclidiană între două documente (calculată ca radicalul sumei dintre pătrățele diferențelor interioare de greutate pentru fiecare termen din cele două documente), depinzând de direcția în care se dorește să se continue. Deoarece documente au numere diferite de cuvinte, se aplică normalizarea acestora, pentru că lungimea lor să devină egală cu 1.(Hotho et al., n.d.)

$$dist(d_1, d_2) = \sqrt{\sum_{k=1}^n (w(d_1, t_k) - w(d_2, t_k))^2} \quad (4)$$

## 2.2. Preprocesarea textului in Natural Language Processing

Procesarea textului sau *Text Processing* folosește comenzi ale calculatorului pentru a putea cauta, modifica, manipula, filtra și genera raportul conținutului unui document. Acest proces constă în câțiva pași cu ajutorul cărora informația crudă se transformă într-o informație pe care mai târziu o putem folosi în modele, iar performanțele acestor modele sunt determinate de cât de mulți pași sunt aplicați și cât de bine. În realizarea acestui proces, voi defini termenul  $T=(t_1, t_2, \dots, t_n)$  ca fiind dicționarul tuturor documentelor care sunt folosite.(Liddy, 2001)

### 2.2.1. Transformarea literelor mari in litere mici

Acest proces este unul simplu, dar necesar pentru a putea obține un set neted de date și pentru a putea reduce mărimea dicționarului, întrucât calculatorul percepe cuvintele care au litere mari diferite de aceleași cuvinte, dar toate literele sunt mici (de exemplu, cuvintele "Proces" și „proces” vor fi interpretate ca două cuvinte diferite).

### 2.2.2. Tokenizare

Procesul de tokenizare este utilizat pentru a putea obține toate cuvintele din dicționar, scoțându-se toate semnele de punctuație și elemente ce nu au legătură cu textul (de exemplu, tabs, line breaks, caractere nule, etc.) și astfel realizându-se un strâng de date, în care fiecare cuvânt este

considerat că fiind un token. Acest proces permite unui sistem de calcul atât să poată lucra cu fiecare cuvânt în parte și să îi găsească frecvența, cât să găsească și locurile și contextele în care acesta tinde să apară.

Cele trei tipuri principale de tokenizare sunt tokenizarea după cuvinte, după caractere și *sub word tokenization*. Ce am descris mai sus intră în categoria tokenizării după cuvinte, întrucât calculatorul construiește un string de cuvinte și le delimitează prin separatori precum virgulă. Avantajele acestui tip de tokenizare sunt că este cel mai simplu și aplicabil model, comparativ cu tokenizarea după caractere, care numără cât de des apare fiecare litera dintr-un anumit alfabet în documentul text folosit, sau *sub word tokenization*, care, pe lângă cuvinte, împarte și afixele cuvintelor, un proces de durată care nu are aplicabilitate în afara contextelor lingvistice, unde identificarea funcționalității unui cuvânt este mai importantă decât sensul cuvântului în sine.

### **2.2.3. Eliminarea elementelor cu semnificație redusă**

Cele două elemente la care se face referință la acest pas sunt cuvintele de legătură, pronume, adverbe, etc. numite și *stop words*, și caracterele care formează mulțimea de punctuație, adică elementele aflate în șirul de caractere: " !"#\$%&'()\*+,-./:;<=>?@[\\]^\_`{|}~ ". Deși aceste elemente sunt importante în coerența textului, ele nu aduc un nivel mare de semnificație în rezultatele finale ale modelelor.

abia acea aceasta această aceea aceeași acei acela acel același acele acelea acest acesta aceste acestea acestei acestia acestui  
acești aceștia adică ai aia aibă aici al ala ale alea alt alta altceva altcineva alte altfel alți alții altul  
am anume apoi ar are as așa asta astea astfel asupra atare atât atata atatea atatia atî atît atita atitea  
atîția atunci au avea avem aveți avut aș ați ba ca cam când care careia carora caruia cat către ce  
cea ceea cei ceilalți cel cele celor ceva chiar ci cînd cine cineva cit cita cite citeva cîți citiva cu  
cui cum cumva cît cîte cîtva cîți cînd cît cite cîtva cîți cã cîci cărei căror căruî către da dacă  
dacă dar dat dată dau de deasupra deci decît deja deși despre deși din dintr dintr- dintre doar doi doilea  
două drept după dă e ea ei el ele era eram este eu ești face fara fata fel fi  
fie fiecare fii fim fiu fiți foarte fost fără i ia iar îi îl imi în înainte înapoi încă încît  
însa într între își itî la le li lor lui lângă lingă m ma mai mea mei mele mereu meu  
mi mie mine mod mult multa multe multi multă mulți măine mîne mă ne ni nici nimeni nimic niste niște  
noastre noastră noi noștri nostru nou noua nouă noștri nu numai o or ori oricare orice oricine oricum oricînd oricît  
oricînd oricît oriunde pai parca patra patru pe pentru peste pic pîna poate pot prea prima primul prin printr- puțin  
puțin puțină puțină pînă pînă sa să-mi sa-ti săi sale sau se si sint sintem spate spre sub sunt suntem  
sunteți sus să săi său t ta tale te ti tine toata toate toată tocmăi tot totî totul totuși totuși  
toți trei treia treilea tu tuturor tăi tău u ul ului un una unde undeva unei uneia unele uneori unui  
unor unora unu unui unuia unul v va vi voastre voastră voi vom vor vostru vouă voștri vreo vreun vă

Figură 1 Lista de stop words romanesti.

## 2.2.4. Stemming si lematizare

Stemming este procesul prin care se elimina părți ale cuvintelor și se încearcă să se ajungă la rădăcina să, adică formă nederivata a cuvântului. Acest proces de obicei implică aducerea cuvintelor la formă lor de singular sau eliminarea afixelor precum sufixul ”ing” din cuvântul ”going”.

Există cazuri în care această metodă rezultă în cuvinte inexistente în vocabular. De exemplu, dacă avem cuvântul ”analysis”, care prin procesul de stemming se transformă în ”analysi”, crezând că ultimul ”s” se referă la formă de plural , lucru care, într-un alt context ar fi corect, însă aici reiese un cuvânt fără nicio semnificație.

Lematizare sau *Lemmatization* este un algoritm prin care se caută lema unui cuvânt, adică formă să de dicționar, încercând să nu se piardă sensul inițial al cuvântului din cadrul documentului. Un exemplu bun pentru acest algoritm este cuvântul ”teeth”, care, după lematizare, se transformă în cuvântul ”tooth”.

Cele două procese sunt în strânsă legătură, însă, există diferențe între acestea, care se pot observa cel mai bine în contextul procesării cuvântului "Caring". Algoritmii de stemming ar arăta cuvântul "Car", iar lematizarea arată "Care". Punctele forte ale algoritmilor de stemming sunt după cum urmează: rapiditatea și funcționalitatea care se aplică pe cuvinte care nu se abat foarte mult de la reguli standard de lexic. Algoritmii de lematizare oferă întotdeauna cuvinte care există în vocabular, comparativ cu stemming, însă aceștia au un nivel de complexitate mai mare, având în spatele lor un vocabular întreg pe care îl folosesc în procesul de căutare și asociere.

## **2.3. Preprocesarea lingvistica**

Pe lângă metodele deja explicate, în unele cazuri este necesară o preprocesare din punct de vedere lingvistic, pentru a oferi un nivel mai mare de înțelegere a elementelor unui text, însă această are și limitări, deoarece, în funcție de context, preprocesarea clasică este suficientă, iar aplicarea preprocesării lingvistice nu este necesară. Câteva dintre metodele de preprocesare lingvistică sunt:

### **2.3.1. Identificarea partii de vorbire**

Numită și part-of-speech tagging (PoS), este un proces important din cadrul NLP care atribuie unui cuvânt o parte de vorbire. Determinarea părții de vorbire corectă este un proces care vine cu provocări, deoarece același cuvânt poate avea două părți de vorbire diferite, motiv pentru care, în realizarea acestui proces, se analizează cuvântul, locul sau în propoziție și cuvintele care îl înconjoară. Spre exemplu, în propoziția "Mă duc în oraș să cumpăr haine", cuvântul "haine" primește atributul de substantiv, dar dacă avem propoziția "Hienele sunt animale haine", cuvântul este scris la fel, însă nu mai poate fi considerat substantiv, motiv pentru care este necesară analizarea cuvintelor învecinate.



### **2.3.2. Parsing**

Acest proces generează un parsing tree, care descrie relațiile din punct de vedere semantic dintre cuvinte. Cu ajutorul acestui parsing tree, putem să ne dăm seama de structura gramaticală a propozițiilor și a sintaxei acesteia. Spre exemplu, dacă avem propoziția "Ana are mere", procesul de parsing utilizează tokenizarea și propoziția se împarte în cuvintele "Ana", "are" și "mere". Apoi, fiecăruia din aceste cuvinte i se atribuie o parte de vorbire folosind PoS, astfel, cuvintelor "Ana" și "mere" li se însușește atributul de substantiv, iar pentru cuvântul "are", atributul de verb.

### **2.3.3. Chunking**

Procesul de chunking consta în fragmentarea unei propoziții după părțile de vorbire ale cuvintelor ce o alcătuiesc. Acesta folosește PoS tagging și are ca scop ușurarea identificării părții de baza a uneia sau mai multor propoziții, unde sunt folosite multe cuvinte pentru a descrie în amănunt diferite situații sau concepte, de unde ne interesează doar anumite parti. De exemplu, propoziția : "Se recomanda să mâncam cereale integrale de câteva ori pe săptămână și macar un fruct în fiecare zi", vă grupa substantivele astfel : "cereale, săptămână, fruct, zi". Luam acest exemplu și îl aplicăm unui întreg articol legat de ce este recomandat să mâncam atunci când ne dorim să ținem o dieta. După ce avem toate substantivele grupate, putem să ne dam seama, în termeni largi, ce este recomandat să mâncam. Dacă printre substantivele se afla "...castraveți, morcovi,..., gogoși...", datorita contextului în care ne aflăm, este evident că ni se recomanda să mâncam castraveți și morcovi, și să încercăm să evitam gogoșile.

### **2.3.4. Dezambiguarea sensului cuvântului**

Dezambiguarea sau Word Sense Disambiguation (WSD) se referă la procesul de identificare a sensului unui cuvânt în contextul în care acesta este folosit. Fiecare sens al cuvintelor trebuie luate în considerare în realizarea dicționarului de cuvinte, lucru care aduce la încărcarea acestuia, însă, șansă că un cuvânt să nu fie interpretat corect scade considerabil. Această metodă se observă cel mai

bine în exemplul cu cele două sensuri ale cuvântului ”haina”. Toate aceste metode sunt folosite pentru ajungerea la o înțelegere cât mai ușoară de către calculator a setului de date folosit, ele construindu-se una pe baza celeilalte.

## **Capitolul 3. Metode de Text Mining**

Metodele de data mining realizează structuri folosind cantități mari de date, care facilitează semnificativ procesele de modelare și analizare ulterioare aplicate asupra datelor. Dezavantajul acestora este faptul că procesul de alegere și menținere al indexilor este unul ce se poate dovedi a ocupa foarte mult timp, lucru care îi face să devină inutili, întrucât ei descriu un moment care nu mai este valabil, așa că nu pot fi folosiți în contexte în care datele sunt într-o continuă schimbare. Cele mai cunoscute metode de Text Mining sunt Information extraction, Clustering, Summarization, Classification, Visualization, Topic discovery și Topic extraction.

### **3.1. Information Extraction**

Extragerea informațiilor (IE) este procesul de extragere a datelor sau informațiilor sub o formă structurată din text nestructurat prin căutarea aparițiilor entităților ce fac parte din textul original și a relațiilor dintre aceste entități. Sistemele IE pot fi folosite pentru a extrage direct informații abstracte din texte sau pentru a extrage date concrete dintr-o colecție de documente, care pot fi mai apoi studiate în continuare cu metode convenționale de data mining pentru a descoperi patternuri. Texte scrise de om pot avea informații care nu sunt întotdeauna ușor de procesat de un computer, din cauza folosirii unei structuri care necesită o înțelegere și folosire a limbajului natural foarte profundă. Având această înțelegere a limbajului natural, sistemele IE pot fi folosite în aceste situații, unde textul este structurat într-o manieră în care avem nevoie de anumite nivele de cunoștințe semantice.(Hobbs, n.d.)

Pentru a se putea dezvolta tehnologia pentru a se crea sisteme îmbunătățite de IE, DARPA (Defense Advanced Research Projects Agency) au inițial și finanțat Conferințe de Înțelegere a Mesajelor (MUC). Au fost realizate 7 astfel de conferințe, denumite de la MUC-1 la MUC-7, prima

avand loc in 1987 si fiecare a avut un alt domeniu. MUC-1 si 2 au avut ca si sursa a textului operatiuni militare, subiectul fiind operatiuni ale flotei. Restul conferintelor au folosit ca si textul sursa raportaje de la stiri in diferite domenii, precum activitati teroriste in America latina, Corporate Joint Ventures, productia microelectronica, negocierea litigiilor de muncă și succesiune în conducerea corporativă, rapoarte de zboruri prabusite si lansari de rachete.

Procesul de extragere a informatiei nu este unul care poate fi realizat de oameni, deoarece, cunostiintele din domeniile respective pot influenta datele, in special pentru ca acestea trebuiesc analizate separat.

### **3.1.1. Extragerea entitatilor**

Găsirea formulărilor lingvistice care reprezintă termeni reali sau “entități” este primul pas în realizării unui sistem IE și este o sarcina foarte des întâlnită. Acești termeni, care de obicei sunt alcătuiți din cuvinte sau adunări de cuvinte, sunt considerați că elemente independente de vocabular, iar identificarea acestora sporesc eficientă procesării documentelor. Câteva exemple sunt nume de oameni, termeni formați din mai multe cuvinte, date calendaristice, numele firmelor, abrevieri, acronime și adrese de mail.

Acest sistem funcționează după anumite reguli de aplicarea unei etichete. Unul din acestea este sistemul IOB, folosit de “The Conference on Computational Natural Language Learning” (CoNLL): Pentru fiecare cuvânt din text, se aplică eticheta “B”, dacă se știe că acesta este la începutul unei entități cunoscute, eticheta “I”, dacă se află în interiorul entității, și “O” dacă se află în afară acesteia. De exemplu, după folosirea sistemului IOB, o propoziție va fi etichetată astfel: “Maria (B) are (O) o (O) prietenă (B) în (O) New (B) York (I)”.

### **3.1.2. Determinarea relatiilor între entitati**

Acest lucru se realizează prin utilizarea un proces de parsing sintactic. Găsirea structurii predicate a unui grup mic de propoziții predeterminate este o problema tipică a sistemelor IE, când vine vorba de relațiile simple între entități. Tehnici de shallow parsing sunt suficiente pentru a rezolva probleme de genul, comparativ cu aplicarea unui sistem de parsing complet. O altă problema greu de rezolvat care poate apărea este ambiguitatea de coreferință, care apare în momentul în care mai multe substantive fac referință la aceiași entitate.

### **3.1.3. Aplicarea tehnicilor de machine learning**

După prelucrarea datelor și identificarea entităților și relațiilor dintre acestea, se aplică tehnici de machine learning, care, din acest punct, pot învăța să folosească informația dată pentru a găsi soluția corectă la o problema pusă în față, sau pentru a identifica reguli noi.

În cadrul găsirii de soluții, un exemplu este completarea de șabloane folosind reguli de potrivire între ce se cere și ce trebuie afișat pentru a completa spații goale în șabloane(exemplu).(Witten, n.d.) Că exemplu, folosim o schemă în care sunt prezentate perechi alcătuite din ,documente și șabloane completate, pe post de date de antrenare, la care apoi se aplică un algoritm de învățare bottom-up pentru a detecta reguli de găsim a locurilor unde trebuie completat șablonul, folosindu-ne de restul scrisului. Regulile sunt reprezentate în formă de pattern-acțiune iar pattern-urile conțin constrângeri ( precum partea de vorbire sau clasele semantice) pentru cuvintele din jurul contextului și pentru cele care completează locurile lipsa.(Mahesh, 2018)

Text	Sablon
Propozitia 1: John is an engineer and lives in Bucharest, Romania. He likes to watch movies.	Entity: John Job: engineer Country: Romania Hobby: movies
Propozitia 2: Olivia works as a developer and he lives with her family in Madrid, Spain. She enjoys playing the piano.	Entity Olivia Job developer Country Spain Hobby piano
Propozitia 3: Roger has a programmer job and resides in Lisbon, Portugal. He loves practicing sports	Entity Daniel Job programmer Country Spain Hobby sports

Figură 2 Exemple completare sabloane

-În acest exemplu, în partea din stangă avem textul oferit pentru interpretare și în dreapta avem șablonul completat. Se pot observa câteva reguli aplicate pentru completarea șablonului:

-Daca exista un substantiv la începutul propoziției sau care are litera mare, acela este numele persoanei.

-Primul substantiv de după primul verb se refera la slujba fiecărei persoane.

-Daca exista doua substantive care au litera mare, iar intre ele se afla o virgula, atunci al doilea este numele tarii unde locuiește fiecare persoana.

-Ultimul substantiv din a doua propoziție se refera la hobby-ul fiecărei persoane.

Procedura urmata pentru a afla reguli noi este de a crea reguli maxime bazate pe fiecare exemplu oferit în setul de antrenare iar apoi acestea se generalizează pe fiecare exemplu în parte. Desi exista diferențe între felurile de scriere ale propozițiilor, toate au în comun faptul că exista doua parti de vorbire la fel, adică doua substantive reprezentate de oraș și țara, care au virgula între ele, și sunt scrise cu prima litera mare. Datorita acestei reguli general valabila, putem să generalizam toate exemplele și să completăm șablonul dat.(Frawley et al., 1992)

### 3.2. Clasificare / Categorizare

Tehnicile de clasificare au ca scop categorizarea documentelor text cu limbaj natural, atribuindu-le clase predeterminate potrivite depinzând de contextul acestora. Categorizarea automată a textului are mai multe aplicații practice, precum indexarea pentru aplicarea de Document Retrieval, identificarea metadatelor, dezambiguarea sensurilor cuvintelor și organizarea cataloagelor cu surse Web. (Patel & Soni, n.d.) De asemenea, până în anii 90', clasificarea textului avea în mod majoritar la baza tehnici ad hoc, care foloseau reguli de clasificare gândite de experți, care erau ulterior implementate în mașini de calcul pentru a fi aplicate în mod automat documentelor noi. De atunci și până în prezent, procesul s-a concentrat pe machine learning, care folosește seturi de antrenare cu documente deja clasificate pentru a găsi reguli pentru a realiza același proces final. Ca și exemplu, dacă avem articole în care se descriu filme, acestea pot avea etichetele de comedie, acțiune, thriller, horror, romanță, etc. Aceste categorii fac parte din setul de "vocabular controlat", care va fi notat cu  $E$ , unde fiecare categoria se notează cu  $e \in E$ . Categoriile se aplică mai întâi elementelor setului de antrenare de date notat cu  $D=(d_1,d_2,...,d_n)$ . Rezultatul final va fi un model de clasificare care trebuie să poată atribui unui document adăugat ulterior o categorie din cele existente. Funcția modelului este de forma  $f(d) = e$ , unde  $f : D \rightarrow E$ .

O parte din documentele folosite pentru crearea funcției modelului sunt necesare pentru calcularea nivelului de performanță al modelului. Pentru calcularea performanței, atribuim manual etichetele potrivite documentelor nefolosite în model și comparăm cu rezultatele acestuia. Documentele atribuite manual comparativ cu restul documentelor reprezintă acuratețea modelului, însă uneori aceasta poate să nu fie semnificativă, motiv pentru care sunt utilizați alți măsurați ai gradului de succes al modelelor, precum precizia și recall-ul. Precizia se referă la partea de documente preluate care sunt relevante, în timp ce recall-ul se referă la partea de documente relevante care sunt preluate.

Clasificatorii determina un anumit grad de apartenență asupra unei categorii sau clase. Dacă se folosesc numai documente cu un grad de apartenență mare pentru clasele țintă, atunci precizia e mare, însă, acest lucru poate determina pierderea din model a unor documente relevante și invers, dacă se realizează o căutare completă, atunci nivelul de recall crește și precizia scade. Scorul F este un compromis asupra acestor două instrumente de măsurat, care ajută la măsurarea performanței totale a claselor și are formula:

$$F = \frac{2}{\frac{1}{\text{precizie}} + \frac{1}{\text{recall}}} \quad (5)$$

Dintre cele mai des întâlnite metode statistice de clasificare, propun Clasificatorul Bayesian Naiv (Naïve Bayesian classifier), Clasificatorul Celui Mai Apropiat Vecin (Nearest Neighbour Classifier) și Arborii decizionali (Decision Tree).

### 3.2.1. Clasificatorul Bayesian Naiv

În realizarea unei familii de clasificatori probabilistici simpli, numiți și clasificatorii bayesieni naivi, se aplică teorema lui Bayes cu ipoteze puternice de independență între descriptori (ipoteze naive). În contextul Text Mining-ului, presupunem că fiecare cuvânt dintr-un document  $d_i$  este generat de un mecanism probabilistic și că acestea au un anumit nivel de legătură cu clasele  $e(d_i)$ . Formulă bayesiană ne arată probabilitatea unei clase, având cuvintele documentului. (Mitchell 1997)

$$p(e_a | t_1, t_2, \dots, t_{n_i}) = \frac{p(t_1, t_2, \dots, t_{n_i} | e_a) p(e_a)}{\sum_{e \in E} p(t_1, t_2, \dots, t_{n_i} | e) p(e)} \quad (6)$$

unde:

- $p(t_1, t_2, \dots, t_{n_i} | e(d_i))$  este distribuția de probabilitate a celor  $n_i$  cuvinte, știindu-se clasa.

- $p(e)$  este probabilitatea anterioară și se referă la probabilitatea ca un document aleator, ale căror cuvinte nu se cunosc, să aparțină clasei  $e$ .

- $p(e_a)$  reprezinta partea documentelor de antrenare.

- $p(e_a|t_1, t_2, \dots, t_{n_i})$  este probabilitatea posterioara si se refera la probabilitatea ca toate cuvintele  $t_1, t_2, \dots, t_n$  apartin clasei  $e_a$ . Clasa cu cea mai mare probabilitate posterioara se poate aplica documentului dat.

Formulă pentru probabilitatea cuvintelor, știindu-se clasa  $e_a$ , se poate scrie într-o manieră mai simplă, deoarece s-a demonstrat că nici ordinea cuvintelor în documente nu este foarte importantă, și nici numărul de apariții a anumitor cuvinte din aceste documente clasificate nu depind de apariția celorlalte cuvinte.

$$p(t_1, t_2, \dots, t_{n_i}|e_c) = \prod_{j=1}^{n_i} p(t_j|e_c) \quad (7)$$

Prin presupunerea independențelor descriptorilor că fiind puternice (naive), formulă Bayes va defini Clasificatorii Bayesieni Naivii (Good, 1965) dar modelul este considerat nerealist, însă, clasificările pe care le realizează sunt surprinzător de corecte.

În procesul de clasificare, probabilitățile estimate se folosesc pentru a clasifică noi documente, lucru posibil datorită pașilor de învățare, pentru care este necesară estimarea probabilităților cuvintelor  $p(t_j|L_c)$  în fiecare clasa în funcție de frecvențele relative în documente a seturilor de antrenare etichetate  $e_a$ .

Cu toate acestea, în unele situații, în care categorizarea documentelor de antrenare de mână este o sarcina dificilă și de durată, se pot folosi și documente necategorizate că și set de antrenare. Acest lucru se poate realiza dacă plecăm de la premiza că de la un set mic de antrenare se poate concluziona o corelație puternică între cuvântul  $t_i$  și clasa  $e_a$ , iar dacă același lucru se poate spune și despre două cuvinte  $t_i$  și  $t_j$ , atunci și  $t_j$  poate ajuta la prezicerea clasei  $e_a$ , lucru care în final, ajută la îmbunătățirea procesului de clasificare.

### **3.2.2. Clasificatorul celui mai apropiat vecin**



Această clasificare este o metodă non-parametrică și se bazează pe un mod de gândire mai simplu, comparativ cu celelalte modele. Metodă are la baza ideea că în Text Mining, există multe metode care pot determina similitudini, cea mai simplă fiind numărând câte cuvinte comune sunt în două documente diferite. Problema apare în momentul în care luăm în considerare faptul că în general, cuvintele au un conținut informațional complex, pentru care se aplică formulă similitudinii  $S(d_i, d_j)$

Astfel, se caută în setul de antrenare documente care sunt similare cu documentul pe care vrem să îl adăugăm, iar etichetele acestor documente similare se pot aplica asupra documentului nou. Dacă există  $k$  astfel de elemente, metodă se transformă în *Clasificarea celor mai apropiați  $k$  vecini*.

Pasul următor este să folosim formulă similitudinii  $S(d_i, d_j)$  pe toate documentele  $d_j$  din setul de antrenare pentru a determina dacă un document  $d_i$  aparține clasei  $e_b$ , iar primele  $k$  documente din setul de antrenare care sunt cele mai similare sunt selectate. Se calculează proporțiile claselor, adunând pentru fiecare doar documente de antrenament (vecini) care fac parte din clasa respectivă, pentru a se determina probabilitatea estimată de apariție a fiecăreia, iar clasa cu probabilitatea cea mai mare, este asociată cu documentul nou  $d_i$ . Pentru determinarea numărului  $k$  de vecini optim, este necesar să aplicăm validarea încrucișată după ce adăugăm mai multe documente de antrenament.

Pe de o parte, metodă celor mai apropiați  $k$  vecini obține rezultate foarte bune, indiferent de volumul datelor. Pe de altă parte, dacă se aplică pe cantități mari de date, trebuie să luăm în considerare faptul că această tehnică realizează similitudinea între documentul care trebuie categorizat și toate celelalte documente din setul de antrenare, lucru care determina un număr foarte mare de calcule

### **3.2.3. Arbori decizionali**

Precum clasificatorul celui mai apropiat vecin, arborii decizionali sunt o metodă non-parametrică, care se folosesc de algoritmi de învățare. Aceștia folosesc un set de reguli pentru a realiza o structură ierarhică, asemănătoare cu un copac, aplicând regulile date într-o manieră secvențială pentru a putea trage concluzii. Procesul de învățare al acestei metode implică o strategie de tipul *divide et impera*, folosind un set de antrenare cât mai cuprinzător. În continuare, pentru setul de documente de antrenare clasificate  $A$  se selectează cuvântul  $t_i$ , care ajută la determinarea clasei potrivite documentelor, iar pentru asta putem să folosim formulă pentru *information gain*, pentru a putea structura cele mai importante cuvinte dintr-un document  $t_i$  într-un clasament, făcând astfel procesul mai simplu, mai rapid și mai eficient. Formula este:

$$IG(t_i) = \sum_{c=1}^2 p(e_a) \log_2 \frac{1}{p(e_a)} - \sum_{m=0}^1 p(t_i = m) \sum_{c=1}^2 p(e_a | t_i = m) \log_2 \frac{1}{p(e_a | t_i = m)} \quad (8)$$

Urmărind structura de arbore, următorul pas este împărțirea documentelor din setul  $A$  în două subseturi,  $A_{i+}$ , unde se vor afla documente ce conțin  $t_i$ , și  $A_{i-}$ , documentele unde  $t_i$  lipsește. Acest algoritm este recursiv, deci aceeași împărțire se aplică încontinuu până când majoritatea documentele aparțin aceleiași clase  $e_a$ .

## Capitolul 4. Aplicația practică

În realizarea aplicației mele, am construit un chatbot, care se folosește de inteligența artificială și natural language processing pentru a putea folosi un set de date de antrenare și a-l interpreta, astfel încât să poată oferi cel mai bun răspuns din lista de posibilități.

Chatbot-ul prezintă utilitate, deoarece îndeplinește mai multe funcții, printre care se numără comunicarea eficientă cu utilizatorul, rezolvarea de calcule matematice de bază, realizarea unei analize din care reiese cele mai frecvente cuvinte dintr-un articol care poate fi dat de utilizator și realizarea unei predicții asupra cărui autor din literatura română vă apărea la subiectul al treilea din cadrul examenului de bacalaureat.

În realizarea aplicației mele, am construit un chatbot, care se folosește de inteligența artificială și natural language processing pentru a putea folosi un set de date de antrenare și a-l interpreta, astfel încât să poată oferi cel mai bun răspuns din lista de posibilități.

Chatbot-ul prezintă utilitate, deoarece îndeplinește mai multe funcții, printre care se numără comunicarea eficientă cu utilizatorul, rezolvarea de calcule matematice de bază, realizarea unei analize din care reies cele mai frecvente cuvinte dintr-un articol care poate fi dat de utilizator și realizarea unei predicții asupra cărui autor din literatura română vă apărea la subiectul al treilea din cadrul examenului de bacalaureat.

Vom folosi un fișier de tipul json pentru a putea stoca seturile de antrenare. Seturile de antrenare sunt realizate din dicționare și arată astfel:

```
{
  "optiuni": [
    {
      "subject": "conversation_start",
      "intrebare": ["hello!", "salutations!", "hi", "hey", "what's up"],
      "raspunsuri": ["Hello, User!", "Good day!", "Salutations!"]
    },
    {
      "subject": "name",
      "intrebare": ["what is your name?", "who are you?", "what do they call you?", "how are you named"],
      "raspunsuri": ["My name is PyPy!", "I am PyPy!"]
    },
  ],
}
```

Figură 3 Parte din setul de antrenare pentru convorbirile cu utilizatorul

```
"optiuni_bac": [
  {
    "subject": "eminescu",
    "intrebare": ["eminescu", "luceafarul", "floare"],
    "raspunsuri": ["Mihai_Eminescu"]
  },
  {
    "subject": "creanga",
    "intrebare": ["creanga", "harap"],
    "raspunsuri": ["Ion_Creanga"]
  },
]
```

Figură 4 Parte din setul de antrenare pentru realizarea de preziceri

Fiecare subset de date este alcătuit din 3 elemente: -subiect, care se referă la categoria din care face parte fiecare tip de conversație pe care utilizatorul o poate avea cu chatbot-ul. -întrebare, care se referă la modele de input-uri ale utilizatorului. De aici, chatbot știe cum ar trebui să arate o întrebare dintr-o anumită categorie, și se va folosi de natural language processing pentru a putea căuta alternative ale acestor expresii, pentru a putea în continuare să răspundă corect la întrebarea

pusa. -răspuns: Se refera la răspunsul static pe care chatbot-ul îl vă oferi în momentul în care identifica întrebarea pusă de utilizator. Pentru că chatbot-ul să arate cât mai uman, în momentul în care se determina întrebarea, el vă alege, într-o manieră aleatoare, unul din răspunsurile posibile. Se poate observa aplicarea unei variații a metodei de clasificare, adăugată pentru a putea manevra mai ușor datele.

## Pasul 2. Încărcarea seturilor de antrenare

Pentru a se putea realiza o analiză cât mai eficientă și corectă, trebuie să trecem prin procese de preprocesare pentru a putea minimiza pe cât posibil numărul de cuvinte distincte cu aceeași rădăcină, pentru a putea grupa convenabil datele și pentru a ușura per total procesul de interpretare al datelor.

În acest sens, datele vor fi trecute prin următoarele procese:

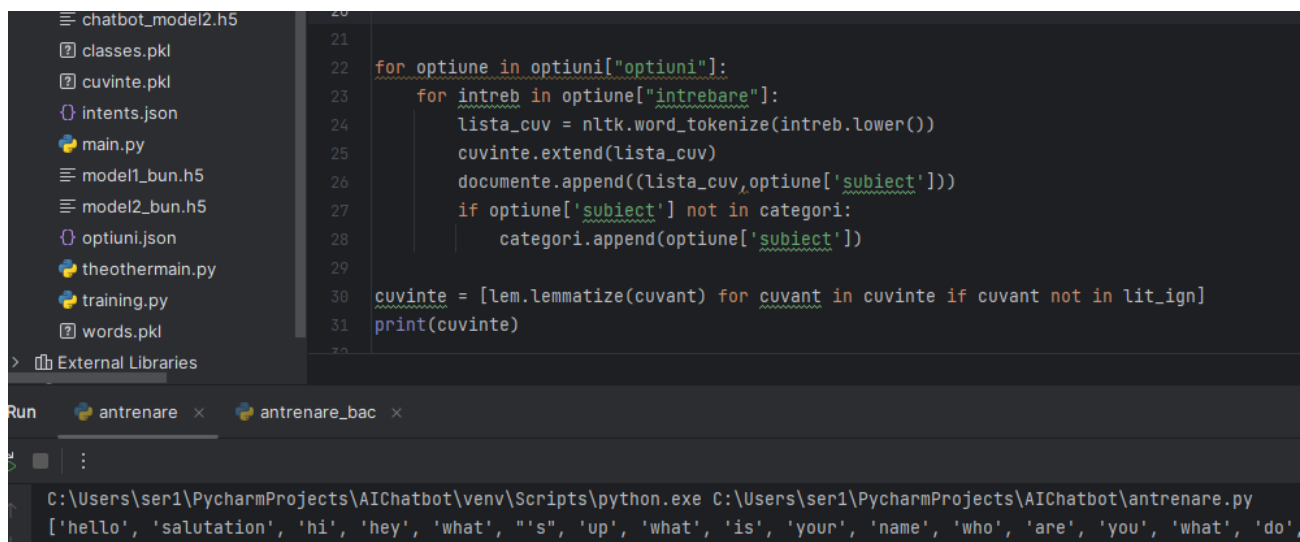
-Folosim funcția `python nltk.word_tokenize` pe întrebările regăsite în setul de antrenare pentru a putea obține toate cuvintele posibile. Aceasta funcție se afla în pachetul `nltk`. La acest pas, cream un dicționar de cuvinte și unul de categorii.

-Folosim funcția `WordNetLemmatizer` pentru a putea reduce pe cât posibil formele cuvintelor, în încercarea de a ajunge la forme cât mai simple și mai apropiate între ele. În acest caz, folosim lematizarea și nu un proces de stemming deoarece acesta este mai eficient și nu vă fi folosit pe cantități foarte mari de date. Funcția `WordNetLemmatizer` se afla în pachetul `nltk.stem`.

-Eliminam cuvintele duplicate din dicționarul de cuvinte folosind funcția `set()`.

-După tokenizarea și lematizarea întrebărilor posibile din cadrul seturilor de antrenare, realizăm dicționare cu perechi de elemente, ce constau în potrivirea fiecărui cuvânt cu o categorie.

-Cu ajutorul funcției `pickle.dump` din cadrul librăriei `pickle`, salvăm documentele realizate ca și file cu extensia `.pkl` (`pickle file`) pentru a le putea apela mai târziu.



```
20
21
22 for optiune in optiuni["optiuni"]:
23     for intreb in optiune["intrebare"]:
24         lista_cuv = nltk.word_tokenize(intreb.lower())
25         cuvinte.extend(lista_cuv)
26         documente.append((lista_cuv, optiune['subiect']))
27         if optiune['subiect'] not in categori:
28             categori.append(optiune['subiect'])
29
30 cuvinte = [lem.lemmatize(cuvant) for cuvant in cuvinte if cuvant not in lit_ign]
31 print(cuvinte)
```

Run antrenare x antrenare\_bac x

C:\Users\ser1\PycharmProjects\AIChatbot\venv\Scripts\python.exe C:\Users\ser1\PycharmProjects\AIChatbot\antrenare.py  
['hello', 'salutation', 'hi', 'hey', 'what', 'is', 'up', 'what', 'is', 'your', 'name', 'who', 'are', 'you', 'what', 'do']

**Figură 5** Rezultatul procesului de tokenizare, si lematizare a datelor

-Vom avea nevoie să implementăm un sistem de bag of words, ce vă ajută rețeaua neuronală să poată înțelege felul în care apar anumite cuvinte în anumite pattern-uri. Pattern-urile sunt alcătuite din lanțuri de 0 și 1, 0 reprezentând cazul în care un cuvânt nu apare într-un sir de cuvinte, iar 1 în cazul în care apare.

-Pattern-urile create se vor adăuga într-un set de antrenare, în care x se referă la valorile efective, iar y se referă la etichete.

-După ce datele au fost preprocesate și pregătite, putem începe să antrenăm rețeaua neuronală.

```

antrenament = []
out_gol = [0] * len(categori)
for document in documente:
    bow = []
    categ_cuv = document[0]
    categ_cuv = [lem.lemmatize(cuvant.lower()) for cuvant in categ_cuv]

    for cuvant in cuvinte:
        if cuvant in categ_cuv:
            bow.append(1)
        else:
            bow.append(0)
    linii=list(out_gol)
    linii[categori.index(document[1])] = 1
    antrenament.append(bow+linii)

random.shuffle(antrenament)
antrenament=np.array(antrenament)

x_antrenament = antrenament[:, :len(cuvinte)]
y_antrenament = antrenament[:, len(cuvinte):]

```

Figură 6 Adaugarea datelor preprocesate în setul de antrenare

### Pas 3. Realizarea rețelelor neuronale

În realizarea rețelei neuronale, ne vom folosi de librăriile tensorflow și keras, care ne permit lucrul cu modele secvențiale simple. Modelul secvențial simplu ajută în realizarea algoritmilor de deep-learning .

-În realizarea rețelelor neuronale avem nevoie de mai multe straturi, care primesc informație de interpretat, realizează niste calcule și întorc informație nouă, transformată, ce va fi folosită ca și informație de interpretat pentru următorul strat.

-În cazul nostru, avem un singur tensor de intrare și un singur tensor de ieșire pe fiecare strat, motiv pentru care putem folosi modelul secvențial simplu.

```

model1 = tf.keras.Sequential()
model1.add(tf.keras.layers.Dense(128, input_shape=(len(x_antrenament[0]),), activation='relu'))
model1.add(tf.keras.layers.Dropout(0.5))
model1.add(tf.keras.layers.Dense(64, activation='relu'))
model1.add(tf.keras.layers.Dropout(0.5))
model1.add(tf.keras.layers.Dense(len(y_antrenament[0]), activation='softmax'))

```

Figură 7 Realizarea straturilor

-Primul strat este unul de tip Dense, care utilizează 128 de neuroni, cu forma variabilei x și cu funcția de activare 'relu' (rectified linear unit).

-Următorul strat este de tip Dropout, care ajuta la evitarea supra-încărcării, prin atribuirea aleatoare a unor unități de intrare cu rata, în cazul afișat, de 0.5, rămânând că restul valorilor de input să fie scalate folosind formula  $1/(1-rata)$ .

-Al treilea strat este asemănător cu primul, folosind în schimb doar 64 de neuroni.

-Stratul următor este de același tip că și stratul al doilea

-Ultimul strat vă folosi la fel de mulți neuroni pe cate subiecte avem și vă folosi funcția de activare 'softmax', care oferă sub forma de procentaj șansele cu care un subiect poate apărea.

-După realizarea straturilor, avem nevoie de o funcție de optimizare, precum cea folosita în problema, SGD (stochastic gradient descent), care ajuta la reducerea calculelor realizate de computer folosind estimății, în locul datelor în sine.

-Ultimul pas este compilarea datelor și repetarea antrenării computerului cu același set de date, de 200 de ori în acest caz.

```
Epoch 1/200
8/8 [=====] - 1s 2ms/step - loss: 2.2052 - accuracy: 0.0833
Epoch 2/200
8/8 [=====] - 0s 1ms/step - loss: 2.2021 - accuracy: 0.0833
Epoch 3/200
8/8 [=====] - 0s 1ms/step - loss: 2.1127 - accuracy: 0.2222
Epoch 4/200
8/8 [=====] - 0s 1ms/step - loss: 2.0298 - accuracy: 0.2778
Epoch 5/200
8/8 [=====] - 0s 1ms/step - loss: 1.9938 - accuracy: 0.4722
Epoch 6/200
8/8 [=====] - 0s 1ms/step - loss: 1.9783 - accuracy: 0.3333
Epoch 7/200
8/8 [=====] - 0s 2ms/step - loss: 1.7870 - accuracy: 0.4167
Epoch 8/200
8/8 [=====] - 0s 1ms/step - loss: 1.8064 - accuracy: 0.4444
Epoch 9/200
8/8 [=====] - 0s 1ms/step - loss: 1.7401 - accuracy: 0.4167
Epoch 10/200
8/8 [=====] - 0s 2ms/step - loss: 1.6829 - accuracy: 0.4444
```

Figură 8 Rezultatele primelor 10 compilari

```

Model: "sequential"
-----
Layer (type)                 Output Shape              Param #
-----
dense (Dense)                (None, 128)              6528
dropout (Dropout)            (None, 128)              0
dense_1 (Dense)              (None, 64)               8256
dropout_1 (Dropout)          (None, 64)               0
dense_2 (Dense)              (None, 9)                585
-----
Total params: 15,369
Trainable params: 15,369
Non-trainable params: 0
-----

```

Figură 9 Rezumatul modelului

Același proces se aplica și pentru a putea realiza interpretări pentru estimarea autorului ce vă apăsă la subiectul III în cadrul examenului de bacalaureat.

```

for optiune in optiuni["optiuni_bac"]:
    for intreb in optiune["intrebare"]:
        lista_cuv = nltk.word_tokenize(intreb.lower())
        bac_cuvinte.extend(lista_cuv)
        bac_documente.append((lista_cuv,optiune['subiect']))
        if optiune['subiect'] not in bac_categorii:
            bac_categorii.append(optiune['subiect'])

bac_cuvinte=sorted(set(bac_cuvinte))
bac_categorii=sorted(set(bac_categorii))

pickle.dump(bac_cuvinte, open('bac_cuvinte.pkl','wb'))
pickle.dump(bac_categorii, open('bac_categorii.pkl','wb'))

antrenament = []
out_gol = [0] * len(bac_categorii)
for document in bac_documente:
    bow = []
    categ_cuv = document[0]
    categ_cuv = [lem.lemmatize(cuvant.lower()) for cuvant in categ_cuv]

    for cuvant in bac_cuvinte:
        if cuvant in categ_cuv:
            bow.append(1)
        else:
            bow.append(0)
    linii=list(out_gol)
    linii[bac_categorii.index(document[1])]=1
    antrenament.append(bow+linii)

random.shuffle(antrenament)
antrenament=np.array(antrenament)

x_antrenament = antrenament[:, :len(bac_cuvinte)]
y_antrenament = antrenament[:, len(bac_cuvinte):]

```

Figură 10 Pregătirea datelor pentru estimarea autorului



```

54 x_antrenament = antrenament[:, :len(bac_cuvinte)]
55 y_antrenament = antrenament[:, len(bac_cuvinte):]
56
57 model2 = tf.keras.Sequential()
58 model2.add(tf.keras.layers.Dense(128, input_shape=(len(x_antrenament[0]),), activation='relu'))
59 model2.add(tf.keras.layers.Dropout(0.5))
60 model2.add(tf.keras.layers.Dense(64, activation='relu'))
61 model2.add(tf.keras.layers.Dropout(0.5))
62 model2.add(tf.keras.layers.Dense(len(y_antrenament[0]), activation='softmax'))
63
64 sgd1 = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
65
66 model2.compile(loss='categorical_crossentropy', optimizer=sgd1, metrics=['accuracy'])
67
68 b=model2.fit(x_antrenament, y_antrenament, epochs=200, batch_size=5, verbose=1)
69 model2.save('model2_bun.h5',b)
70 model2.summary()

```

```

Run antrenare x antrenare_bac x
Epoch 1/200
8/8 [=====] - 0s 2ms/step - loss: 2.7965 - accuracy: 0.1538
Epoch 2/200
8/8 [=====] - 0s 1ms/step - loss: 2.7726 - accuracy: 0.1538
Epoch 3/200
8/8 [=====] - 0s 1ms/step - loss: 2.7926 - accuracy: 0.1282
Epoch 4/200
8/8 [=====] - 0s 1ms/step - loss: 2.7495 - accuracy: 0.1026
Epoch 5/200
8/8 [=====] - 0s 1ms/step - loss: 2.7348 - accuracy: 0.1538
Epoch 6/200
8/8 [=====] - 0s 1ms/step - loss: 2.7685 - accuracy: 0.1026
Epoch 7/200
8/8 [=====] - 0s 1ms/step - loss: 2.7350 - accuracy: 0.1282
Epoch 8/200
8/8 [=====] - 0s 1ms/step - loss: 2.7282 - accuracy: 0.1282
Epoch 9/200
8/8 [=====] - 0s 1ms/step - loss: 2.6825 - accuracy: 0.1795
Epoch 10/200
8/8 [=====] - 0s 1ms/step - loss: 2.6349 - accuracy: 0.2308

```

Figură 11 Realizarea modelului si primele 10 compilari

#### Pas 4. Construirea propriu zisa a chatbot-ului și funcțiilor sale

Chatbot-ul are în realizarea să 4 funcții de baza și 5 funcții ce ajuta la comunicarea cu utilizatorul, prezicerea subiectului al treilea la bacalaureat, la realizarea de calcule matematice de baza și calcularea frecventelor celor mai importante 20 de cuvinte dintr-un document ce poate fi modificat de utilizator. Fiecare din funcțiile sale prezinta niste reguli și recomandări pentru a putea funcționa în condiții optime și pentru a oferi utilizatorului răspunsurile pe care aceasta și le dorește.

-Chatbot-ul raspunde la comenzi în limba engleza, cunoaște informații din domeniu Text Mining-ului și are o gama variata de întrebări din acest domeniu la care poate să răspundă.

-Pentru a apela funcția matematica, se recomanda folosirea spatiilor intre oricare 2 caractere și folosirea doar de cifre și de caractere din mulțimea +,-,\*,/,^ și paranteze rotunde.

-Pentru a-i comanda chatbot-ului să realizeze o analiza de frecventa a celor mai utilizate cuvinte dintr-un document, se recomanda folosirea cuvântului “frequency” sau orice derivat al sau.

-Daca utilizatorul dorește să afle ce prezice chatbot-ul în contextul subiectului III de anul acesta la examenul de bacalaureat, bazat pe articole și păreri extrase din surse online, se recomanda utilizarea cuvântului “article” sau orice derivat al sau.

Funcțiile principale ale chatbot-ului: Funcția curățare are ca parametru întrebările pe care le are utilizatorul pentru chatbot. În aceasta funcție, propoziția este transformată în cuvinte și stocată, iar apoi este lematizată pentru a putea fi comparată cu setul de antrenare utilizat, cu scopul de a găsi cele mai potrivite răspunsuri pentru întrebările utilizatorului.

```
def curatare(prop):  
    cuv_prop = nltk.word_tokenize(prop.lower())  
    cuv_prop = [lem.lemmatize(cuvant) for cuvant in cuv_prop]  
    return cuv_prop
```

Figură 12 Funcția curatare

Funcția bow (bag of words) are scopul de a transforma setul de cuvinte în pattern-uri alcătuite din 0 și 1, aceasta fiind forma reală pe care calculatorul o poate înțelege și de pe urma căreia poate realiza estimări. Aceasta funcție are ca parametrii de intrare numai liste realizate după procesul de tokenizare.

```
def bow(prop):  
    cuv_prop=curatare(prop)  
    bow=[0]*len(cuvinte)  
    for cuv in cuv_prop:  
        for i,cuvant in enumerate(cuvinte):  
            if cuvant == cuv:  
                bow[i]=1  
    return np.array(bow)
```

Figură 13 Funcția bow

Funcția cea\_mai\_buna\_alegere, după cum implica și numele, ajută la selectarea celei mai potrivite opțiuni pentru fiecare întrebare a utilizatorului. Aceasta funcție se folosește de modelul de rețea neuronală explicat la pasul anterior, și cu o eroare de maxim 10%, prezintă prima cea mai

probabila categorie din care face parte propoziția și o introduce într-o lista ce conține numele categoriei și probabilitatea cu care propoziția face parte din aceasta. Funcția are ca parametru de intrare strict șiruri de bag of words, într-ucut folosim funcția `model.predict`, care necesita un format pe care calculatorul să îl poată înțelege.

```
def cea_mai_buna alegere(prop):
    bow1 = bow(prop)
    pred = model1.predict(np.array([bow1]))[0]
    err=0.1
    rez=[[i,r]for i, r in enumerate(pred) if r>err]
    rez.sort(key=lambda x:x[1],reverse=True)
    lista_fin=[]
    for r in rez:
        lista_fin.append({'optiunea': categori[r[0]], 'Probabilitate': str(r[1])})
    return lista_fin
```

Figură 14 Funcția `cea_mai_buna alegere`

Funcția răspuns are ca funcție principală că după se folosească de funcțiile anterioare pentru a identifica din setul de antrenare, valorile de răspuns care corespund cu categoriile din care acestea fac parte. Din aceasta funcție pot reieși mai multe rezultate, depinzând de dorințele utilizatorului. În aceasta funcție de afla răspunsuri pentru conversațiile de baza ale chatbot-ului, rezultatele calculelor matematice și rezultatele predicțiilor pentru bac.

```
<user> Hello!
1/1 [=====] - 0s 72ms/step
<PyPy> Good day!
<user> Who are you?
1/1 [=====] - 0s 17ms/step
<PyPy> I am PyPy!
<user> What can you do?
1/1 [=====] - 0s 17ms/step
<PyPy> I can perform several tasks!
I can perform basic mathematical calculations!
I can have chats with users!
I can predict different outcomes using given articles!
<user> And how are you feeling today?
1/1 [=====] - 0s 19ms/step
<PyPy> I feel great! How about you?
<user> Im doing just fine!
1/1 [=====] - 0s 17ms/step
<PyPy> I am happy to hear that!
<user> Thank you!
1/1 [=====] - 0s 18ms/step
<PyPy> I'm so happy I got to help you!
<user> You are the best bot!
1/1 [=====] - 0s 18ms/step
<PyPy> I feel humbled!
<user> |
```

Figură 15 Model 1 conversație cu chatbot

```

<user> Hello!
1/1 [=====] - 0s 69ms/step
<PyPy> Good day!
<user> How are you doing?
1/1 [=====] - 0s 17ms/step
<PyPy> I feel awesome! You?
<user> I dont feel so good
1/1 [=====] - 0s 18ms/step
<PyPy> You are going to be alright very soon, I am sure!
<user> thank you
1/1 [=====] - 0s 18ms/step
<PyPy> I'm so happy I got to help you!
<user> SO, anyway, what is text mining
1/1 [=====] - 0s 17ms/step
<PyPy> Text mining is the science that is involved in transforming unstructured text data and turning it into new useful data
<user> and what is data mining?
1/1 [=====] - 0s 17ms/step
<PyPy> Data Mining is the process of analysing huge data and extracting useful information from it!
<user> could you tell me what natural language processing is?
1/1 [=====] - 0s 18ms/step
<PyPy> Natural language processing is the science that helps make the connection between human language and computer understanding!
<user> |

```

Figură 16 Model 2 conversatie cu chatbot

```

<user> Hi!
1/1 [=====] - 0s 70ms/step
<PyPy> Salutations!
<user> so, what are some text mining methods?
1/1 [=====] - 0s 18ms/step
<PyPy> Some of the most well-known text mining methods include Information extraction, Clustering, Summarization, Classification, Visualization, Topic discovery and Topic extraction.
<user> what is information extraction?
1/1 [=====] - 0s 17ms/step
<PyPy> The process of information extraction relies on finding entities and categorizing them in order to be easier to manipulate them and use them in databases!
<user> what is clustering?
1/1 [=====] - 0s 18ms/step
<PyPy> Clustering is the process of creating groups with unlabelled texts that are more similar between other members of the same group rather than elements outside the group!
<user> how about summarization
1/1 [=====] - 0s 17ms/step
<PyPy> Summarization is the method of trying to simplify huge amounts of text by extracting only the main parts of the original text! This is a problem that some machine learning algorithms struggle with.
<user> Thank you for your information!
1/1 [=====] - 0s 10ms/step
<PyPy> The pleasure is mine!
<user> Goodbye!
1/1 [=====] - 0s 18ms/step
<PyPy> Have a good day!

```

Figură 17 Model 3 conversatie cu chatbot

Funcțiile secundare ale chatbot-ului:

-Funcția *check\_math* verifică dacă o propoziție ar trebui preluată ca fiind o ecuație matematică. Are ca parametru propozitia dată de utilizator și folosește funcțiile *isupper()* și *islower()* pentru a verifica dacă există litere de orice mărime în text.

```

def check_math(prop):
    if prop.isupper() == 0 or prop.islower() == 0:
        return 1
    else:
        return 0

```

Figură 18 Funcția check\_math

-Funcția `math` se folosește de funcția `check_math`, și rezolvă problema de matematică oferită de utilizator, luând în vedere mai întâi existența parantezelor. Dacă există paranteze, acestea vor fi eliminate, aplicând reguli de bază ale ordinii efectuării operațiilor, că apoi, după eliminarea parantezelor, să se efectueze restul de operații și să se dea un răspuns corect.

```
<user> 1 + 1
1/1 [=====] - 0s 75ms/step
<PyPy> Calculating...
2.0
<user> 2 + 3 - 5 * 4 / 2
1/1 [=====] - 0s 18ms/step
<PyPy> Calculating...
-5.0
<user> ( 4 + 3 ) ^ 2 - ( 2 * 3 - 1 ) ^ 2
1/1 [=====] - 0s 17ms/step
<PyPy> Calculating...
24.0
```

Figură 19 Rezultate ale funcției `math`

-Funcția `bow_bac` este aceeași funcție precum `bow`, însă, aceasta se aplică pe setul de date din cadrul listei numită “`optiuni_bac`” din fișierul json numit “`optiuni`”. Datele din cele două liste din fișier au referințe separate, motiv pentru care, acestea trebuie folosite separat. Funcția creează seturi de pattern-uri de 0 și 1, acestea urmând să fie interpretate de calculator.

```
def bow_bac(prop):
    cuv_prop=curatare(prop)
    bow=[0]*len(bac_cuvinte)
    for cuv in cuv_prop:
        for i,cuvant in enumerate(bac_cuvinte):
            if cuv == cuv:
                bow[i]=1
    return np.array(bow)
```

Figură 20 Funcția `bow_bac`

-Funcția `cea_mai_buna_alegere_bac` este aceeași precum funcția `cea_mai_buna_alege` și este utilizată deoarece aplicația prezintă două modele, și acestea trebuie să poată lucra în același timp,

fară să se intersecteze. O altă diferență între cele două funcții este faptul că aceasta nu prezintă probabilitatea de apariție a claselor, deoarece nu ne interesează această valoare.

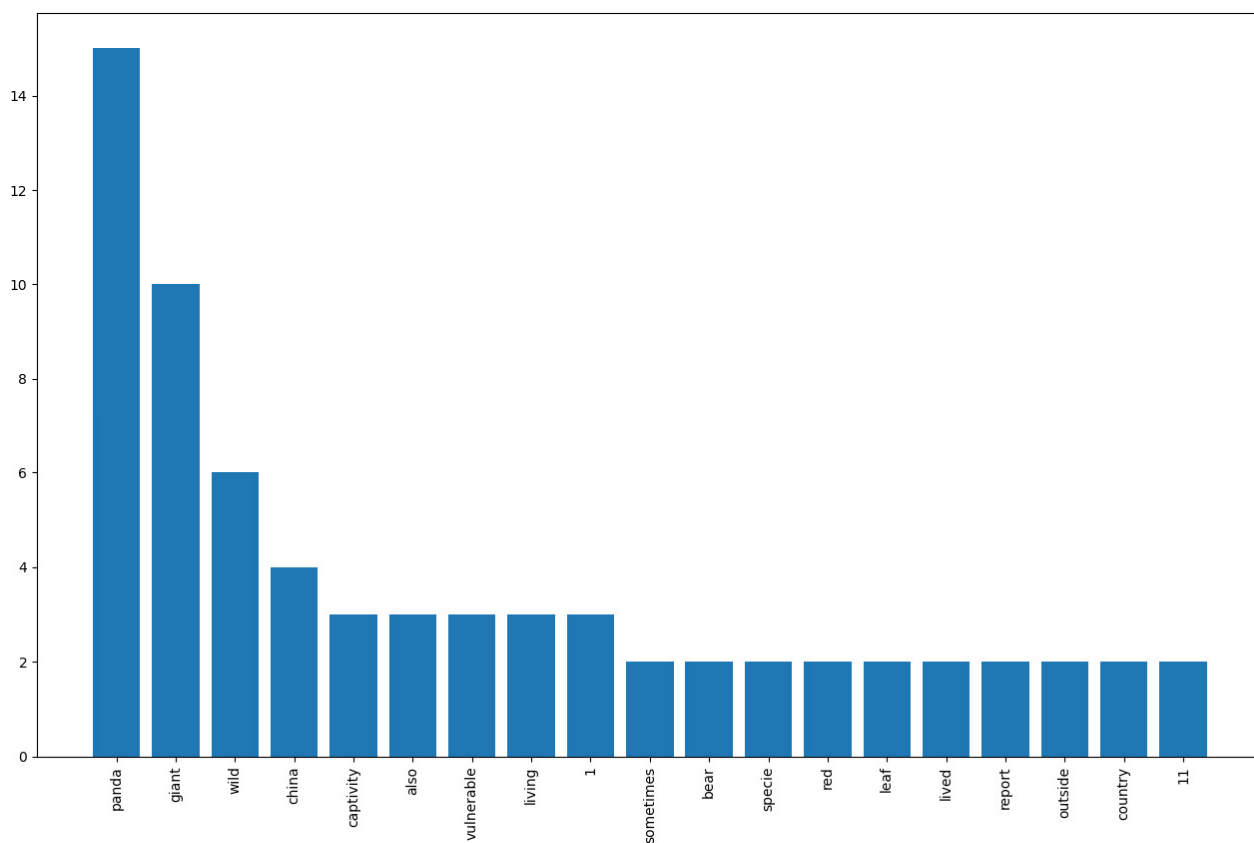
```
def cea_mai_buna_alegere_bac(prop):  
    bow1 = bow_bac(prop)  
    pred = model2.predict(np.array([bow1]))[0]  
    err=0.25  
    rez=[[i,r] for i, r in enumerate(pred) if r>err]  
    rez.sort(key=lambda x:x[1], reverse=True)  
    lista_fin=[]  
    for r in rez:  
        lista_fin.append(bac_categorii[r[0]])  
    return lista_fin
```

Figură 21 Funcția cea\_mai\_buna\_alegere\_bac

-Funcția `cele_mai_dese_cuvinte` realizează un proces de preprocesare în care se elimină punctuațiile, se tokenizează și se lematizează pe valorile după ce s-a aplicat funcția `.lower()` și se elimină cuvintele cu puțină semnificație precum cele din figura 2.1. Următorul proces este eliminarea elementelor duplicate și calcularea frecvențelor fiecărui cuvânt, care apoi se sortează de la cuvântul cu cea mai mare frecvență la cel cu cea mai mică frecvență.

```
def cele_mai_dese_cuvinte(input):  
    a=0  
    cuv=[]  
    cuv1=[]  
    freq=[]  
    freq1 = []  
    input=input.translate(str.maketrans(string.punctuation, ' '*len(string.punctuation)))  
    input=nltk.word_tokenize(input)  
    for i in input:  
        cuv.append(lem.lemmatize(i))  
  
    for i in cuv:  
        if i not in cuv1 and i not in stopwords.words('romanian') and i not in stopwords.words('english'):  
            cuv1.append(i)  
    for i in cuv1:  
        a=0  
        for j in cuv1:  
            if i==j:  
                a=a+1  
        freq.append({'frecventa': a, 'cuvant': i})  
        freq1.append(a)  
  
    freq=sorted(freq, key=lambda freq: (freq['frecventa']), reverse=True)  
    return freq
```

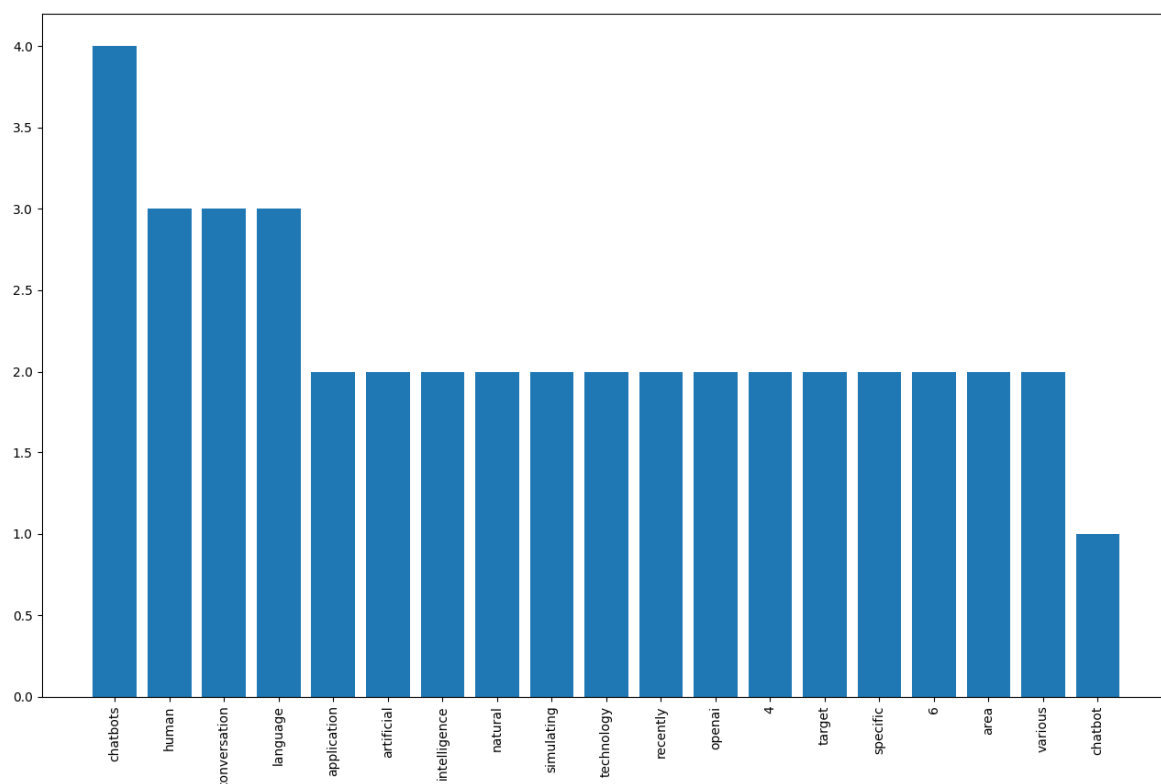
Figură 22 Funcția cele\_mai\_dese\_cuvinte



Figură 23 Frecvențele celor mai folosite cuvinte în documentul despre ursii panda

După inserarea unui articol legat de ursii panda, putem observa histograma cuvintelor cele mai importante și cu ajutorul ei putem trage mai multe concluzii:

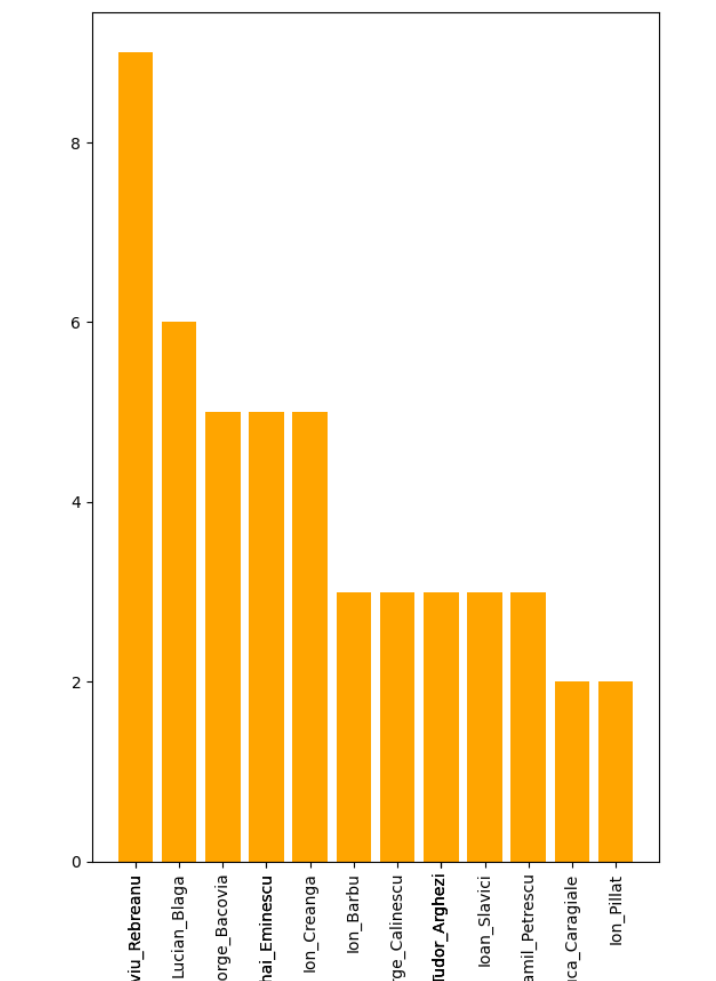
- Ursul panda se mai numește și “Giant Panda”
- Acesta trăiește atât în sălbăcie cât și în captivitate, și este originat din China
- Se cunoaște faptul că este un animal vulnerabil, însă nu pe cale de dispariție.
- Ei sunt o specie de ursi, care mănâncă frunze și care se înrudește cu pandaul roșu.



Figură 24 Frecvențele celor mai folosite cuvinte în documentul despre chatbots

Folosind aceeași logică precum cea de la figura anterioară, din figura de la acest pas putem să concluzionăm niste idei de bază legate de chatboti precum: -Chatbotii sunt algoritmi de inteligență artificială, care sunt utilizați pentru a conversa cu oameni folosind limbaj natural, încercând să simuleze omul. -Ei sunt folosiți în diferite arii de expertiză și sunt capabili de a rezolva probleme foarte specifice. De asemenea, putem observa că unul dintre cei mai cunoscuți chatboti este openAI.





Figură 25 Histograma posibilitatilor de aparitie a autorilor la examenul de bacalaureat

Algoritmul folosește rețele neuronale pentru a antrena setul de date alcătuit din posibilitățile autorilor de a apărea și a operelor lor. Diferite articole și păreri online ale oamenilor despre acest subiect vor avea rolul de a ne face o imagine privind la posibilitatea fiecărui autor în parte, date care mai apoi ne vor ajuta să îl determinăm pe cel cu probabilitatea cea mai mare. În acest caz, autorul care vă pica cel mai probabil este Liviu Rebreanu.

## Concluzii

În concluzie, Text Mining este o știință care are aplicabilitate largă, care conferă posibilitatea aflării de date și analizei lor în vederea soluționării unor probleme în domenii diverse, pornind de la domeniul analizelor statistice, continuând cu domeniul medical, și ajung când chiar și la domeniul militar.

Rezultatul dezvoltării unui chatbot ajută la crearea unei imagini de ansamblu asupra funcțiilor pe care acesta a ajuns să le îndeplinească prin folosirea de diferite tehnici de modelare a datelor, precum information extraction și information retrieval, care au la bază forme ale limbajului natural, folosite pentru a stabili o legătură mai firească între om și mașinărie.

În procesul de creație al chatbot-ului, au fost efectuați următorii pași:

- S-au importat datele de antrenament necesare pentru a crea un model ce poate realiza previziuni.

- Datele au fost preprocesate prin mai multe metode precum: modificarea literelor în caractere mici, tokenizare, eliminarea punctuației și elementelor cu semnificație redusă, lematizare și eliminarea duplicatelor.

- Aplicarea lor în contextul unui model realizat prin intermediul unei metode de Text Mining, precum Information extraction, Classification, Visualization, și alte tipuri de analize.

- În procesele de creare și învățare ale chatbot-ului, am utilizat rețele neuronale, care au dus la înlesnirea comunicării între computer și om, folosind tehnici de prelucrare a limbajului natural.

După realizarea predicției referitoare la subiectul III al examenului de bacalaureat, am ajuns la concluzia că dintr-o listă selectă de autori clasici ai literaturii române, începând cu Liviu Rebreanu și terminând cu Ion Pillat, am realizat o histogramă, conform căreia romancierul Liviu Rebreanu să fie ales de analizat la acest subiect.

## Bibliografie

- Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992). *Knowledge Discovery in Databases: An Overview*.
- Hobbs, J. R. (n.d.). *Chapter 21 Information Extraction*.
- Hotho, A., Nürnberger, A., & Paaß, G. (n.d.). *A Brief Survey of Text Mining*. <http://www.crisp-dm.org/>
- Liddy, E. D. (2001). *Natural Language Processing*. <https://surface.syr.edu/istpub>
- Mahesh, B. (2018). Machine Learning Algorithms-A Review Machine Learning Algorithms-A Review View project Self Flowing Generator View project Batta Mahesh Independent Researcher Machine Learning Algorithms-A Review. *International Journal of Science and Research*. <https://doi.org/10.21275/ART20203995>
- Patel, F. N., & Soni, N. R. (n.d.). Text mining: A Brief survey. In *International Journal of Advanced Computer Research*.
- Text Mining: The state of the art and the challenges*. (2000). <https://www.researchgate.net/publication/2471634>
- Vijay Gaikwad, S. (2014). Text Mining Methods and Techniques. In *International Journal of Computer Applications* (Vol. 85, Issue 17).
- Witten, I. H. (n.d.). *Text mining*.
- Yang, H. C., & Lee, C. H. (2005). A text mining approach for automatic construction of hypertexts. *Expert Systems with Applications*, 29(4), 723–734. <https://doi.org/10.1016/j.eswa.2005.05.003>
- Yang, H.-C., & Lee, C.-H. (2005). A text mining approach for automatic construction of hypertexts. *Expert Systems with Applications*, 29(4), 723–734. <https://doi.org/10.1016/j.eswa.2005.05.003>

## ANEXE

### ANEXA 1. Cod Sursa in Python pentru setul de antrenare al Chatbot-ului

```
documente= []
```

```
lit_ign = ['?', '!', '.', ', ', '']
```

```
for optiune in optiuni["optiuni"]:
```

```
    for intreb in optiune["intrebare"]:
```

```
        lista_cuv = nltk.word_tokenize(intreb.lower())
```

```
        cuvinte.extend(lista_cuv)
```

```
        documente.append((lista_cuv,optiune['subiect']))
```

```
        if optiune['subiect'] not in categori:
```

```
            categori.append(optiune['subiect'])
```

```
cuvinte = [lem.lemmatize(cuvant) for cuvant in cuvinte if cuvant not in lit_ign]
```

```
print(cuvinte)
```

```
cuvinte=sorted(set(cuvinte))
```

```
categori=sorted(set(categori))
```

```
pickle.dump(cuvinte, open('cuvinte.pkl','wb'))  
  
pickle.dump(categori, open('categori.pkl','wb'))
```

```
antrenament = []  
  
out_gol = [0] * len(categori)  
  
for document in documente:  
  
    bow = []  
  
    categ_cuv = document[0]  
  
    categ_cuv=[lem.lemmatize(cuvant.lower()) for cuvant in categ_cuv]  
  
    for cuvant in cuvinte:  
  
        if cuvant in categ_cuv:  
  
            bow.append(1)  
  
        else:  
  
            bow.append(0)  
  
    linii=list(out_gol)  
  
    linii[categori.index(document[1])]=1  
  
    antrenament.append(bow+linii)
```

```

random.shuffle(antrenament)

antrenament=np.array(antrenament)

x_antrenament = antrenament[:, :len(cuvinte)]
y_antrenament = antrenament[:, len(cuvinte):]

model1 = tf.keras.Sequential()

model1.add(tf.keras.layers.Dense(128, input_shape=(len(x_antrenament[0]),),
activation='relu'))

model1.add(tf.keras.layers.Dropout(0.5))

model1.add(tf.keras.layers.Dense(64, activation='relu'))

model1.add(tf.keras.layers.Dropout(0.5))

model1.add(tf.keras.layers.Dense(len(y_antrenament[0]), activation='softmax'))

sgd1 = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True)

#adam1=tf.keras.optimizers.Adam(learning_rate=0.01)

model1.compile(loss='categorical_crossentropy', optimizer=sgd1, metrics=['accuracy'])

a=model1.fit(x_antrenament, y_antrenament, epochs=200, batch_size=5, verbose=1)

model1.save('model1_bun.h5',a)

```

```
model.summary()
```

ANEXA 2. Cod Sursa in Python pentru setul de antrenare al datelor legate de subiectul III in cadrul examenului de bacalaureat

```
import random
```

```
import json
```

```
import pickle
```

```
import numpy as np
```

```
import nltk
```

```
import tensorflow as tf
```

```
from nltk.stem import WordNetLemmatizer
```

```
lem = WordNetLemmatizer()
```

```
optiuni=json.loads(open('optiuni.json').read())
```

```
bac_cuvinte=[]
```

```
bac_categori=[]
```

```
bac_documente=[]
```

```
lit_ign = ['?', '!', '.', ',', '']
```

```
for optiune in optiuni["optiuni_bac"]:
```

```
    for intreb in optiune["intrebare"]:
```

```
        lista_cuv = nltk.word_tokenize(intreb.lower())
```

```
        bac_cuvinte.extend(lista_cuv)
```

```
        bac_documente.append((lista_cuv, optiune['subiect']))
```

```
        if optiune['subiect'] not in bac_categori:
```

```
            bac_categori.append(optiune['subiect'])
```

```
bac_cuvinte=sorted(set(bac_cuvinte))
```

```
bac_categori=sorted(set(bac_categori))
```

```
pickle.dump(bac_cuvinte, open('bac_cuvinte.pkl', 'wb'))
```

```
pickle.dump(bac_categori, open('bac_categori.pkl', 'wb'))
```

```
antrenament = []
```

```
out_gol = [0] * len(bac_categori)
```

```
for document in bac_documente:
```

```
    bow = []
```

```
    categ_cuv = document[0]
```



```

categ_cuv =[lem.lemmatize(cuvant.lower()) for cuvant in categ_cuv]

for cuvant in bac_cuvinte:

    if cuvant in categ_cuv:

        bow.append(1)

    else:

        bow.append(0)

    linii=list(out_gol)

    linii[bac_categori.index(document[1])]=1

    antrenament.append(bow+linii)

random.shuffle(antrenament)

antrenament=np.array(antrenament)

x_antrenament = antrenament[:, :len(bac_cuvinte)]

y_antrenament = antrenament[:, len(bac_cuvinte):]

model2 = tf.keras.Sequential()

model2.add(tf.keras.layers.Dense(128,                    input_shape=(len(x_antrenament[0]),),
activation='relu'))

```

```

model2.add(tf.keras.layers.Dropout(0.5))

model2.add(tf.keras.layers.Dense(64, activation='relu'))

model2.add(tf.keras.layers.Dropout(0.5))

model2.add(tf.keras.layers.Dense(len(y_antrenament[0]), activation='softmax'))


sgd1 = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True)


model2.compile(loss='categorical_crossentropy', optimizer=sgd1, metrics=['accuracy'])


b=model2.fit(x_antrenament, y_antrenament, epochs=200, batch_size=5, verbose=1)

model2.save('model2_bun.h5',b)

model2.summary()

```

ANEXA 3. Cod Sursa in Python pentru crearea chatbot-ului si apelarea sa

```

import random

import json

import pickle

import re

import string

import matplotlib.pyplot as plt

```

```
from matplotlib import pyplot as plt

import numpy as np

import nltk

import tensorflow as tf

from nltk.stem import WordNetLemmatizer

from nltk.corpus import stopwords


import pandas as pd

from tensorflow.python.keras.models import load_model

lem = WordNetLemmatizer()


optiuni=json.loads(open('optiuni.json').read())


cuvinte = pickle.load(open('cuvinte.pkl','rb'))

categori = pickle.load(open('categori.pkl','rb'))

bac_cuvinte = pickle.load(open('bac_cuvinte.pkl','rb'))

bac_categori = pickle.load(open('bac_categori.pkl','rb'))


model1=tf.keras.models.load_model('model1_bun.h5')

model2=tf.keras.models.load_model('model2_bun.h5')
```

```

def cele_mai_dese_cuvinte(input):

    a=0

    cuv=[]

    cuv1=[]

    freq=[]

    freq1 = []

    input=input.translate(str.maketrans(string.punctuation, '*len(string.punctuation)))

    input=nltk.word_tokenize(input)

    for i in input:

        cuv.append(lem.lemmatize(i))


    for i in cuv:

        if i not in cuv1 and i not in stopwords.words('romanian') and i not in
stopwords.words('english'):

            cuv1.append(i)

    for i in cuv1:

        a=0

        for j in cuv:

            if i==j:

                a=a+1

```

```

    freq.append({'frecventa': a, 'cuvant': i})

    freq1.append(a)

    freq=sorted(freq, key = lambda freq: (freq['frecventa']),reverse=True)

    return freq

def check_math(prop):

    if prop.isupper()==0 or prop.islower()==0:

        return 1

    else:

        return 0

def math(x:list[str]):

    v: list[str] = []

    d:int=0

    y:str

    a:int =0

    b:int =0

    while "(" in x:

        i = 0

        for j in range(0, len(x)):

            if x[j] == ")":

```

```
b = j
```

```
break
```

```
for j in range(b, -1, -1):
```

```
    if x[j] == "(":
```

```
        a = j;
```

```
        break
```

```
for j in range(a+1, b):
```

```
    v.append(x[j])
```

```
    i = i + 1
```

```
y=math(v)
```

```
d=b-a
```

```
while d>0:
```

```
    x.pop(a)
```

```
    d=d-1
```

```
x[a]=y
```

```
break
```

```
if "(" in x:
```

```
    math(x)
```

```
while "^" in x:
```

```
    for i in range(0, len(x)):
```

```

if x[i] == "^":

    r = pow(float(x[i - 1]), float(x[i + 1]))

    x[i - 1] = str(r)

    x.pop(i)

    x.pop(i)

    break

while "*" in x or "/" in x:

    for i in range(0, len(x)):

        if x[i] == "*" or x[i] == "/":

            if x[i] == "*":

                r = float(x[i - 1]) * float(x[i + 1])

                x[i - 1] = str(r)

                x.pop(i)

                x.pop(i)

                break

            if x[i] == "/":

                r = float(x[i - 1]) / float(x[i + 1])

                x[i - 1] = str(r)

                x.pop(i)

                x.pop(i)

                break

```

```

while "+" in x or "-" in x:

    for i in range(len(x)):

        if x[i] == "+" or x[i] == "-":

            if x[i] == "+":

                r = float(x[i - 1]) + float(x[i + 1])

                x[i - 1] = str(r)

                x.pop(i)

                x.pop(i)

                break

            if x[i] == "-":

                r = float(x[i - 1]) - float(x[i + 1])

                x[i - 1] = str(r)

                x.pop(i)

                x.pop(i)

                break

    return x[len(x)-1]

```

```

def curatare(prop):

    cuv_prop = nltk.word_tokenize(prop.lower())

```



```

cuv_prop = [lem.lemmatize(cuvant) for cuvant in cuv_prop]

return cuv_prop

```

```

def bow(prop):

    cuv_prop=curatare(prop)

    bow=[0]*len(cuvinte)

    for cuv in cuv_prop:

        for i,cuvant in enumerate(cuvinte):

            if cuvant == cuv:

                bow[i]=1

    return np.array(bow)

```

```

def bow_bac(prop):

    cuv_prop=curatare(prop)

    bow=[0]*len(bac_cuvinte)

    for cuv in cuv_prop:

        for i,cuvant in enumerate(bac_cuvinte):

            if cuvant == cuv:

                bow[i]=1

    return np.array(bow)

```

```

def cea_mai_buna_alegere(prop):

    bow1 = bow(prop)

    pred = model1.predict(np.array([bow1]))[0]

    err=0.1

    rez=[[i,r]for i, r in enumerate(pred) if r>err]

    rez.sort(key=lambda x:x[1],reverse=True)

    lista_fin=[]

    for r in rez:

        lista_fin.append({'optiunea': categori[r[0]], 'Probabilitate': str(r[1])})

    return lista_fin

def cea_mai_buna_alegere_bac(prop):

    bow1 = bow_bac(prop)

    pred = model2.predict(np.array([bow1]))[0]

    err=0.3

    rez=[[i,r]for i, r in enumerate(pred) if r>err]

    rez.sort(key=lambda x:x[1],reverse=True)

    lista_fin=[]

    for r in rez:

        lista_fin.append(bac_categori[r[0]])

    return lista_fin

```

```

def raspuns(lista_optiuni,json_optiuni,input,art,doc):

    yy=[]

    y_bac1=[]

    categorie = lista_optiuni[0]['optiunea']

    opt_lista=json_optiuni['optiuni']

    if categorie == 'article':

        k = cele_mai_dese_cuvinte(art)

        print(k)

        x = []

        y = []

        aa=[]

        bb=[]

        y_bac=[]

        g = 1

        for i in k:

            x.append(i['frecventa'])

            g = g + 1

            if g == 20:

                break

        g = 1

        for i in k:

```

```

        y.append(i['cuvant'])

        g = g + 1

        if g == 20:

            break

    for i in k:

        if cea_mai_buna_alegere_bac(i['cuvant']):

y_bac.append({'cuvant':cea_mai_buna_alegere_bac(i['cuvant'])[0],'frecventa':i['frecventa']})

    """for i in y:

        if cea_mai_buna_alegere_bac(i):

            y_bac.append(cea_mai_buna_alegere_bac(i)[0])

    """

    print("y_bac",y_bac)


    lst_bac=json_optiuni['optiuni_bac']

    print("lst_bac",lst_bac)

    for ya in y_bac:

        for i in lst_bac:

            if i['subiect']==ya['cuvant']:

                yy.append({"nume":i['raspunsuri'][0],"frecventa":ya['frecventa']})

```

```

g=1

for i in yy:

    aa.append(i['frecventa'])

    g = g + 1

    if g == 17:

        break

```

```

g = 1

for i in yy:

    bb.append(i['nume'])

    g = g + 1

    if g == 17:

        break

```

```

plt.bar(bb,aa,color='orange')

plt.style.use('fivethirtyeight')

plt.xticks(bb, rotation=90)

plt.show()

```

```

if categorie == 'frequency':

    f = cele_mai_dese_cuvinte(doc)

    x1 = []

```

```

y1 = []

g = 1

for i in f:

    x1.append(i['frecventa'])

    g = g + 1

    if g == 20:

        break

g = 1

for i in f:

    y1.append(i['cuvant'])

    g = g + 1

    if g == 20:

        break

plt.bar(y1, x1)

plt.style.use('fivethirtyeight')

plt.xticks(y1, rotation=90)

plt.show()

for i in opt_lista:

    if i['subiect']==categorie:

        rasp=random.choice(i['raspunsuri'])

```

```
return rasp
```

```
b=1
```

```
while b==1:
```

```
    u_input= input("<user> ")
```

```
    u_input= u_input.lower()
```

```
    with open('article.txt', 'r') as file:
```

```
        art = file.read().replace("\n", "")
```

```
    with open('docfreq.txt', 'r') as file:
```

```
        doc = file.read().replace("\n", "")
```

```
    art=art.lower()
```

```
    doc=doc.lower()
```

```
    a=cea_mai_buna_alegere(u_input)
```

```
    print(f"<PyPy> {raspuns(a,optiuni,u_input,art,doc)}")
```

```
    if check_math(u_input):
```

```
        print(math(u_input.split()))
```

```
    if a[0]['optiunea']=='conversation_end':
```

```
        b=0
```

ANEXA 4. Seturi de antrenare chatbot

```

{
  "optiuni": [
    {
      "subiect": "conversation_start",
      "intrebare": ["hello!", "salutations!", "hi", "hey", "what's up"],
      "raspunsuri": ["Hello, User!", "Good day!", "Salutations!"]
    },
    { "subiect": "name",
      "intrebare": ["what is your name?", "who are you?", "what do they call you?", "how are
you named"],
      "raspunsuri": ["My name is PyPy!", "I am PyPy!"]
    },
    { "subiect": "occupation",
      "intrebare": ["what can you do", "what do you do", "what is your purpose"],
      "raspunsuri": ["I can perform several tasks!\nI can perform basic mathematical
calculations!\nI can have chats with users!\nI can predict different outcomes using given articles!\nI
can show the most frequent words in any article!"]
    },
    { "subiect": "today",
      "intrebare": ["how is your day?", "how are you feeling today?", "how are you?", "how do
you feel?"]
    }
  ]
}

```



"raspunsuri": ["I feel great! How about you? ", "I feel awesome! You? ", "I feel amazing! And you?"]

},

{"subject": "good",

"intrebare": ["i also feel good", "i feel alright", "im good", "im ok", "im fine"],

"raspunsuri": ["I am happy to hear that!", "That's great!", "I'm so glad you are ok!"]

},

{"subject": "bad",

"intrebare": ["i dont feel so good", "i feel bad", "i dont feel good", "im not alright", "i feel sad"],

"raspunsuri": ["So sorry to hear that...", "I hope you get well soon!", "You are going to be alright very soon, I am sure!"]

},

{"subject": "thanks",

"intrebare": ["thanks!", "thank you", "i thank you", "i appreciate it", "thank you for saying that"],

"raspunsuri": ["I'm glad to help!", "The pleasure is mine!", "I help any time!", "I'm so happy I got to help you!"]

},

{"subject": "compliment",

"intrebare": ["you are the best", "you are amazing", "you helped me a lot", "you helped", "you are helpful"],

"raspunsuri": ["Oh, stop it...", "You are making me blush...", "I feel humbled!"]

```

    },

    { "subject": "text_mining",

      "intrebare": ["what is text mining?", "tell me something about text mining", "what can you
tell me on text mining", "what's text mining"],

      "raspunsuri": ["Text mining is the science that is involved in transforming unstructured
text data and turning it into new useful data"]

    },

    { "subject": "conversation_end",

      "intrebare": ["goodbye", "see you later", "bye", "bye bye", "quit"],

      "raspunsuri": ["I hope to see you soon!", "Have a good day!", "I'm glad we got to
speak!", "Thanks for talking to me!"]

    }

  ,

    { "subject": "math",

      "intrebare": ["1 + 1", "1 * 1", "1 / 1", "1 - 1", "1 ^ 1"],

      "raspunsuri": ["Calculating..."]

    }

  ,

    { "subject": "article",

      "intrebare": ["Check article", "Mine article", "Use the article"],

      "raspunsuri": ["Article Analyzed Successfully!"]

    }

  }

```

```
,

{ "subject": "frequency",

  "intrebare": ["check frequency","frequency","search document frequency"],

  "raspunsuri": ["Frequencies identified successfully!"]

}

,

{ "subject": "temp",

  "intrebare": ["is it cold?","is it hot?","How is the weather like?"],

  "raspunsuri": ["The weather is fine!","The weather is pleasant!","The weather is not that
great!","It's fine for a walk!"]

},

{ "subject": "data_mining",

  "intrebare": ["What is data mining?", "Tell me something about data mining","What can
you tell me on data mining","what's data mining"],

  "raspunsuri": ["Data Mining is the process of analysing huge data and extracting useful
information from it!"]

},

{ "subject": "natural_language_processing",

  "intrebare": ["what is natural language processing", "tell me something about natural
language processing","what can you tell me on natural language processing","what's natural
language processing"],

  "raspunsuri": ["Natural language processing is the science that helps make the connection
between human language and computer understanding!"]

}
```

```

    },

    {

        "subject": "tm_methods",

        "intrebare": ["tell me some text mining methods", "do you know text mining
methods","what are the text mining methods you know?","i want to know some text mining
methods"],

        "raspunsuri": ["Some of the most well-known text mining methods include Information
extraction, Clustering, Summarization, Classification, Visualization, Topic discovery and Topic
extraction."]

    },

    {

        "subject": "preprocessing",

        "intrebare": ["tell me some preprocessing steps", "do you know preprocessing
steps","what are the preprocessing steps you know?","i want to know some preprocessing steps"],

        "raspunsuri": ["The more frequently used preprocessing steps in text mining are
tokenization, bag of words, part of speech tagging, stemming, lemmatization, stopword elimination
and making string into lower case!"]

    },

    {

        "subject": "tokenization",

        "intrebare": ["what is tokenization", "what do you know about tokenization","how do you
define tokenization","i want to know what tokenization is"],

        "raspunsuri": ["Tokenization is the process of turning string sentences into a list of
words!"]

```

```

    },

    {

        "subject": "stop_words",

        "intrebare": ["what are stop words", "what do you know about stop words", "how do you
define stop words", "i want to know what stop words are"],

        "raspunsuri": ["Stop words are a list of words from different languages that bare minimum
significance to a sentence's meaning and are usually eliminated!"]

    },

    {

        "subject": "stemming",

        "intrebare": ["what is stemming", "what do you know about stemming", "how do you
define stemming", "i want to know what stemming is"],

        "raspunsuri": ["The process of stemming usually means cutting a word down to it's stem,
trying to find the origin word!"]

    },

    {

        "subject": "lemmatizing",

        "intrebare": ["what is lemmatizing", "what do you know about lemmatizing", "how do you
define lemmatizing", "i want to know what lemmatizing is"],

        "raspunsuri": ["The process of lemmatizing is a more complex process then stemming,
that can accurately obtain the origin of a word without losing it's meaning!"]

    },

    {

```

"subject": "information\_extraction",

"intrebare": ["what is information extraction", "what do you know about information extraction", "how do you define information extraction", "i want to know what information extraction is"],

"raspunsuri": ["The process of information extraction relies on finding entities and categorizing them in order to be easier to manipulate them and use them in databases!"]

},

{

"subject": "information\_retrieval",

"intrebare": ["what is information retrieval", "what do you know about information retrieval", "how do you define information retrieval", "i want to know what information retrieval is"],

"raspunsuri": ["The process of information retrieval is that of gaining the documents and metadata that answer questions, rather than actually looking for the answers themselves!"]

},

{

"subject": "clustering",

"intrebare": ["what is clustering", "what do you know about clustering", "how do you define clustering", "i want to know what clustering is"],

"raspunsuri": ["Clustering is the process of creating groups with unlabelled texts that are more similar between other members of the same group rather than elements outside the group!"]

},

{

"subject": "summarization",

"intrebare": ["what is summarization", "what do you know about summarization","how do you define summarization","i want to know what summarization is"],

"raspunsuri": ["Summarization is the method of trying to simplify huge amounts of text by extracting only the main parts of the original text! This is a problem that some machine learning algorithms struggle with."]

},

{

"subject": "classification",

"intrebare": ["what is classification", "what do you know about classification","how do you define classification","i want to know what classification is"],

"raspunsuri": ["Classification is the process that puts together NLP and machine learning to rearrange documents into groups so that they are easier to work with later!"]

},

{

"subject": "visualization",

"intrebare": ["what is visualization", "what do you know about visualization","how do you define visualization","i want to know what visualization is"],

"raspunsuri": ["Visualization is the way of interpreting numerical values using visual representations such as graphs from mining text!"]

},

{

"subject": "topic\_discovery",

"intrebare": ["what is topic discovery", "what do you know about topic discovery", "how do you define topic discovery", "i want to know what topic discovery is"],

"raspunsuri": ["Topic discovery is a method that tries to find the best words to describe the topic of a given document!"]

},

{

"subject": "topic\_extraction",

"intrebare": ["what is topic extraction", "what do you know about topic extraction", "how do you define topic extraction", "i want to know what topic extraction is"],

"raspunsuri": ["Topic extraction uses machine learning to understand and the put together big numbers of text data collections by giving labels or classes to every text's topic"]

},

{

"subject": "parsing",

"intrebare": ["what is parsing", "what do you know about parsing", "how do you define parsing", "i want to know what parsing is"],

"raspunsuri": ["Parsing is a method that generated a tree-like model that uses a divide and conquer strategy to obtain semantic information about the text such as relationships between words"]

},

{

"subject": "chunking",

"intrebare": ["what is chunking", "what do you know about chunking", "how do you define chunking", "i want to know what chunking is"],



"raspunsuri": ["Chunking is the process of obtaining the parts of speech of words in a given document!"]

},

{

"subiect": "term\_disambiguation",

"intrebare": ["what is term disambiguation", "what do you know about term disambiguation", "how do you define term disambiguation", "i want to know what term disambiguation"],

"raspunsuri": ["Term disambiguation is a process that extracts the meaning of a word in a given context and has a crucial role in the recognition of terms"]

}

],

"optiuni\_bac": [

{

"subiect": "eminescu",

"intrebare": ["eminescu", "luceafarul", "floare"],

"raspunsuri": ["Mihai\_Eminescu"]

},

{

"subiect": "creanga",

```
"intrebare": ["creanga", "harap"],

"raspunsuri": ["Ion_Creanga"]

},

{

"subiect": "caragiale",

"intrebare": ["caragiale", "scrisoare"],

"raspunsuri": ["Ion_Luca_Caragiale"]

},

{

"subiect": "slavici",

"intrebare": ["slavici", "moara"],

"raspunsuri": ["Ioan_Slavici"]

},

{

"subiect": "bacovia",

"intrebare": ["bacovia", "plumb", "lacustra"],

"raspunsuri": ["George_Bacovia"]

},

{

"subiect": "eliade",

"intrebare": ["eliade", "maitreyi"],
```

```
"raspunsuri": ["Mircea_Eliade"]

},

{

"subiect": "blaga",

"intrebare": [" blaga ", "corola"],

"raspunsuri": ["Lucian_Blaga"]

},

{

"subiect": "arghezi",

"intrebare": ["arghezi", "testament","mucigai"],

"raspunsuri": ["Tudor_Arghezi"]

},

{

"subiect": "barbu",

"intrebare": ["barbu ", "ryga"," barbu ","barbu "," barbu"," barbu "],

"raspunsuri": ["Ion_Barbu"]

},

{

"subiect": "sadoveanu",

"intrebare": ["sadoveanu", "ancutei","baltagul"],

"raspunsuri": ["Mihail_Sadoveanu"]
```

```
},  
  
{  
  
  "subiect": "calinescu",  
  
  "intrebare": ["calinescu", "enigma","otilia"],  
  
  "raspunsuri": ["George_Calinescu"]  
  
},  
  
{  
  
  "subiect": "petrescu",  
  
  "intrebare": ["petrescu", "ultima"],  
  
  "raspunsuri": ["Camil_Petrescu"]  
  
},  
  
{  
  
  "subiect": "pillat",  
  
  "intrebare": ["pillat", "vremuri"],  
  
  "raspunsuri": ["Ion_Pillat"]  
  
},  
  
{  
  
  "subiect": "stanescu",  
  
  "intrebare": ["stanescu", "leaica","dulcele"],  
  
  "raspunsuri": ["Nichita_Stanescu"]  
  
},
```

```
{  
  
  "subiect": "sorescu",  
  
  "intrebare": ["sorescu", "iona"],  
  
  "raspunsuri": ["Marin_Sorescu"]  
  
},  
  
{  
  
  "subiect": "rebreanu",  
  
  "intrebare": ["rebreanu", "ion", "padurea"],  
  
  "raspunsuri": ["Liviu_Rebreanu"]  
  
},  
  
{  
  
  "subiect": "nni",  
  
  "intrebare": ["abcdefghijklmnopqrstuvwxyz"],  
  
  "raspunsuri": ["nu"]  
  
}  
  
]  
  
}
```