UNIVERSITATEA DIN BUCUREŞTI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INTELIGENȚĂ ARTIFICIALĂ

Lucrare de disertație
Analiza Sentimentelor Bazată pe Aspecte

Coordonator științific **Masterand**

Conf. dr. Popescu Marius **Giuşcă Iulian**

București, iunie 2019

Abstract

În lucrarea de față sunt descrise abordări de rezolvare pentru analiza sentimentelor bazată pe aspecte. Tema se descompune în trei subprobleme: identificarea aspectelor dintr-o recenzie, identificarea expresiilor care fac referire la un aspect, și determinarea polarității aspectelor. Pentru identificarea aspectelor am folosit clasificatorul SVM ca punct de plecare, îmbunătățind rezultatul prin folosirea rețelelor neuronale de convoluție; pentru a doua subproblemă am folosit metoda Câmpurilor Aleatorii Condiționate pentru marcarea secvențelor de cuvinte cu sistemul IOB, iar pentru determinarea polarității am implementat o metodă de regresie liniară multinomială. Am implementat și o metodă de fragmentare a unei recenzii care identifică un context al unui aspect și îl separă de restul recenziei, apelând tehnicile de analiză a sentimentelor pe aceste fragmente.

Cuprins

Introducere....1
Motivarea temei....1
Alte lucrări în domeniu și descrierea problemei....2
Conținutul lucrării....3
Capitolul I - Identificarea aspectelor....5
1.1. Descrierea seturilor de date....5
1.2. Metoda 1 – Mașini Vector Suport....6
1.2.1. Aspecte teoretice....6
1.2.2. Obținerea caracteristicilor....11
1.2.3. Selectarea modelului....13
1.3. Metoda 2 – Rețele Neuronale de Convoluție....15
1.3.1. Vectorii de caracteristici....18
1.3.2. Stratul de convoluție....22
1.3.3. Stratul de agregare....25
1.3.4. Stratul softmax....25
1.3.5. Antrenarea rețelei....26
1.4. Rezultate....29
Capitolul II – Identificarea Expresiilor Țintă....32
2.1. Descrierea setului de date....32
2.2. Problema Etichetării Secvențiale....33
2.2.1. Caracteristicile folosite....33
2.2.2. Câmpuri aleatorii condiționate....35
2.3. Rezultate....38
Capitolul III – Determinarea polarității....41
3.1. Descrierea datelor....41
3.2. Separarea aspectelor....42
3.3. Metoda 1 – Regresie Logistică Multinomială....45
3.4. Rezultate....47
Concluzii....49
Anexe....51
Anexa 1: Tabel cu valorile hiperparametrilor pentru modelul SVC....51
Referințe bibliografice....53

Introducere

Motivarea temei

Procesul de luare a unei decizii este puternic influențat de ceea ce cred alți oameni despre obiectul de interes. Odată cu evoluția tehnologiei, a evoluat și modul în care oamenii caută informații – în mediul on-line. Date statistice prezentate în studiul lui Horrigan [19] confirmă acest fapt: 81% dintre utilizatori au căutat recenzia unui produs pe internet cel puțin o dată; în peste 73% din cazuri, recenziile restaurantelor sau hotelurilor au influențat alegerea făcută; utilizatorii tind să plătească mai mult pe un produs cu un rating de 5 stele decât pe unul de 4 stele ș.a.m.d.

Totuși, conform studiului, 58% dintre utilizatori au întâmpinat dificultăți în căutarea informațiilor dorite: informația lipsește, este greu de găsit, sau există o cantitate foarte mare de informații. Prin urmare, a crescut interesul pentru sistemele în care opiniile sunt tratate în mod automat. Interesul este cu atât mai crescut, cu cât și companiile au înțeles că succesul propriu depinde de percepția publicului despre bunurile și serviciile oferite. Un astfel de sistem poartă numele de analiza sentimentelor.

Analiza sentimentelor reprezintă o aplicație a procesării limbajului natural, ce are ca scop identificarea opiniilor pozitive, negative, sau a emoțiilor dintr-un text [59]. Sentimentul poate fi cuantificat cu ajutorul unor valori: pozitiv, negativ sau neutru, ce poartă numele de polaritatea textului. Polaritatea poate fi exprimată în mai multe moduri: fie la nivelul întregului text, ori la nivel de propoziție, prin intermediul unui scor ce caracterizează întregul text analizat. Așa cum este prezentat în lucrări precum Pang et al. [40], Turney [57], se poate determina dacă o recenzie a unui produs exprimă, per total, un sentiment pozitiv, negativ, sau unul neutru. Pentru o astfel de aplicație, sunt valoroase informațiile subiective, adică texte în care se exprimă emoții, trăiri, stări de spirit, (ex: *I had a great time seeing this movie*). Textul obiectiv este lipsit de astfel de detalii, oferindu-se informații care nu ajută la stabilirea polarității (ex: *The film started at 3 p.m.*).

Problema devine mai dificilă când avem de determinat sentimentul în texte unde sunt exprimate mai multe opinii, adesea de polarități diferite, ca în următorul exemplu: *I liked the atmosphere of the place, but the food was not great.* Sau, există situații când un utilizator este

interesat doar de un singur aspect în legătură cu un produs, cum ar fi durata de viață a bateriei unui laptop, și trebuie să treacă prin foarte multe recenzii până găsește informația dorită.

O aplicație propusă pentru identificarea tuturor opiniilor și aspectelor ce fac legătură cu un obiect poartă numele de analiza sentimentelor bazată pe aspecte.

Alte lucrări în domeniu și descrierea problemei

Analiza sentimentelor bazată pe aspecte (*Aspect Based Sentiment Analysis; ABSA*) a devenit un instrument important pentru sumarizarea opiniilor provenite din recenziile on-line [43]. În ultimii ani, au fost elaborate astfel de sisteme pentru diverse tipuri de recenzii: de film (Thet et al. [55]), de produse electronice (Hu și Liu [21]), de restaurante (Ganu et al. [10]) etc. În toate lucrările amintite anterior, obiectivul este reprezentat de identificarea aspectelor dintr-o recenzie, urmată de determinarea polarității pentru fiecare aspect în parte.

Analiza sentimentelor bazată pe aspecte se descompune, așadar, în subprobleme ce pot fi rezolvate în mod independent. O astfel de definire a subproblemelor este introdusă în lucrearea lui Popescu și Etzioni [46] în anul 2005. Ei au împărțit analiza sentimentelor bazată pe aspecte astfel: mai întâi efectuează identificarea aspectelor dintr-o recenzie, apoi identifică toate opiniile asociate aspectelor respective, urmată de determinarea polarității opiniilor. Ca ultim pas, ei efectuează și o clasificare, sau o sortare a opiniilor bazată pe cât de puternic este exprimat sentimentul în recenzie.

O definire mai recentă a subproblemelor este introdusă la The International Workshop of Semantic Evaluation SemEval. Din 2014 și până în 2016, analiza sentimentelor bazată pe aspecte a fost o temă abordată la SemEval [43][44][45], la care numeroase echipe au participat și au oferit soluții (29 echipe și 245 de soluții trimise în anul 2016). La această convenție, tema a fost împărțită în mai multe categorii: ABSA la nivel de propoziție sau ABSA la nivel de text. Ei au împărțit subproblemele astfel: identificarea aspectelor din text (*Aspect Category Detection; ACD*), identificarea cuvintelor sau a expresiilor despre care se exprimă opinii (*Opinion Target Expression, OTE*), și determinarea polarității pentru astfel de expresii identificate (*Sentiment Polarity*). Pentru fiecare subproblemă se oferă seturi de date corespunzător adnotate.

Aspectele reprezintă entități (cuvinte, expresii, concepte, categorii ș.a.m.d.) ce exprimă caracteristici mai specifice despre obiectul de interes. De exemplu, pentru un restaurant, putem

avea recenzii legate de mâncare, băutură, servire, și altele, acestea reprezentând aspecte. Cel mai adesea, aspectele vin sub forma unei liste predefinite de entități. De exemplu, în lucrarea lui Ganu et al. [10], pentru restaurante, lista este alcătuită din 6 aspecte: FOOD, AMBIENCE, PRICE, SERVICE, ANCEDOTES și MISCELLANEOUS. Recenzii, sau chiar și propoziții, pot avea asociate mai multe aspecte. În propoziția menționată anterior, *I liked the atmosphere of the place, but the food was not great,* aspectele asociate sunt AMBIENCE, pentru care se exprimă o părere pozitivă, și FOOD, părere negativă. Aspectele se pot identifica cu ajutorul unor expresii lingvistice denumite cuvinte țintă. De exemplu, în propoziția anterioară, aspectele AMBIENCE, respectiv FOOD sunt identificate prin cuvintele țintă *atmosphere*, respectiv *food*.

Conținutul lucrării

Lucrarea de față este împărțită în trei capitole, fiecare dedicat rezolvării uneia din cele trei subprobleme menționate anterior. La finalul fiecărui capitol sunt incluse rezultatele obținute în aplicație, ce sunt comparate cu rezultatele participanților de la convenția SemEval 2016.

În primul capitol sunt prezentate două metode pentru identificarea aspectelor: o metodă folosind mașinile vector suport și una folosind rețele neuronale de convoluție. În cazul metodei mașinilor vector suport, se prezintă fundamentele matematice iar accentul se pune pe determinarea valorilor optime pentru parametrii modelului printr-o metodă de căutare grilă, și calcularea vectorului de caracteristici prin formarea unui lexicon de termeni și calcularea celor trei măsuri: precizie, rata de regăsire și măsura F1. Pentru cea de-a doua metodă, se prezintă arhitectura rețelei neuronale de convoluție, modul de selecție al parametrilor și se descriu vectorii de încorporare GloVe folosiți pentru a obține setul de caracteristici.

În cel de-al doilea capitol este prezentată metoda câmpurilor aleatorii condiționate (CRF), ce rezolvă problema identificării expresiilor lingvistice țintă care identifică un aspect. Este introdus sistemul de etichetare IOB cu care vor fi etichetate cuvintele dintr-o recenzie și este descrisă problema CRF la nivel formal.

În cel de-al treilea capitol este prezentată metoda regresiei logistice multivariate pentru determinarea polarității aspectelor dintr-o recenzie. Se pornește de la premisa că, în acel punct al aplicației, se cunosc aspectele și respectiv expresiile țintă corespunzătoare prin rezolvarea subproblemelor 1 și 2 – se poate lucra direct cu datele de ieșire provenite din cele două

subprobleme, sau informațiile se pot prelua direct din setul de date adnotat, pentru a rezolva subproblema în mod independent. Am descris la nivel formal regresia logistică multivariată și am propus o metodă care descompune o propoziție în secvențe mai scurte de cuvinte, pentru a izola contextul în care se vorbește despre un aspect. Acest lucru va asigura faptul că o singură propoziție descrie un singur aspect, lucru ce va facilita determinarea polarității.

În final, vor fi prezentate concluzii, observații adiționale și eventuale direcții de urmat în viitor pentru îmbunătățirea rezultatelor.

Capitolul I - Identificarea aspectelor

În acest capitol, vom aborda identificarea aspectelor din recenzii folosind două metode supervizate de clasificare: prin intermediul mașinilor vector suport, și folosind rețele neuronale de convoluție.

1.1. Descrierea seturilor de date

Am folosit seturile de date oferite de către SemEval în cadrul convenției organizate în anul 2016 [45], și anume recenziile despre laptop-uri și recenziile despre restaurante. Există 450 de recenzii pentru antrenare și 80 folosite pentru testare în cazul laptop-urilor, respectiv 350 de recenzii pentru antrenare și 90 pentru testare în cazul restaurantelor (Tabelul 1.1).

În ambele seturi de date, o recenzie este alcătuită din mai multe propoziții în care se exprimă opinii clare despre diferite caracteristici ale obiectului sau locației descrise. O recenzie exprimă opinii despre unul sau mai multe aspecte, iar polaritățile pot să difere de la un aspect la altul, ca în exemplul: „*The food here is rather good, but only if you like to wait for it.*”, unde avem o părere pozitivă despre mâncare și una negativă despre serviciu.

		Recenzii	Propoziții	n=0	n=1	n=2	n>2
Laptop	Antrenare	450	2500	461	1352	534	153
	Testare	80	808	235	394	143	36
Restaurant	Antrenare	350	2000	292	1151	381	176
	Testare	90	676	89	410	128	49

Tabelul 1.1: Date statistice despre cele două seturi de date: numărul de recenzii, numărul de propoziții, și numărul de propoziții adnotate cu n aspecte

După cum se poate observa și în Tabelul 1.1, cele mai multe propoziții sunt adnotate cu un singur aspect (coloana marcată cu $n = 1$), însă avem un număr considerabil de propoziții în care există mai multe aspecte. Există și propoziții în care nu se exprimă nici o opinie, așa cum este firesc, rolul acestora fiind acela de a întregi contextul într-o recenzie.

Un aspect adnotat în seturile de date este alcătuit dintr-o pereche E#A formată dintr-o entitate E și un atribut A, ca de exemplu: RESTAURANT#GENERAL, FOOD#QUALITY ș.a.m.d. Atât entitățile, cât și atributele provin dintr-o listă predefinită de termeni. Așa cum e prezentat și în Tabelul 1.2, există 81 de perechi unice E#A în cazul setului de date despre laptop-uri și 12 în cazul restaurantelor.

În primul caz, datorită numărului mare de aspecte raportat la numărul de date, este mai indicată efectuarea clasificării pe entități și pe atribute în mod separat, așa cum vor confirma și rezultatele obținute, prezentate în cele ce urmează. În cel de-al doilea caz, numărul relativ mic de aspecte nu afectează performanța metodelor în mod deosebit, astfel că putem efectua clasificarea direct pe grupurile de E#A existente, fără a le mai separa.

	Nr. entități	Nr. atribute	Nr. E#A
Laptop	22	9	81
Restaurant	6	5	12

Tabelul 1.2: Date statistice cu referire la numărul de entități, atribute și perechi unice de entitate#atribut

1.2. Metoda 1 – Mașini Vector Suport

Prima metodă pentru identificarea aspectelor din recenziile oferite ca date de intrare urmează implementarea prezentată in lucrarea lui Xenos et al. [60] și folosește mașinile vector suport pentru descoperirea entităților și atributelor corespunzătoare. Având în vedere faptul că aspectele provin dintr-o listă predefinită de entități și atribute, problema se transformă în una de clasificare a textelor.

1.2.1. Aspecte teoretice

Mașinile Vector Suport reprezintă modele cu un algoritm de învățare asociat, folosite pentru clasificarea datelor. Fiind dat un set de intrări pentru antrenare, fiecare exemplu etichetat ca aparținând unei categorii din două posibile, o mașină vector suport creează un model care clasifică noi exemple în una din cele două categorii. Mai formal, ele proiectează vectorii de

intrare într-un spațiu multidimensional al caracteristicilor Z, în mod non-liniar, fiind dată o proiectare a-priori [8]. În spațiul Z se construiește apoi o suprafață liniară de decizie, cu proprietăți speciale care permit un grad de generalizare ridicat al modelului.

Pentru un astfel de model, problema principală constă în determinarea unui hiperplan care poate să generalizeze bine datele, deoarece dimensiunile vectorilor de caracteristici pot fi foarte mari. O soluție a fost oferită în anul 1965, conform Vapnik [58], prin determinarea unui hiperplan optim pentru clasele separabile. Un astfel de hiperplan este definit ca o funcție de decizie liniară cu o margine maximală între vectorii dintre celor două clase, așa cum este ilustrat și în figura 1.1. A fost observat că un număr redus de date de antrenare este necesar pentru determinarea marginilor celor două clase; aceste date reprezintă vectorii suport.

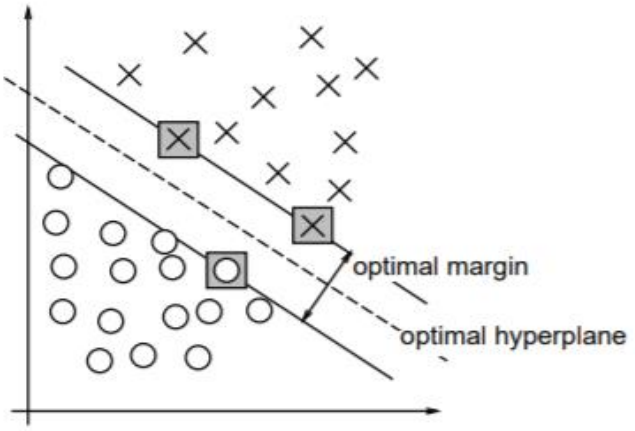

Figura 1.1: Exemplu de hiperplan ce separă două clase. Vectorii suport, evidențiați în celule, reprezintă marginea de separare dintre cele două clase (imagine preluată din [8])

A fost arătat că pentru cazul în care vectorii de antrenare sunt separați prin intermediul unui astfel de hiperplan optimal, probabilitatea de a clasificare eronată a unui vector de testare este mărginită de raportul dintre numărul de vectori suport și numărul total de vectori de antrenare (inecuația 1.1). Cu alte cuvinte, dacă se poate construi un hiperplan cu ajutorul unui număr mic de vectori suport raportat la numărul total de vectori, se spune că o problemă generalizează bine.

$$P_{error} \leq \frac{Nr.\ vectori\ suport}{Nr.\ vectori\ de\ antrenare} \qquad (1.1)$$

Este de observat faptul că acest raport nu ia în calcul dimensiunea vectorului de caracteristici, însă Cortes și Vapnik [8] au demonstrat că pentru valori mici ale raportului, cum ar fi 0.03, se poate construi un hiperplan care generalizează bine probleme într-un spațiu al trăsăturilor de dimensiune 10^9.

Hiperplanul optim poate fi definit, la nivel formal, conform formulei (1.2), ponderile w_0 și variabilele b_0 fiind determinate în urma procesului de antrenare.

$$w_0 \cdot z + b_0 = 0 \quad (1.2)$$

Ponderile se definesc ca o combinație liniară a vectorilor suport, conform formulei (1.3), unde n reprezintă numărul vectorilor suport:

$$w_0 = \sum_{i=1}^{n} \alpha_i z_i \quad (1.3)$$

În final, funcția de decizie $I(z)$ unde z reprezintă vectorul ce trebuie clasificat poate fi definită conform formulei (1.4), unde $z_i \cdot z$ reprezintă produsul scalar dintre vectorii suport și vectorul de caracteristici.

$$I(z) = sign\left(\sum_{i=1}^{n} \alpha_i z_i \cdot z + b_0\right) \quad (1.4)$$

Fiind dat un set de perechi de observații $(x_i, y_i), i = 1, \dots, l$ ca date de antrenare, unde $x_i \in R^n$ și $y \in \{-1, 1\}^l$, datele se numesc liniar separabile dacă există un vector de ponderi w și un scalar b pentru care inegalitatea (1.5) este valabilă pentru toate datele din setul de antrenare:

$$y_i(w \cdot x_i + b) \geq 1, i = 1, \dots, l \quad (1.5)$$

Vectorii suport x_i sunt acei vectori pentru care inegalitatea 1.5 devine:

$$y_i(w \cdot x_i + b) = 1 \quad (1.6)$$

Există cazuri când datele nu pot fi separate fără eroare, astfel că se dorește separarea datelor de antrenare cu un număr minim de erori. Introducem un set de variabile nenegative, denumite deviații, $\xi_i \geq 0$, iar formula 1.5 devine:

$$y_i(w * x_i + b) \geq 1 - \xi_i, \qquad i = 1 \dots l, \xi_i \geq 0. \quad (1.7)$$

În acest caz, trebuie minimizată suma deviațiilor $\phi(\xi)$ (formula 1.8), pentru o valoare $\alpha > 0$ suficient de mică.

$$\phi(\xi) = \sum_{i=1}^{t} \xi_i^{\alpha} \quad (1.8)$$

Această sumă descrie numărul de erori făcute în urma antrenării – adică inegalitatea (1.7) este valabilă pentru fiecare observație din setul de date de antrenare. Minimizând suma erorilor, se pot identifica perechile de date $(x_{i_1}, y_{i_1}), \dots, (x_{i_k}, y_{i_k})$ ce nu au fost clasificate corect, urmând apoi excluderea lor în momentul antrenării. Putem construi hiperplanul de separare pentru datele corect clasificate conform

$$\frac{1}{2} w * w + CF\left(\sum_{i=1}^{t} \xi_i^{\alpha}\right), \quad (1.9)$$

unde C reprezintă o constantă iar funcția F este o funcție convexă și monotonă. Pentru o valoare C suficient de mare și $\alpha > 0$ suficient de mică, formula (1.9) va determina hiperplanul care minimizează erorile de clasificare eronată. Este de remarcat, însă, faptul că obținerea unui astfel de hiperplan de separare reprezintă o problemă NP-Completă, astfel că pentru a simplifica, parametrul α va lua valoarea 1 [8].

Pentru a construi un hiperplan în spațiul caracteristicilor, vectorii de oferiți la intrare n-dimensionali necesită a fi transformați în vectori de caracteristici N-dimensionali, prin intermediul unei funcții de transformare

$$\phi : R^n \rightarrow R^N,$$
$$\phi(x_i) = \phi_1(x_i), \phi_2(x_i), \dots, \phi_N(x_i), \qquad i = 1, \dots, l \qquad (1.10)$$

Astfel, clasificarea vectorului x se efectuează prin transformarea lui $x \rightarrow \phi(x)$ și preluarea semnului funcției (1.11), unde $w \cdot \phi(x)$ reprezintă produsul scalar dintre ponderile w al hiperplanului de separare și vectorul de caracteristici obținut în urma apelării funcției de transformare a datelor de intrare în date relevante pentru model:

$$f(x) = w \cdot \phi(x) + b. \qquad (1.11)$$

Vectorul w poate fi scris ca o combinație liniară de vectori suport în spațiul caracteristicilor:

$$w = \sum_{i=1}^{l} y_i \alpha_i \phi(x_i), \qquad (1.12)$$

iar ecuația (1.11) devine

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i \phi(x_i) \cdot \phi(x) + b. \quad (1.13)$$

Produsul scalar $\phi(u) \cdot \phi(v) = K(u, v)$ poartă numele de **nucleu** și poate fi diferit de la o problemă de clasificare la alta. În cele din urmă, funcția de decizie ia forma finală:

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i \phi(x_i) \cdot \phi(x) + b. \quad (1.14)$$

Există mai multe modalități de a determina nucleul unor astfel de modele. Cele mai uzuale modalități sunt:

$$liniar: K(x_i, x_j) = x_i^T x_j, \qquad (1.15)$$

$$polinomial: K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0, \qquad (1.16)$$

$$funcţii\ de\ bază\ radiale\ (RBF): K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0, \qquad (1.17)$$

$$sigmoid: K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r). \qquad (1.18)$$

Considerând cele de mai sus, o mașină vector suport caută să rezolve următoarea problemă de optimizare [20]:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i,$$
$$\text{unde } y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \qquad (1.19)$$
$$\text{cu } \xi_i \geq 0.$$

1.2.2. Obținerea caracteristicilor

O etapă importantă folosind modelele mașini vector suport corespunde preprocesării datelor de intrare și formării de caracteristici. Urmând lucrarea lui Karampatsis et al. [23], am format n-grame din propozițiile de intrare (unigrame și bigrame), și am creat un lexicon ce conține n-gramele astfel obținute. Am considerat mai multe tipuri de lexicon în cazul unigramelor: unul în care un cuvânt este inclus în forma sa originală din text, unul în care am păstrat doar rădăcina cuvintelor, folosind instrumentul PorterStemmer din platforma NLTK [47], și unul în care sunt incluse cuvintele în forma lor de dicționar, conform instrumentului WordNetLemmatizer din platforma NLTK [39]. De exemplu, cuvintele *guest* și *guests* reprezintă două intrări diferite în primul lexicon, pe când, în cel de-al doilea, ambele cuvinte corespund intrării *guest*, care este rădăcina cuvântului, și la fel și pentru cel de-al treilea lexicon.

Este recomandat, conform Hsu et al. [20], ca valorile caracteristicilor să provină din intervalul $[-1, 1]$ sau $[0, 1]$, deoarece caracteristici cu valori foarte mari ar domina peste cele cu valori mici. Un alt motiv este acela că valorile mari creează probleme numerice atunci când se efectuează produsul dintre doi vectori de caracteristici. În acest sens, pentru fiecare termen din lexicon, am determinat trei măsuri: precizie, rata de regăsire (*recall*) și măsura F1, deoarece valorile acestora provin din intervalul $[0, 1]$ și oferă informații relevante cu privire la datele de antrenare. Măsurile sunt obținute conform algoritmului 1. Pentru calcularea caracteristicilor, am considerat doar acele n-grame care apar de cel puțin două ori, deoarece ele sunt mult mai reprezentative pentru identificarea statisticilor în datele folosite la antrenarea modelului pentru identificarea aspectelor.

Algoritm 1: Formarea lexiconului și obținerea scorurilor pentru fiecare termen

Necesită: *sentences*: listă de propoziții; *e#a*: listă de aspecte adnotate pentru fiecare propoziție

```
1: aspects = new Set()
2: for ea in e#a do
3:     e, a = split(ea)
4:     aspects.add(e)
5:     aspects.add(a)
6: lexicon = new Set()
```

```
 7:  for aspect in aspects do
 8:      aspect_lexicon = new Set()
 9:      aspect_sentences = new Set()
10:     for i in range(len(sentences)) do
11:         if aspect in e#a[i] do
12:             aspect_sentences.add(sentence[i])
13:     tokens, occurences = getTokens(sentences) #împarte propozițiile în n-grame
14:     aspect_tokens, aspect_occurences = getTokens(aspect_sentences)
15:     for i, token in enumerate(aspect_tokens) do
16:         if aspect_occurences[i] > 1 do   #considerăm doar n-gramele ce apar de cel puțin 2 ori
17:             index = occurences.index(token)
18:             precision = aspect_occurences[i] /  occurences[index]
19:             recall = aspect_occurences[i] / len(aspect_sentences)
20:             f1 = 2 * precision * recall / (precision + recall)
21:             aspect_lexicon.add(token, aspect_occurences[i], precision, recall, f1)
22:     lexicon.add(aspect_lexicon)
23: return lexicon
```

Observăm că vom obține un set de caracteristici diferit pentru fiecare aspect în parte (la linia 7, iterăm setul de aspecte). Acest lucru este necesar deoarece vom antrena o mașină vector suport pentru fiecare aspect existent, și deci avem nevoie de un set de caracteristici diferit pentru fiecare mașină. Lexiconul aspectului este format cu ajutorul propozițiilor adnotate cu aspectul respectiv, propoziții pe care le obținem la liniile 10-12. Liniile 13-14 corespund împărțirii textului în n-grame, după caz (unigrame, rădăcina unigramelor, bigrame), funcția returnând lista n-gramelor și frecvența aparițiilor în setul de propoziții corespunzător. Având aceste date, cele trei măsuri se calculează conform Karampatsis et al. [23]:

$$Precision_c = \frac{Nr.apariții\ în\ propoziții\ adnotate\ c}{Nr\ apariții\ în\ toate\ propozițiile} \quad (1.20),$$

$$Recall_c = \frac{Nr.apariții\ în\ propoziții\ adnotate\ c}{Nr.propoziții\ adnotate\ c} \quad (1.21),$$

$$F1_c = \frac{2 * Precision_c * Recall_c}{Precision_c + Recall_c} \quad (1.22).$$

Pentru fiecare măsură, am determinat valorile minime, maxime, media și mediana termenilor din lexicon, obținând astfel un set de 12 caracteristici pentru fiecare n-gramă formată din propozițiile date la intrare.

1.2.3. Selectarea modelului

După formarea caracteristicilor, avem nevoie de un model care va clasifica în mod corect datele de intrare. O mașină vector suport poate grupa exemplele în doar două categorii, pe când noi avem cel mult $|E| * |A|$categorii (sau $|E\#A|$categorii dacă nu efectuăm separarea entităților de atribute). Pentru a rezolva această problemă, vom antrena o mașină pentru fiecare aspect în parte. Cele două categorii reprezentative pentru fiecare mașină vor semnifica dacă un exemplu de recenzie este sau nu adnotat cu aspectul respectiv. Pentru fiecare mașină vector suport, vom eticheta exemplele de antrenare cu 1 (dacă aspectul este exprimat în exemplu) sau 0 (dacă aspectul nu este exprimat).

Pentru a forma modelul, avem de selectat unul din cele 4 nuclee: liniar, sigmoid, polinomial sau funcții de bază radiale, variabila C de estimare a erorii și restul parametrilor specifici unui nucleu (parametrul γ specific nucleului sigmoid, polinomial și RBF, parametrul d specific nucleului polinomial, parametrul r specific nucleului sigmoid și polinomial).

Scopul este să alegem parametrii, pornind de la datele de antrenare, pentru a putea prezice cât mai corect datele de testare. Nu dorim să obținem acuratețe cât mai mare pe datele de antrenare în urma alegerii parametrilor, deoarece poate să apară fenomenul de specializare (*overfitting*). Pentru a evita acest lucru, aplicăm o metodă de validare încrucișată (*cross-validation*) prin care o parte din datele de antrenare vor fi folosite pentru obținerea parametrilor, și o parte va fi considerată necunoscută și va fi folosită pentru testarea modelului. Urmând metoda din Hsu et al. [20], se împart datele în k seturi egale, și selectăm $k - 1$ seturi pentru antrenare și un set „necunoscut” pentru testarea modelului. Performanța modelului testat pe setul „necunoscut” de date va reflecta performanța folosind un set de date independent.

Parametrii sunt aleși urmând metoda prezentată de Hsu et al. [20]. Se aplică o metodă ce poartă numele de căutare grilă (*grid search*, în literatura de specialitate), și încearcă să efectueze o căutare exhaustivă pe întreg domeniul de valori posibile, pentru a găsi cele mai bune valori pentru parametrii modelului. Există, însă, foarte multe valori posibile, iar timpul de rulare ar fi

foarte mare pentru a determina cele mai bune valori. Astfel, căutarea poate fi făcută în două etape: prima corespunde unei căutări mai generale, în care se aleg valori la anumite intervale din domeniu, pentru a găsi o regiune „bună" pentru care hiperparametrii estimează satisfăcător modelul, și a doua corespunde unei căutări mai fine în jurul valorilor identificate la prima etapă.

Luăm ca exemplu funcțiile de bază radiale, pentru care avem de estimat setul de parametri (C, γ). În prima etapă, selectăm o secvență de valori ce cresc exponențial, ca de exemplu $C = 2^{-5}, 2^{-3}, 2^{-1}, \dots, 2^{13}, 2^{15}$ și $\gamma = 2^{-15}, 2^{-13}, 2^{-11}, \dots, 2^{1}, 2^{3}$. Presupunem că în urma căutării, găsim că $C = 2^{3}$ și $\gamma = 2^{-5}$ oferă cea mai bună acuratețe folosind validarea încrucișată amintită anterior. Apoi, la etapa a doua mai efectuăm o căutare grilă în vecinătatea valorilor determinate anterior, ca de exemplu $C = 2^{1}, 2^{1.25}, 2^{1.5}, \dots, 2^{4.75}, 2^{5}$ și $\gamma = 2^{-7}, 2^{-6.75}, 2^{-6.5}, \dots, 2^{-3.25}, 2^{-3}$, pentru a îmbunătăți acuratețea obținută la etapa 1.

Prin această metodă de căutare grilă, fiecare mașină vector suport creată va fi specializată pentru identificarea aspectelor corespunzătoare. Valorile obținute care maximizează acuratețea modelului la validarea încrucișată au fost incluse în Anexa 1. Este de remarcat faptul că mașinile vector suport folosesc nuclee diferite pentru aspecte diferite pentru a aproxima cel mai bun model.

Nucleul liniar a fost folosit pentru antrenarea mașinilor vector suport corespunzătoare aspectelor *Shipping* și *Ports* (aspecte corespunzătoare setului de date Laptop), acest fapt indicând că datele, în cazul acestor două aspecte, sunt separabile liniar. Acest lucru poate fi explicat prin faptul că avem puține propoziții adnotate cu aceste aspecte în setul de date folosit: mai exact, sunt 12 apariții ale aspectului *Shipping* și 6 apariții ale aspectului *Ports* (Anexa 1). Hiperparametrul de estimat în cazul nucleului liniar este doar constanta *C*, ce poate lua valori dintr-un domeniu larg.

În multe alte cazuri, rezultatul cel mai bun a fost obținut aplicând nucleul funcțiilor de bază radiale RBF. Acesta se folosește atunci când datele nu pot fi separate în mod liniar. Datele sunt proiectate într-un spațiu multidimensional, urmând apoi a se efectua operațiile pentru clasificarea acestora [4]. Dezavantajul folosirii nucleului RBF este acela că procesul de clasificare este mult mai lent, deoarece operațiile de proiectare în spațiul multidimensional sunt mai costisitoare. Spațiul caracteristicilor poate avea un număr infinit de dimensiuni [4], deci nu avem control asupra acestui număr. Este introdus hiperparametrul $\gamma = \frac{1}{2\sigma^2}$ ce trebuie estimat, în plus față de constanta C.

Nucleul polinomial reprezintă o altă alternativă pentru clasificarea aspectelor. Nu este la fel de folosit precum nucleul RBF [4], deoarece funcționează pe același principiu: datele de intrare vor fi proiectate într-un spațiu multidimensional, ce poate fi infinit, și nu garantează că rezultatul obținut va fi mai bun decât folosind nucleul RBF, fapt confirmat și în urma experimentelor efectuate, unde nucleul polinomial a determinat obținerea unui scor al acurateței mai bun în doar două cazuri: pentru aspectul *OS* și aspectul *Keyboard* (Anexa 1). Nucleul polinomial se dovedește a fi superior atunci când gradul polinomului d este mic; hiperparametrul ia valoarea 3 în ambele cazuri. Acest lucru este confirmat și în lucrarea lui Goldberg și Elhadad [13], nucleul polinomial cu gradul mic fiind folosit cu succes în aplicații de procesare a limbajului natural, reprezentând și o metodă mai rapidă de calculare a funcțiilor de decizie. Am variat hiperparametrul d cu valorile 3, 4 și 5 în urma procesului de căutare grilă, testând astfel modelul pe diferite cazuri unde gradul polinomului ia valori mici.

Nucleul sigmoid maximizează acuratețea obținută în urma procesului de validare încrucișată pentru multe din modelele create. Acest nucleu mai este cunoscut și sub numele de nucleul perceptronului multistrat, provenind din domeniul rețelelor neuronale, unde funcția reprezentând nucleul sigmoid e folosită ca activator [33]. Mașinile vector suport care folosesc nucleul sigmoid sunt echivalente rețelelor neuronale cu două straturi de neuroni. Nucleul sigmoid ia valori pozitive, iar în lucrarea lui Boughorbel et al. [3] a fost arătat că nucleele pozitiv definite oferă rezultate bune în diferite experimente. Am avut de estimat, similar nucleului RBF, valoarea constantei C și variabila γ care în acest caz poartă numele de pantă. Cel mai adesea, panta ia valoarea $\frac{1}{D}$, unde D reprezintă numărul de dimensiuni obținut în urma proiecției datelor [3]

1.3. Metoda 2 – Rețele Neuronale de Convoluție

A doua metodă pentru identificarea aspectelor presupune modelarea unei rețele neuronale de convoluție pentru fiecare aspect în parte, similar metodei anterioare. Vom urma metoda prezentată în lucrarea lui Toh și Su [56] pentru formarea arhitecturii rețelei.

O rețea neuronală de convoluție reprezintă un model de rețea de tip feedforward, sau perceptron multistrat cu multiple straturi ascunse, și este folosită cu succes în domenii precum vedere artificială [29] sau recunoașterea vorbirii [15], dar nu numai, fiind la fel de eficientă și

aplicată metodelor de procesare a limbajului natural [7][26]. Această metodă face parte din familia de metode de învățare automată ce poartă numele de învățare adâncă (*deep learning*). Numele familiei provine de la faptul că există multe straturi într-o astfel de rețea, aceasta fiind o proprietate caracteristică rețelelor neuronale adânci. Pornind de la stratul de intrare, o rețea neuronală adâncă creează o proiecție de neuroni virtuali și atribuie ponderi aleatorii conexiunilor dintre aceștia. În urma procesului de antrenare, aceste ponderi vor fi actualizate până când o rețea reușește să identifice în mod corect un tipar între datele de intrare și etichetele corespunzătoare [51].

Arhitectura rețelei poate să difere de la o implementare la alta, în funcție de natura problemei ce trebuie rezolvată. Rețeaua implementată în această aplicație este alcătuită din două componente.

Prima componentă se ocupă cu extragerea caracteristicilor din recenziile date la intrare. Sunt caracteristice două operații, corespunzătoare pentru două straturi ale rețelei: o operație de convoluție și una de agregare. Prin operația de convoluție se obțin trăsături de dimensiuni mai mici decât cele oferite la intrare, spre deosebire de alte straturi ale rețelelor neuronale, cu ajutorul unor filtre. Dimensiunea trăsăturilor finale depinde de dimensiunea acestor filtre, care poate să varieze. Filtrele se obțin în urma procesului de antrenare al rețelei. Prin operația de agregare se efectuează o reducere a dimensiunii datelor obținute la stratul de convoluție.

A doua componentă se ocupă de clasificarea datelor de intrare, folosind caracteristicile provenite în urma procesului de agregare. În acest strat se determină probabilitatea unui exemplu de a aparține unei clase.

O rețea neuronală $f_\theta(\cdot)$ cu L straturi, unde θ sunt parametrii rețelei, poate fi exprimată ca o compunere de funcții $f_\theta^l(\cdot)$, cu $l = 1, \dots, L$ [7]. Fiecare funcție f_θ^l reprezintă un strat al rețelei.

$$f_\theta(\cdot) = f_\theta^L\left(f_\theta^{L-1}(\dots f_\theta^1(\cdot)\dots)\right) \quad (1.23)$$

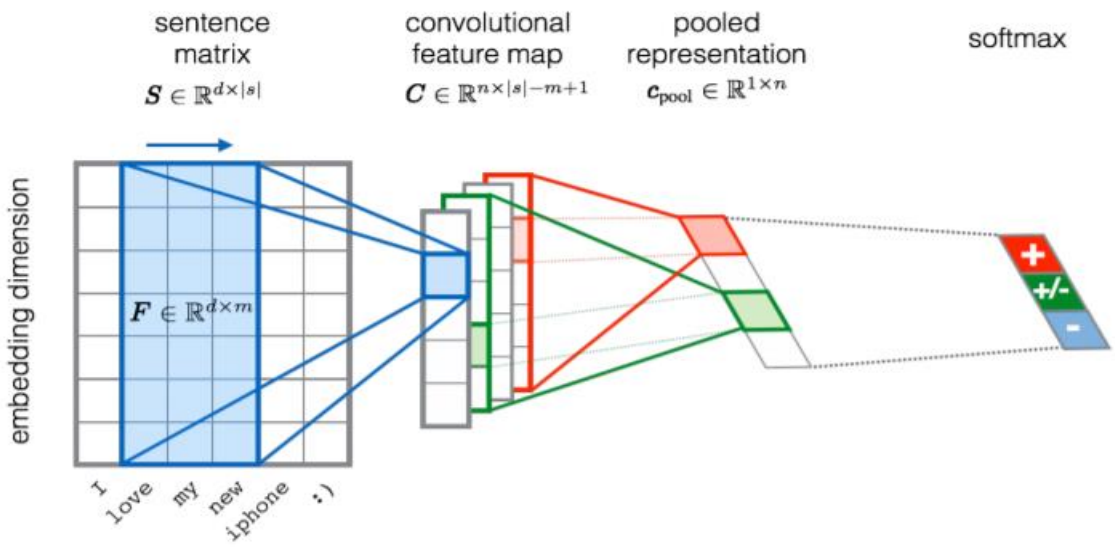

Figura 1.2: Arhitectura rețelei implementate, formată dintr-un strat de convoluție, un strat de agregare și un strat de clasificare a vectorilor obținuți [52]

Pentru implementarea rețelei, am folosit platforma Keras [6], ce este bazată pe platforma TensorFlow. Keras permite realizarea unor arhitecturi de rețele neuronale într-o manieră ușoară și rapidă, ceea ce facilitează experimentarea. Interfața pentru programare este realizată pentru a fi simplu de folosit, și concepută pentru a minimiza numărul de pași pe care un utilizator trebuie să îi facă pornind de la datele de intrare și până la modelul final. Structura fundamentală din Keras este modelul, ce reprezintă un mod de a organiza straturile unei rețele neuronale. Cel mai simplu este modelul secvențial, ce presupune suprapunerea liniară a straturilor, care va fi folosit și în această implementare.

Tensorflow este o platformă de învățare automată, capabilă de a opera cu sisteme complexe în medii eterogene. Aceasta oferă capabilitatea de a folosi diferite arhitecturi ale procesoarelor și plăcilor video în implementări eficiente, capabile să proceseze cantități mari de date în timp redus. Tensorflow permite realizarea unor rețele neuronale într-o manieră ușoară și rapidă. Reprezentarea informației este realizată prin intermediul unui graf, care înregistrează toate stările și etapele de calcul ce vor fi parcurse în mod explicit. Acest mod de funcționare diferă de alte sisteme prin două detalii semnificative: modelul suportă rularea concurentă a mai multor instrucțiuni din cadrul aceluiași graf , și valorile vectorilor pot avea o stare mutabilă care poate fi cunoscută între execuții diferite ale aceluiași graf – acest avantaj este unul crucial pentru

eficiența unor sisteme care sunt antrenate, precum a rețelelor neuronale care au un număr foarte ridicat de straturi sau a sistemelor care procesează un volum mare de date [1].

1.3.1. Vectorii de caracteristici

Datele de intrare sunt propozițiile provenite din recenzii, ce sunt tratate ca și secvențe de cuvinte. Fiecare cuvânt este indexat și se formează un vocabular V. Cuvintele sunt transmise rețelei sub formă de indici, un indice reprezentând intrarea din dicționar a cuvântului respectiv. Acești indici nu oferă, însă, suficientă informație despre un cuvânt. Pentru fiecare cuvânt dintr-o propoziție, se dorește obținerea unui vector de caracteristici.

O metodă de a obține caracteristici presupune folosirea vectorilor de încorporare (*word embeddings*). S-a arătat, în articolele lui Liu et al. [34] și Pennington et al. [42] că folosind vectori de încorporare se obțin rezultate foarte bune în diferite probleme de procesare a limbajului natural, necesitând doar pre-procesări minime ale datelor de intrate pentru folosirea acestora. Pentru a obține caracteristici din recenzii, am folosit vectorii de încorporare GloVe [12], ce conțin un vocabular de 400.000 de cuvinte, și, pentru fiecare cuvânt, un vector de caracteristici de dimensiune d ce poate să varieze (50, 100, 200, 300 – un lexicon separat pentru fiecare dimensiune a vectorilor de caracteristici). Vocabularul a fost obținut prin prelucrarea unui corpus alcătuit din 6 miliarde de cuvinte (1.4 miliarde provenind din corpus-ul Wikipedia din anul 2014 și 4.3 miliarde din corpus-ul Gigaword5).

GloVe, sau Vectori Globali pentru Reprezentarea Cuvintelor, reprezintă vectori ce conțin date statistice despre aparițiile cuvintelor într-un corpus [42]. Datele se obțin aplicând tehnici de reducere a dimensiunii asupra matricelor ce conțin date referitoare la co-aparițiile cuvintelor dintr-un corpus. Cu alte cuvinte, valorile exprimă cât de des apare un cuvânt în contextul altui cuvânt. Se pornește de la ideea că raportul probabilităților co-aparițiilor dintre perechi de câte două cuvinte încifrează o semnificație în legătură cu relația dintre cele două cuvinte. Pentru a exemplifica, vom considera un scenariu practic. Înainte de aceasta, vom introduce câteva notații.

Fie matricea X de co-apariție a cuvintelor dintr-un corpus. Elemenul X_{ij} reprezintă de câte ori apare cuvântul j în contextul lui i. Contextul semnifică acele cuvinte din jurul unui cuvânt țintă. Pentru formarea fișierelor GloVe, s-a considerat un context de 10 cuvinte la stânga și 10 cuvinte la dreapta cuvântului țintă. Astfel, $X_i = \sum_k X_{ik}$ reprezintă numărul de cuvinte care

apar în total în contextul lui i. Se definește $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$ probabilitatea ca un cuvânt *j* să apară în contextul lui *i*.

Considerăm două cuvinte, ca de exemplu *i = ice* și *j = steam.* Relația dintre aceste două cuvinte poate fi stabilită studiind raporturile probabilităților de co-apariție ale cuvintelor *i* și *j* cu alte cuvinte de probă, k. Pentru cuvinte *k* care au legătură (se găsesc în contextul cuvântului) cu $i = ice$ dar nu și cu $j = steam$, cum ar fi $k = solid$, raportul $\frac{P_{ik}}{P_{jk}}$ va fi unul mare. În situația inversă când un cuvânt *k*, de exemplu *gas* are legătură cu *j* dar nu și cu *i*, raportul $\frac{P_{ik}}{P_{jk}}$ este unul mic. În cazul în care avem cuvinte care fie nu apar deloc în contextele lui *i* și *j*, fie apar în mod egal, cum ar fi *fashion*, respectiv *water*, raportul $\frac{P_{ik}}{P_{jk}}$ va fi apropiat de 1.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figura 1.3: Valorile pentru probabilitățile de co-apariție ale unui cuvânt k în contextele lui ‚ice' și ‚steam'[12]

După cum se poate observa și în figura 1.3, în contextul cuvântului *ice* se găsește mult mai des cuvântul *solid* decât *gas,* iar în contextul lui *steam* vom găsi *gas* mai des decât *solid,* proprietăți fizice care, de altfel, corespund și realității. O proprietate comună a celor două cuvinte, *water*, se găsește la fel de mult în ambele contexte, pe când un cuvânt care nu prezintă o legătură directă, cum ar fi *fashion*, va fi la fel de rar în ambele contexte – în aceste două cazuri, raportul dintre probabilitățile de apariție va fi apropiat de 1, semnalându-se faptul că aceste cuvinte nu sunt potrivite pentru a diferenția conceptul exprimat de un cuvânt (*ice*) față de un alt concept (*steam*).

Aceste raporturi vor fi folosite pentru a construi un model

$$F(w_i, w_j, w_k) = \frac{P_{ik}}{P_{jk}} \qquad (1.24),$$

unde $w \epsilon R^d$ sunt vectorii de caracteristici de dimensiune d. Se dorește ca F să încifreze informația din raport în spațiul vectorial. Se iau în considerare doar acele funcții F care depind de diferența vectorilor celor două cuvinte țintă i și j, iar modelul (1.24) devine:

$$F(w_i - w_j, w_k) = \frac{P_{ik}}{P_{jk}}. \quad (1.25)$$

Am obținut în membrul drept un scalar, iar în membrul stâng, parametrii funcției F sunt vectori. Pentru a păstra structura liniară a vectorilor, se consideră produsul vectorial al parametrilor, iar modelul devine

$$F((w_i - w_j)^T w_k) = \frac{P_{ik}}{P_{jk}}. \quad (1.26)$$

Într-o matrice de co-apariții, distincția dintre un cuvânt țintă și un cuvânt ce provine din context este una arbitrară, astfel că se pot inter-schimba rolurile dintre ele, deci modelul final ar trebui să fie invariant la acest mod de etichetare. Modelul (1.26) nu prezintă această proprietate. Dacă se consideră că funcția F este un homomorfism al mulțimilor $(R, +)$ și $(R, *)$, atunci modelul devine

$$F((w_i - w_j)^T w_k) = \frac{F(w_i{}^T w_k)}{F(w_j{}^T w_k)}, \quad (1.27)$$

$$F(w_i{}^T w_k) = P_{ik} = \frac{X_{ik}}{X_i}. \quad (1.28)$$

Deoarece $F(w_i{}^T w_k)$ = exp, obținem că

$$w_i{}^T w_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i). \quad (1.29)$$

Termenul $\log(X_i)$ este independent de k, astfel că îl putem considera o constantă, sau un deplasament (*bias*) b_i al vectorului w_i. Adăugând un deplasament vectorului w_k, obținem un model simetric.

$$w_i{}^T w_k + b_i + b_k = \log(X_{ik}) \quad (1.30)$$

Pentru a evita cazurile când argumentul logaritmului este 0, iar algoritmul nu ar converge, putem adăuga o unitate pentru ca $\log(X_{ik})$ să devină $\log(1 + X_{ik})$. O problemă a acestui model este aceea că va cântări toate co-aparițiile în mod egal, chiar și pe acelea care apar foarte rar. Autorii articolului [42] folosesc o metodă de regresie ce calculează suma celor mai mici pătrate ponderate. Este introdusă o funcție de cost $f(X_{ij})$, iar modelul final devine

$$J = \sum_{i,j=1}^{V} f(X_{ij}) (w_i{}^T w_j + b_i + b_j - \log(X_{ij}))^2, \quad (1.31)$$

unde V reprezintă dimensiunea vocabularului. Funcția de cost f a fost calculată astfel:

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha}, & x < x_{max} \\ 1, & altfel \end{cases} \quad (1.32)$$

Diverse experimente și aproximări ale parametrilor au fost făcute, pentru ca în final lui x_{max} să îi fie setată valoarea 100, iar lui α valoarea ¾ [42]. În final, se obțin două seturi de vectori de caracteristici: W și W'. Dacă matricea de co-apariții X este simetrică, atunci și cei doi vectori de caracteristici sunt egali, diferența constând doar în ordinea în care cuvintele vor apărea în vocabular. Acești doi vectori sunt însumați, deoarece pe diferite rețele neuronale acest lucru a dus la o îmbunătățire a performanței [42], iar rezultatul obținut reprezintă vectorul de încorporare.

Odată obținuți vectorii de caracteristici pentru fiecare cuvânt, datele noastre de intrare vor fi de forma unor matrice de propoziții $S \epsilon R^{|s|*d}$, unde $|s|$ reprezintă lungimea maximă a numărului de cuvinte dintr-o propoziție din setul de date, iar d reprezintă dimensiunea vectorilor de caracteristici. În cazul în care avem propoziții cu număr de cuvinte mai mic decât $|s|$, se vor completa ultimele rânduri cu zero-uri.

În mod formal, pentru fiecare cuvânt $w \epsilon V$, unde V este vocabularul obținut din datele de antrenare, se aplică o funcție $LT_W(\cdot)$ care extrage din matricea $W \in R^{|V|*d}$ de caracteristici încorporate un vector d-dimensional de caracteristici. Notăm $\langle W \rangle_w$ linia din matricea W, de la indexul w, și obținem următoarea formulă:

$$LT_W(w) = \langle W \rangle_w \quad (1.33)$$

Fiind dată o propoziție, sau o secvență de $|s|$ cuvinte $s = [w_1, w_2, \dots, w_{|s|}]$, formula (1.33) aplicată întregii propoziții devine

$$S = LT_W(s) = \begin{pmatrix} \langle W \rangle_{w_1} \\ \langle W \rangle_{w_2} \\ \vdots \\ \langle W \rangle_{w_{|s|}} \end{pmatrix} \quad (1.34)$$

Matricea obținută aplicând formula (1.34) va fi transmisă stratului de convoluție al rețelei neuronale.

1.3.2. Stratul de convoluție

Intuiția din spatele operației de convoluție poate fi exprimată prin intermediul unui scenariu propus de Goodfellow et al. [14], în care presupunem că dorim să aflăm poziția unui satelit efectuând măsurători obținute cu ajutorul unui laser. O astfel de măsurătoare poate fi exprimată sub forma $x(t)$, și reprezintă poziția satelitului la timpul t. Măsurătorile obținute vor fi perturbate de zgomot, și din acest motiv se dorește colectarea mai multor măsurători și efectuarea unei medii a acestora. Simpla medie aritmetică nu este de ajuns deoarece măsurători mai recente sunt mai importante decât cele din momentele anterioare lor, deci este necesar un set de ponderi $F(a)$ unde a reprezintă vârsta măsurătorii. Aplicând media aritmetică ponderată la fiecare moment t înregistrat, se va obține o estimare c de forma:

$$c(t) = \int x(a)F(t-a)da. \quad (1.35)$$

Formula (1.35) reprezintă operația de convoluție. La nivel formal, x este reprezentat de datele de intrare pentru stratul de convoluție al rețelei, iar F va fi un filtru, sau un nucleu. Acest nucleu are dimensiuni mai mici decât dimensiunea intrărilor în rețea. Acesta se va „suprapune" peste datele de intrare și va crea noi caracteristici de convoluție la diferite momente de timp t. În cazul nostru, lucrăm cu șiruri de cuvinte provenite din recenzii, deci abstractizând exemplul prezentat anterior, putem considera că o măsurătoare *x(t)* reprezintă cuvântul de la poziția *t* din șirul de cuvinte. Scopul acestui strat al rețelei este acela de a identifica tipare, sau secvențe de cuvinte ce se repetă în recenziile pe care dorim să le analizăm [52].

Putem rescrie formula (1.35) în raport cu datele obținute în stratul de input pentru rețeaua noastră neuronală, iar o trăsătură de convoluție se poate defini astfel:

$$c_i = f_\theta(F \cdot S_{i:i+h-1} + b) \quad (1.36)$$

$S_{i:i+h-1}$ reprezintă matricea formată în stratul de input obținută prin extragerea liniilor $i{:}\, i+h-1$, unde h reprezintă dimensiunea ferestrei filtrului de convoluție aplicat.

$$S_{i:i+h-1} = \begin{pmatrix} \langle W \rangle_{w_i} \\ \langle W \rangle_{w_{i+1}} \\ \vdots \\ \langle W \rangle_{w_{i+h-1}} \end{pmatrix} \quad (1.37),$$

Filtrul $F \in R^{h\,x\,d}$ reprezintă o matrice cu h linii (dimensiunea ferestrei) și d coloane (dimensiunea vectorului de caracteristici obținut la stratul anterior), iar variabila b reprezintă un deplasament. Aceste două variabile vor fi estimate în momentul antrenării rețelei.

Filtrul este aplicat fiecărei ferestre de cuvinte $\{S_{1:h}, S_{2:h+1}, \dots, S_{|s|-h+1:|s|}\}$, și se obțin caracteristicile de convoluție $c = [c_1, c_2, \dots, c_{|s|-h+1}]$, cu $c \in R^{|s|-h+1}$ în urma produsului scalar efectuat dintre o fereastră de cuvinte și filtru. Se observă faptul că dimensiunea vectorului de caracteristici de convoluție depinde de dimensiunea filtrului folosit.

În figura 1.3 este reprezentat un exemplu simplu pentru a ilustra modul de obținere al acestor caracteristici. Un filtru de dimensiune $h*d$ (în cazul nostru, $h = 2$ se consideră dimensiunea ferestrei filtrului, iar $d = 5$ se consideră lungimea vectorului de caracteristici) se „suprapune" cu matricea de caracteristici. Având propoziția *I like this movie very much!*, se extrage prima fereastră de cuvinte, corespunzătoare lui *I* și *like*. Se efectuează produsul scalar dintre cele două matrice, iar rezultatul obținut reprezintă primul vector de caracteristici de convoluție. Procesul se repetă pentru fiecare fereastră de două cuvinte succesive: *like* și *this, this* și *movie* ș.a.m.d. Dimensiunea matricei de convoluție obținut la final corespunde cu de câte ori a fost aplicat produsul scalar pe propoziția de intrare. Pentru o propoziție cu un număr de $|s|$ cuvinte, vom avea în final un număr de $|s| - h + 1$ de vectori caracteristici de convoluție.

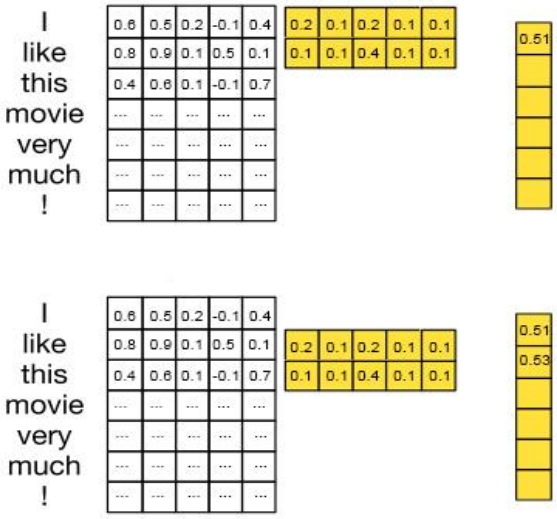

Figura 1.3: Exemplu de obținere a caracteristicilor de convoluție, aplicând un filtru de dimensiune 2x5 în mod succesiv pe ferestre de dimensiune 2 (imagine preluată din articolul [25])

Ca și activator pentru stratul de convoluție, am folosit funcția liniară rectificată ReLU, deoarece, așa cum este arătat și în articolul lui Nair și Hinton [38], viteza de antrenare este mai mare și în unele cazuri, rezultatele oferă o acuratețe mai bună decât folosind alți activatori.

Funcția de activare ReLU a fost introdusă în lucrarea [17], având atât fundamente matematice cât și biologice. Acest activator este folosit la scară largă în multe aplicații de învățare adâncă [32]. Funcția poate fi scrisă ca $f(x) = \max(0, x)$, adică pentru orice valoare $x < 0$ din domeniul funcției, rezultatul va fi 0, iar pentru $x \geq 0$, funcția devine liniară. Spre deosebire de alți activatori, cum ar fi activatorul sigmoid, ReLU oferă avantajul că gradientul nu dispare, fapt asigurat de funcția $f(x)$ pentru care valoarea gradientului va fi constantă pentru oricare ar fi $x > 0$. Un alt avantaj este acela că reprezentarea va fi rară, adică nu toți neuronii oferă un răspuns în același timp, proprietatea având fundamente biologice; acest fenomen apare când $x \leq 0$, spre deosebire de activatorul sigmoid, pentru care mereu se obține o valoare nenegativă. Dezavantajul va fi acela că daca există un număr mare de neuroni care nu se activează, adică valoarea funcției $f(x)$ este 0, atunci rețeaua nu mai poate să învețe, fenomen ce poartă numele de *dying ReLu* [36].

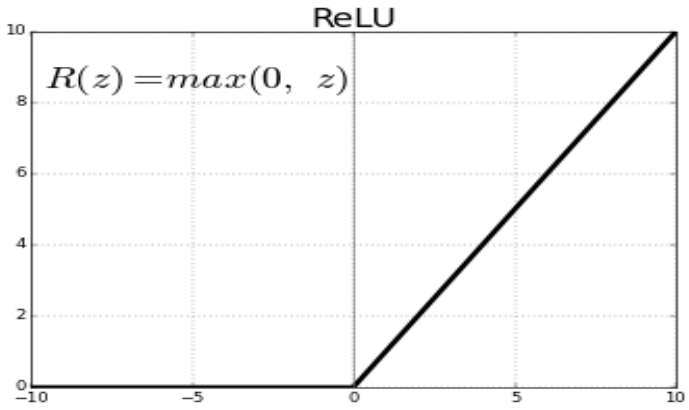

Figura 1.4: Funcția de activare ReLU [50]

1.3.3. Stratul de agregare

A doua operație specifică rețelelor neuronale de convoluție reprezintă agregarea vectorilor de caracteristici calculați la stratul anterior, obținându-se o sumarizare statistică a datelor. Există mai multe metode de agregare, cele mai folosite fiind extragerea valorii maxime dintr-un vector, sau media valorilor. În lucrarea lui Kalchbrenner et al. [22] a fost folosită o metodă de agregare generalizată, prin care se extrag cele mai mari k valori din vector. Pentru modelarea rețelei noastre, am folosit doar extragerea valorii maxime.

Operația de agregare este necesară deoarece aceasta asigură că datele vor fi în cea mai mare măsură invariante la diferite translații ale datelor de intrare [14]. Această proprietate este importantă deoarece oferă informații despre existența unei trăsături într-un exemplu și nu despre locația sa, în condițiile în care existența este de interes pentru modelul generat.

Un alt avantaj al acestei metode este că reducând volumul de date, va crește și performanța modelului din punct de vedere al stocării memoriei și al calculelor efectuate. Dintr-un vector de caracteristici de convoluție obținut la stratul anterior vom păstra doar valoarea maximă înregistrată în acesta.

La nivel formal, agregarea reprezintă o funcție $pool: R^d \rightarrow R,$, unde d reprezintă lumgimea unui vector de caracteristici c, iar vectorul rezultat în urma aplicării operațiilor de agregare poate fi scris sub forma

$$c_{pooled} = \begin{bmatrix} pool\big(\alpha(c_1 + b_1 * e)\big) \\ \vdots \\ pool\big(\alpha(c_n + b_n * e)\big) \end{bmatrix},$$

unde α reprezintă funcția activator de la stratul precedent, $c_i, i = 1, \dots, n$ reprezintă vectorul de caracteristici obținut la stratul de convoluție pentru propoziția i, modificat de deplasamentul b_i (fiecare element din vector este modificat prin adăugarea valorii deplasamentului; e reprezintă un vector unitate de dimensiunile vectorului c_i), iar n reprezintă numărul de neuroni de pe stratul ascuns anterior.

1.3.4. Stratul softmax

Ultimul strat din rețeaua neuronală de convoluție corespunde stratului softmax, care primește vectorul de caracteristici obținut în stratul de agregare și determină funcția de

probabilitate peste categoriile posibile. În cazul nostru, antrenăm o rețea neuronală de convoluție pentru fiecare aspect existent, deci etichetele vor reprezenta dacă o recenzie este adnotată cu aspectul respectiv (eticheta 1) sau nu (eticheta 0).

Notăm x vectorul de caracteristici obținut la stratul precedent. Se calculează

$$P(y = j \mid x, s, b) = softmax_j\,(x^T w + b) = \frac{e^{x^T w_j + b_j}}{\sum_{k=1}^{K} e^{x^T w_k + b_k}}, \quad (1.38)$$

unde K reprezintă numărul de categorii, iar w_k și b_k reprezintă vectorul de ponderi, respectiv deplasamentul pentru categoria k.

Am folosit ca și activator funcția sigmoidă, deoarece codomeniul acesteia este $(0, 1)$, iar probabilitățile iau la rândul lor valori din acest interval.

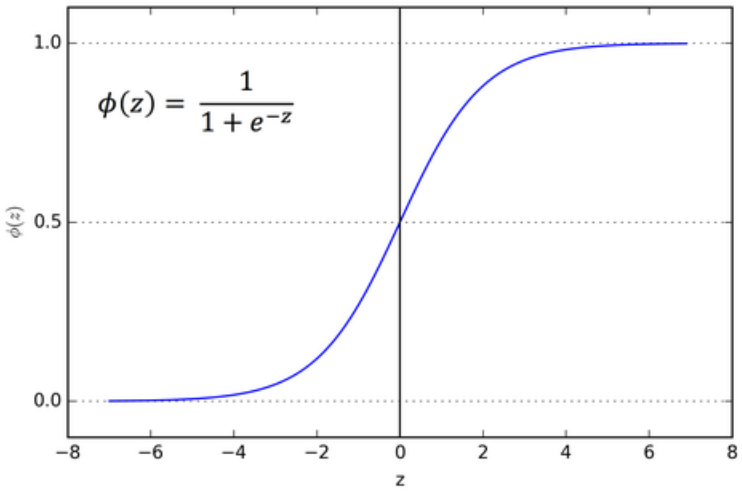

Figura 3: Funcția de activare sigmoidă [22]

1.3.5. Antrenarea rețelei

Scopul antrenării unei rețele neuronale este de a reuși, pentru datele de intrare x, să se prezică etichetele y asociate, prin intermediul unor parametri ai rețelei θ. Se dorește găsirea parametrilor θ pentru care se minimizează eroarea de clasificare. Sunt mai multe metode de a alege această funcție de pierdere a calității (*loss function*), o metodă populară pentru cazurile când există mai multe clase fiind metoda entropiei încrucișate categorice (*categorical cross-entropy*). Pentru un număr de N date de antrenare, și CE reprezentând metoda entropiei încrucișate, funcția de pierdere a calității ia forma:

$$loss(\theta) = \frac{1}{N}\sum_{i=1}^{N} CE \quad (1.39)$$

Folosind metoda entropiei încrucișate, rețeaua va calcula probabilitatea ca un exemplu oferit la intrare să aparțină categoriilor de clasificare. Formal, entropia încrucișată poate fi exprimată în baza formulei (1.38), unde $softmax_i$ reprezintă rezultatul obținut în urma aplicării stratului softmax al rețelei, iar C reprezintă numărul de clase existent.

$$CE = -\sum_{i}^{C} y_i \log(softmax_i) \quad (1.40)$$

Vectorul $y = (y_1, y_2, \dots, y_C)$ al etichetării claselor este un vector de tipul *one-hot* în care toate elementele iau valoarea zero, cu excepția uneia, care va lua valoarea 1; poziția la care această valoare se găsește în vector este un indicator pentru clasa reprezentată. În cazul entropiei încrucișate categorice, se elimină din sumă toate elementele pentru care elementele y_i din vector iau valoarea 0 și se păstrează doar elementul pentru care $y_j = 1$, iar conform ecuațiilor (1.38) și (1.40), formula devine:

$$CE = -\log\left(softmax_j\right) = -\log\left(\frac{e^{x^T w_j + b_j}}{\sum_{k=1}^{K} e^{x^T w_k + b_k}}\right). \quad (1.41)$$

Revenim la funcția de pierdere a calității, pe care dorim să o minimizăm. Gradientul descendent reprezintă o metodă de determinare a minimului local al unei funcții, deci acesta va trebui determinat pentru entropia încrucișată.

Notăm $s_j = x^T w_j + b_j$ din formula (1.41). Pentru clasa pozitivă, adică $y_j = 1$, gradientul poate fi exprimat astfel:

$$\frac{\partial}{\partial s_p} = \left(\frac{e^{s_p}}{\sum_{j=1}^{C} e^{s_p}} - 1\right) \quad (1.42)$$

Pentru clasa negativă, gradientul poate fi exprimat astfel:

$$\frac{\partial}{\partial s_n} = \left(\frac{e^{s_n}}{\sum_{j=1}^{C} e^{s_n}}\right) \quad (1.43)$$

În cele din urmă, gradientul funcției de pierdere a calității poate fi exprimat astfel:

$$g = \frac{1}{N}\nabla_\theta \sum_{i=1}^{N} CE = \frac{1}{N}\left(\frac{\partial}{\partial s_p} + (N-1)\frac{\partial}{\partial s_n}\right) \quad (1.44)$$

Pentru optimizarea parametrilor rețelei neuronale de convoluție am folosit algoritmul Adam, o metodă eficientă din punct de vedere computațional și adecvată pentru probleme cu un volum mare de date. Numele de Adam este un derivat de la estimarea momentelor adaptive (*adaptive moment estimations*). Acest algoritm prezintă o metodă de optimizare a parametrilor, pornind de la gradientul descendent raportat la parametrii θ pe care îi actualizează în mod iterativ. Metoda a fost introdusă în lucrarea lui Kingma și Ba [27] în anul 2015, devenind o metodă populară în antrenarea rețelelor neuronale.

Algoritm 2: Adam

Necesită: η dimensiunea unui pas (implicit 0.001)

Necesită: β_1, β_2 rata de degenerare exponențială în intervalul [0,1) (implicit 0.9, 0.999)

Necesită: δ constantă pentru stabilizare numerică (implicit 10^{-8})

Necesită: θ_0 parametrii modelului de optimzat

1: $s_0 = 0, r_0 = 0$ primul și al doilea moment al gradientului

2: $t = 0$

3: **while** θ_t nu converge **do**

4: t = t+1

5: $g_t = \frac{1}{N} \nabla_\theta \sum_{i=1}^{N} CE$ (gradientul obținut cf. Formulei (43) raportat la parametrii θ_{t-1})

6: $s_t = \beta_1 \cdot s_{t-1} + (1 - \beta_1) \cdot g_t$ actualizăm primului moment, cu un deplasament

7: $r_t = \beta_2 \cdot r_{t-1} + (1 - \beta_2) \cdot g_t^2$ actualizarea momentului doi, cu un deplasament

8: $s_t' = s_t/(1 - \beta_1^t)$ corectarea deplasamantului pentru primul moment

9: $r_t' = r_t/(1 - \beta_2^t)$ corectarea deplasamentului pentru momentul doi

10: $\theta_t = \theta_{t-1} - \eta \cdot s_t'/(\sqrt{r_t'} + \delta)$ actualizarea parametrilor

11: **end while**

12: return θ_t

1.4. Rezultate

Pentru clasificarea folosind mașinile vector suport, am folosit implementarea din platforma scikit_learn [54]. O etapă importantă a fost reprezentată de căutarea celor mai bune valori pentru hiperparametrii modelului. A fost folosită metoda de căutare grilă, prin care variam nucleul, parametrul C, și unde era cazul, parametrii γ și d. Valorile determinate pentru acești parametri sunt incluse în Anexa 1.

Am experimentat cu diverse moduri de obținere a caracteristicilor, ținând cont de unigrame, unigrame din care am păstrat doar rădăcina, bigrame, forma din dicționar a cuvintelor ș.a.m.d.

	Caracteristici	Precision	Recall	F1
Laptop	Unigram	43.875%	49.311%	46.434%
	Stem_unigram	44.249%	50.563%	47.196%
	Uni+Stem	45.125%	51.564%	**48.130%**
	Bigram	37.015%	39.424%	38.181%
	Uni+Bigram	43.663%	47.434%	45.470%
	Uni+Stem+Bigram	46.437%	48.936%	47.653%

Tabel 1.3: Rezultatele obținute folosind metoda SVC, variind caracteristicile

În fiecare caz, am calculat cele trei măsuri: precizia, rata de regăsire și măsura F1. La conferința SemEval2016 [45], a fost obținut un clasament în funcție de scorul F1 rezultat, aceasta fiind măsura după care vom compara în continuare rezultatele obținute. Pentru setul de date Laptop, folosind mașinile vector suport, am obținut un scor F1 de 48.130%, ținând cont doar de unigrame și de rădăcina cuvintelor. Acest scor depășește restul metodelor propuse la conferința din anul 2016, folosind acest set de date, la categoria Constrained, unde nu sunt permise folosirea datelor auxiliare pentru rezolvarea problemei. Metoda mașinilor vector suport demonstrează astfel că reprezintă un instrument puternic de clasificare dacă parametrii modelului au fost aleși în mod corespunzător.

La categoria *Constrained*, în 2016, cel mai bun scor F1 înregistrat folosind setul de date Laptop a fost de 47.891%, metoda folosită fiind prezentată în lucrarea Hercig et al. [18]. Aceștia au folosit ca și metodă de clasificare entropia încrucișată și un mod foarte vast de a forma

caracteristici: au experimentat cu forma de dicționar a cuvintelor, au considerat o fereastră de context de 5 cuvinte înainte și după un verb, o fereastră de 5 cuvinte la finalul unei propoziții, au ținut cont de apariția unigramelor/bigramelor în text, prezența semnelor de punctuație, au calculat mai multe scoruri precum TF-IDF pentru fiecare cuvânt din datele de antrenare, și altele.

Cel mai bun scor înregistrat folosind setul de date Laptop a fost de 51.937% cu metoda aparținând lui Toh și Su [56]la categoria *Unconstrained*, unde era permis să se folosească și alte date pentru antrenare sau formarea de caracteristici, cum ar fi vectorii de încorporare. Aceștia au propus o metodă ce folosește rețele neuronale de convoluție, antrenând câte o rețea pentru fiecare aspect. Pentru optimizarea rețelei, au folosit funcția de optimizare Adadelta [62], restul parametrilor precum dimensiunea filtrului sau a neuronilor de pe stratul ascuns fiind determinate printr-o metodă de căutare grilă. Au folosit vectori de încorporare folosind seturile de date conținând recenzii provenind din mai multe domenii de pe Amazon [2] și recenzii ale utilizatorilor preluate de pe Yelp, Yelp Phoenix Academic Dataset [61], și au folosit și o serie de caracteristici provenite din datele de antrenare: au ținut cont de apariția unigramelor/bigramelor, au folosit o listă de expresii țintă care apar frecvent în datele de antrenare, au ținut cont de părțile de vorbire ale cuvintelor ș.a.m.d.

	Precision	Recall	F1
Rezultat	86.50%	65.616%	**74.626%**
NLANGP [56]	75.49%	69.44%	72.34%
NileT [24]	72.69%	73.08%	72.88%
AUEB [60]	67.75%	75.77%	71.54%

Tabel 1.4: Rezultatul obținut folosind rețelele neuronale de convoluție pe setul de date Restaurant, comparativ cu alte rezultate obținute la SemEval2016

Rețelele neuronale de convoluție au fost folosite împreună cu setul de date Restaurant. În acest caz, nu am mai separat entitățile de atribute, și am considerat că un aspect este format din perechea E#A. Rezultatul obținut poate fi observat în tabelul 1.4. Metoda depășește cel mai bun scor înregistrat în anul 2016 pe același set de date, unde aveam măsura F1 înregistrată 73.031%, în lucrarea lui Toh și Su [56]. Aceștia au antrenat modelul în mod similar ca în cazul anterior. Deosebirea dintre cele două implementări este că au fost folosiți vectori de încorporare diferiți

(am folosit GloVe). Nu am folosit alte caracteristici suplimentare. Pentru optimizarea rețelei am folosit algoritmul de optimizare Adam, pe când ei au folosit Adadelta.

Chiar dacă nu am folosit alte caracteristici suplimentare decât cele oferite de vectorii globali GloVe, aceștia s-au dovedit a fi superiori pentru probleme de clasificare față de alți vectori existenți. Metoda de optimizare Adam poate fi un alt factor pentru care am obținut un rezultat mai bun, fiind o metodă recent implementată ce a căpătat popularitate în domeniul învățării automate.

Capitolul II – Identificarea Expresiilor Țintă

În acest capitol, vom descrie o metodă de identificare a expresiilor țintă din cadrul unor recenzii, adică acele cuvinte care identifică un aspect. Am folosit o metodă ce folosește câmpuri aleatorii condiționate, urmând lucrarea lui Xenos et al. [60].

2.1. Descrierea setului de date

Ca și în capitolul precedent, am folosit setul de date oferit de către SemEval în anul 2016. Am folosit doar recenziile despre restaurante, doar acestea fiind adnotate cu expresii țintă, pentru a fi folosite pentru antrenare și testare. Expresiile țintă sunt cuvinte sau șiruri de cuvinte ce se regăsesc în interiorul unei recenzii și identifică entitatea *E* din perechea entitate-atribut *E#A* ce formează un aspect [45]. Pentru fiecare expresie țintă, sunt adnotate și pozițiile la care aceasta se găsește în șirul de caractere, fiind marcat pentru fiecare expresie unde începe și unde se termină în cadrul propoziției, așa cum avem în exemplul următor:

Ex: *We, there were four of us, arrived at noon - the place was empty - and the s**taff** acted like we were imposing on them and they were very rude.*
Opinion target: *staff*; **Aspect**: *SERVICE#GENERAL*; **From**: *75*; **To**: *80.*

Există și situații când, pentru o propoziție, nu există o expresie țintă, ca în următorul exemplu:

Ex: *They never brought us complimentary noodles, ignored repeated requests for sugar, and threw our dishes on the table.*
Opinion target: NULL; **Aspect**: *SERVICE#GENERAL;* **From**: 0; **To**: 0

Cele două propoziții oferite ca exemplu provin din aceeași recenzie. Cea de-a doua propoziție continuă ideea exprimată în prima, în care este vorba despre personalul restaurantului, iar acest fapt se deduce din context. În astfel de cazuri, când expresia țintă se găsește într-o propoziție anterioară, sau se vorbește la general despre un aspect, fără a exista un cuvânt cheie ancorat acesteia, expresia va fi marcată NULL.

		Recenzii	Propoziții	NULL	1 cuvânt	2 cuvinte	>2 cuvinte
Restaurant	Antrenare	350	2000	627	1373	316	191
	Testare	90	676	209	485	106	59

Tabelul 2.1: Date statistice cu referire la setul de date folosit

Avem un număr considerabil de cazuri când un aspect nu are asociată o expresie țintă, atât în datele de antrenare cât și în datele de testare, așa cum se poate observa în Tabelul 2.1. Cele mai multe expresii sunt formate dintr-un singur cuvânt, însă există și cazuri în care expresia e alcătuită dintr-o secvență formată din mai multe cuvinte consecutive.

2.2. Problema Etichetării Secvențiale

Problema identificării expresiilor țintă poate fi considerată o problemă de etichetare secvențială. Fiecare cuvânt din propoziție va fi etichetat conform sistemului IOB (Inside – Outside – Beginning), astfel: un cuvânt ce exprimă începutul expresiei țintă va fi etichetat cu *B*, restul cuvintelor ce aparțin unei expresii se etichetează cu *I*, iar cuvintele care nu fac parte din expresie se etichetează cu *O* [60] . Sistemul de identificare al expresiilor țintă va prezice etichetele cuvintelor din setul de antrenare, bazându-se pe observațiile făcute în momentul antrenării.

2.2.1. Caracteristicile folosite

Există multe posibilități de obținere a vectorului de caracteristici, iar aria de experimentare e largă: se poate varia dimensiunea ferestrei de context pentru fiecare cuvânt, numărul minim de apariții, părțile de vorbire, dimensiunea vectorilor ș.a.m.d.

În cele din urmă, am format caracteristicile urmând metoda lui Xenos et al. [60], ținând cont de următoarele aspecte morfologice: am marcat dacă un cuvânt începe cu majusculă, dacă un cuvânt conține majuscule - altele în afară de prima literă, dacă toate literele sunt majuscule, dacă toate literele sunt minuscule, dacă toate caracterele sunt litere/numere, dacă există semne de punctuație în cuvânt. Pentru fiecare astfel de aspect morfologic, se creează un vector separat de caracteristici unde etichetăm cu 1 cuvintele dintr-o propoziție care întrunesc condiția și 0 altfel.

Am creat diferite fișiere, fiecare reprezentând un lexicon format din diferite caracteristici: un lexicon ce cuprinde toate părțile de vorbire identificate în datele de antrenare, un lexicon cu toate expresiile țintă identificate, trei fișiere lexicon în care se marchează toate prefixele identificate de 1, 2 și 3 litere, și trei fișiere lexicon în care se marchează sufixele de 1, 2 și 3 litere. Pentru fiecare cuvânt din propoziții și pentru fiecare astfel de lexicon format, se formează un vector de zero-uri cu o singură valoare de 1 (one-hot vector) ce indică elementul din lexicon corespunzător cuvântului curent.

Am extins vectorul de caracteristici, experimentând cu diferiți vectori de încorporare. Am testat vectorii globali pentru reprezentarea cuvintelor GloVe, prezentați în capitolul anterior, și am folosit vectori de dimensiune 200 pentru fiecare cuvânt în parte. Am folosit și vectorii de încorporare a informațiilor prezentați în lucrarea lui Xenos et al. [60], ce provin din setul de recenzii de produse Amazon [2]. Setul de date reprezintă o colecție largă de recenzii, obținute de pe site-ul Amazon și colecționate în perioada mai, 1996 – iulie, 2014. Sunt, în total, 142.8 milioane de recenzii, conținând o serie de informații cu privire la autor, numărul de stele, aspecte despre produs, descrierea produsului ș.a.m.d. Pentru obținerea vectorilor de caracteristici, din setul de date Amazon, s-a folosit Gensim[11], un instrument prin care se extrag categorii semantice din documente, într-un mod cât mai eficient și facil pentru utilizator.

Ca și GloVe, algoritmul Gensim are la bază relațiile de co-apariție dintre cuvintele folosite în recenziile on-line. Odată identificate anumite tipare, noi documente pot fi exprimate în mod succint în noua formă semantică, și apoi se caută un tipar pentru identificarea categoriei din care recenzia face parte [11]. Se obține, astfel, un vector de 200 de caracteristici pentru fiecare cuvânt dintr-o propoziție.

Pentru fiecare cuvânt, am considerat o fereastră de context de dimensiune variabilă, luând în considerare cuvintele de la stânga și de la dreapta cuvântului curent. Dacă ne aflăm la marginea propoziției, sau o recenzie este mai scurtă decât lungimea ferestrei de context, atunci folosim cuvinte speciale, incluse în vocabular, special adăugate pentru a marca acest fapt. De exemplu, presupunem că avem un șir de cuvinte și dorim să obținem vectorul de caracteristici pentru cuvintele din fereastra de context de dimensiune 2 a primului cuvânt. La stânga acestuia nu se află nimic, așadar vom considera cuvintele *$start1* și *$start2*, reprezentate în vectorul de încorporare cu valori aleatoare. Similar pentru finalul propoziției, unde vom considera cuvintele *$end1* și *$end2*. În cazul în care un cuvânt nu se găsește inclus în vocabularul vectorilor de

încorporare, urmând implementarea din [60], căutam la stânga, în mod iterativ, un cuvânt până ce acesta este găsit în vocabular. În condițiile în care nu se găsește un astfel de cuvânt, parcurgând în mod iterativ secvența de cuvinte la stânga ajungem la finalul propoziției, caz în care se ia în considerare vectorul de caracteristici corespunzător cuvântului *$start1*. După identificarea tuturor cuvintelor, se vor concatena vectorii de caracteristici și se obține un singur vector reprezentativ pentru fereastra de context. Un cuvânt este caracterizat de un vector de dimensiune 200, deci o fereastră de context de dimensiune 3 va avea dimensiunea 1400, iar o fereastră de dimensiune 2, dimensiunea 1000 a vectorului final. Nu am considerat ferestre de context mai largi deoarece acestea diluează sensul cuvântului țintă.

În final, toate aceste trăsături menționate mai sus sunt concatenate și se formează un singur vector de caracteristici, ce descrie un singur cuvânt.

2.2.2. Câmpuri aleatorii condiționate

Câmpurile aleatorii condiționate (conditional random fields) reprezintă o metodă de rezolvare a problemei etichetării secvențiale. Ele au fost introduce în 2001 în lucrarea lui Lafferty et al. [31], ca o alternativă pentru folosirea modelelor Markov ascunse sau a gramaticilor stohastice, având avantajul că presupunerile de independență folosite în cele două modele sunt mai relaxate. Modelele Markov ascunse și gramaticile stohastice sunt modele generative, calculându-se o probabilitate a unei secvențe de etichete condiționată de observații, iar pentru acest lucru este necesară generarea tuturor posibilităților de observații, problemă ce devine nefezabilă dacă se ține cont de modul în care caracteristicile din observații relaționează între ele.

Un model condițional calculează probabilitățile posibilelor etichetări ale observațiilor, ce sunt oferite la intrarea în algoritm ca date de antrenare sau testare, și nu generate, ca în modelele anterioare [31]. Pot fi folosite caracteristici arbitrare, între care există relații de dependență, fără a fi necesar ca modelul să cunoască distribuția acestor dependențe. Vectorul de caracteristici poate încifra în același timp diferite atribute ale aceluiași cuvânt; putem avea informații despre cuvânt (partea de vorbire, sufixe, prefixe ș.a.m.d.) sau despre caracterele ce alcătuiesc cuvântul (majuscule, minuscule, cifre, litere ș.a.m.d.). Probabilitatea unei secvențe nu va depinde doar de

observația curentă, ci poate fi influențată și de observații anterioare sau posterioare, în urma procesului de antrenare al modelului.

În cele ce urmează vom defini metoda câmpurilor aleatorii condiționate la nivel formal. Notăm cu X variabila aleatoare ce ia valori în mulțimea datelor de etichetat (în cazul nostru, a cuvintelor dintr-o recenzie) și cu Y variabila aleatoare ce ia valori în mulțimea etichetărilor posibile. Fiecare componentă Y_i din Y ia valori din mulțimea etichetelor (I, O sau B conform etichetării IOB descrisă anterior). Modelul obținut se notează $p(Y|X)$ și exprimă probabilitatea unei secvențe de etichete Y condiționată de observația X.

Putem considera secvența de etichete ca fiind noduri ale unui graf $G = (V, E)$ unde mulțimea nodurilor V indexează secvența de etichete $Y = (Y_v)_{v \in V}$, iar E reprezintă mulțimea de muchii a grafului. O pereche (X, Y) poartă numele de câmp aleator condiționat și respectă proprietatea Markov

$$p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, (wv) \in E) \qquad (2.1)$$

Putem considera graful G ca fiind un lanț cu $V = \{1, 2, \dots, m\}$ noduri și $E = \{(i, i+1)\}$ muchii, cu $i = 1, \dots, m-1$. Acest graf poate fi considerat ca fiind un arbore în care oricare două noduri adiacente formează o clică, adică o submulțime de noduri cu proprietatea că oricare două noduri distincte din clică sunt adiacente. Atunci, conform Teoremei Fundamentale a Câmpurilor Aleatorii (Teorema Hammersley – Clifford) [31], probabilitatea condiționată $p(Y|X)$ ia forma:

$$p_\theta(Y|X) = \exp\left(\sum_{e \in E, k} \lambda_k f_k(e, Y|_e, X) + \sum_{v \in V, k} \mu_k g_k(v, Y|_v, X)\right). \quad (2.2)$$

În formula (2.2), $Y|_S$ reprezintă acele componente din Y asociate subgrafului format din componentele S. Funcțiile f_k și g_k calculează caracteristicile obținute din observații. În cazul nostru, ele au fost calculate anterior (capitolul 2.2.1). Parametrii $\theta = (\lambda_1, \lambda_2, \dots; \mu_1, \mu_2, \dots)$ vor trebui estimați din datele de antrenare $D = \left\{\left(X^{(i)}, Y^{(i)}\right)\right\}$, cu $i = 1, \dots, N$, cu $p'(X, Y)$ o distribuție empirică a datelor și N numărul total al datelor de antrenare.

Putem scrie probabilitatea condiționată în formă matriceală. Pentru fiecare element i din observația X, definim o matrice pătratică de dimensiunea alfabetului etichetelor $M_i(X) = [M_i(Y', Y|X)]$, unde

$$M_i(Y', Y|X) = \sum_k \lambda_k f_k\left(e_i, Y|_{e_i} = (Y', Y), X\right) + \sum_k \mu_k g_k\left(v_i, Y|_{v_i} = Y, X\right). \quad (2.3)$$

Muchia e_i corespunde muchiei $(Y_{i-1}, Y_i) \epsilon\, E$ iar v_i corespunde nodului $Y_i \epsilon V$. Introducem două noduri $Y_0 = start$ și $Y_{N+1} = stop$ pentru a simplifica expresia. Atunci, conform [31], funcția de partiție Z_θ se obține prin înmulțirea matricelor

$$Z_\theta(\mathrm{X}) = (M_1(X)M_2(X) \ldots M_{N+1}(X), \quad (2.4)$$

iar ecuația (2.2) devine

$$p_\theta(Y|X) = \frac{\prod_{i=1}^{N+1} M_i(Y_{i-1}, Y_i|\, X)}{Z_\theta(\mathrm{X})} \quad (2.5)$$

Pentru estimarea parametrilor θ, se dorește maximizarea funcției obiectiv

$$O(\theta) = \sum_{x,y} p'(X, Y) \log p_\theta(Y|X). \quad (2.6)$$

Pentru aceasta, se folosește un algoritm bazat pe metoda scalării iterative îmbunătățite (improved iterative scaling IIS), introdusă în lucrarea lui Della Pietra et al. [9]. Ponderile se modifică sub forma $\lambda_k = \lambda_k + \delta\lambda_k$ și $\mu_k = \mu_k + \delta\mu_k$, unde $\delta\lambda_k$ și respectiv $\delta\mu_k$ reprezintă variabile în formulele:

$$E'[f_k] = \sum_{x,y} p'(X)p(X, Y) \sum_{i=1}^{N+1} f_k\left(e_i, Y|_{e_i}, X\right) e^{\,\delta\lambda_k T(X,Y)} \quad (2.7)$$

$$V'[g_k] = \sum_{x,y} p'(X)p(X, Y) \sum_{i=1}^{N+1} g_k\left(v_i, Y|_{v_i}, X\right) e^{\,\delta\mu_k T(X,Y)} \quad (2.8)$$

unde $T(X, Y)$ reprezintă numărul total de caracteristici:

$$T(X, Y) = \sum_{i,k} f_k(\, e_i, Y|_{e_i}, X) + \sum_{i,k} g_k(\, v_i, Y|_{v_i}, X) \quad (2.9)$$

Pentru a simplifica operațiile din formulele (2.7) și (2.8), conform lui Lafferty et al. [31], se definește un nou vector de caracteristici

$$s(X, Y) = S - \sum_i \sum_k f_k\left(e_i, Y|_{e_i}, X\right) - \sum_i \sum_k g_k\left(v_i, Y|_{v_i}, X\right) \quad (2.10)$$

unde S reprezintă o constantă astfel aleasă pentru ca $s\left(X^{(i)}, Y\right) \geq 0, \forall X^{(i)}, Y$ date de antrenare, respectiv etichetări ale acestora. Astfel, $T(X, Y) = S$.

Se introduce vectorul $\alpha_i(X)$ pentru care $\alpha_0(Y|X) = 1$ dacă $Y = start$ și 0 altfel; pentru restul elementelor, $\alpha_i(X) = \alpha_{i-1}(X)M_i(X)$, pentru $i = 0, \ldots, N + 1$. Acesta va purta numele de

vector „înainte”. Vom defini și un vector „înapoi” $\beta_i(X)$ unde $\beta_{N+1}(Y|X) = 1$ $dacă$ $Y = stop$ și 0 altfel, iar restul elementelor $\beta_i(X)^T = M_{i+1}(X)\beta_{i+1}(X)$.

Ecuațiile parametrilor de actualizare:

$$\delta\lambda_k = \frac{1}{S} log \frac{E'[f_k]}{E[f_k]}, \delta\mu_k = \frac{1}{S} log \frac{E'[g_k]}{E[g_k]} \quad (2.11)$$

$$E[f_k] = \sum_X p'(X) \sum_{i=1}^{N+1} \sum_Y f_k\left(e_i, Y|_{e_i} = (Y', Y), X\right) \times \frac{\alpha_{i-1}(Y'|X) M_i(Y', Y|X) \beta_i(Y|X)}{Z_\theta(X)} \quad (2.12)$$

$$E[g_k] = \sum_X p'(X) \sum_{i=1}^{N} \sum_Y g_k\left(v_i, Y|_{v_i} = Y, X\right) \times \frac{\alpha_i(Y|X)\beta_i(Y|X)}{Z_\theta(X)} \quad (2.13)$$

2.3. Rezultate

Metoda câmpurilor aleatorii condiționate a fost implementată cu ajutorul platformei pystruct [48][49]. Pentru această subproblemă, am calculat precizia, rata de regăsire și măsura F1, cea din urmă fiind și măsura după care s-a alcătuit clasamentul soluțiilor trimise la convenția SemEval 2016 [45].

Am aplicat această metodă folosind doar setul de date Restaurant, deoarece numai acesta conține adnotările expresiilor țintă. Am efectuat mai multe experimente, în care variam caracteristicile folosite. Rezultatele pot fi analizate în Tabelul 2.2.

Folosind doar caracteristici obținute din setul de date oferit de SemEval, fără a folosi vectori de încorporare, obținem scorul 56.844%. Caracteristicile folosite sunt următoarele: s-a ținut cont de existența prefixelor și sufixelor de 1, 2 respectiv 3 caractere, s-a ținut cont de partea de vorbire a fiecărui cuvânt din fereastra de context, am considerat caracteristici morfologice ale cuvintelor (litere mari, mici, numere pe post de caractere, semne de punctuație etc.), și am ținut cont și de expresiile frecvent găsite, care identifică un aspect.

Apoi am extins experimentul, adăugând vectorii de încorporare. Folosind doar GloVe cu fereastra de context de dimensiune 2, fără a include nici o altă informație despre propozițiile testate, am obținut măsura F1 54.921%, rezultat foarte apropiat de primul experiment când am inclus caracteristicile despre date. Cu vectorii globali ai reprezentării cuvintelor, cel mai bun rezultat a fost obținut atunci când am combinat vectorii de încorporare cu trăsăturile cuvintelor ce oferă informații despre sufixe/prefixe și expresiile care identifică un aspect.

Cu vectorii proveniți din setul de date Amazon am obținut îmbunătățiri substanțiale – folosind doar acești vectori, fără altă informație adițională, cea mai bună măsura F1 obținută în final a fost de 60.003%. Am efectuat apoi o serie de alte experimente. Cel mai bun rezultat a fost obținut combinând acești vectori de încorporare, cu informații despre sufixe/prefixe și expresiile frecvent întâlnite.

Am experimentat și cu dimensiunea ferestrelor de context și am obținut cel mai bun rezultat atunci când dimensiunea ferestrei era 3 (trei cuvinte la stânga și trei cuvinte la dreapta cuvântului țintă). Pentru restul experimentelor, am folosit dimensiunea 3 a ferestrei de context.

	Precision	Recall	F1
GloVe	57.433%	52.619%	54.921%
Toate	60.129%	53.899%	56.844%
GloVe+Amazon	59.62%	54.831%	57.125%
Amazon_context_1	0.6	55.878%	57.866%
Amazon_context_2	61.85%	56.81%	59.223%
Amazon_context_3	63.506%	56.926%	60.003%
GloVe+Suf_Pre+Freq	63.144%	58.440%	60.701%
Amazon+POS	64.03%	58.44%	61.107%
Amazon+Suf_Pre+Morf.	64.491%	59.837%	62.077%
Amazon+Suf_Pre	64.779%	59.953%	62.273%
Amazon+Toate	65.404%	60.302%	62.749%
Amazon+Suf_Pre+Freq+POS	65.652%	60.302%	62.864%
Amazon+Suf_Pre+Freq	65.13%	60.884%	62.936%

Tabelul 2.2:Rezultatele obținute aplicând metoda câmpurilor aleatorii condiționate pe setul de date Restaurant

Cea mai bună măsură pentru categoria *Constrained* a fost 66.9% înregistrată în lucrarea lui Hercig et al. [18], implementând tot o metodă de câmpuri aleatorii condiționate. Diferența constă în modul de alegere al caracteristicilor, aceștia experimentând cu diverse moduri de a forma caracteristici: au considerat sufixe ale cuvintelor ce apar de cel puțin 5 ori în datele de

antrenare, apariția unigramelor/bigramelor în text, s-au ținut cont de părțile de vorbire ale cuvintelor, ș.a.m.d.

La categoria *Unconstrained*, cel mai bun rezultat a fost de 72.34% cu metoda propusă de Toh și Su [56]. Ei au folosit tot o metodă de câmpuri aleatorii condiționate, însă rezultatul a fost îmbunătățit folosind o rețea neuronală recurentă care ține cont de relațiile de dependență dintre cuvinte. Ca și vector de încorporare, au folosit un set de date Google conținând 100 miliarde de cuvinte provenite din știri.

Metoda câmpurilor aleatorii condiționate dovedește astfel că este sensibilă la modul de alegere al caracteristicilor.

Capitolul III – Determinarea polarității

În acest capitol, tema este reprezentată de determinarea polarității recenziilor. A fost abordată o metodă de clasificare ce folosește regresia logistică, urmând lucrarea lui Pang et al. [40] la nivelul pre-procesării textului și formării de caracteristici. Am implementat o metodă de fragmentare a recenziilor care izolează un aspect. Adică, într-o recenzie adnotată cu mai multe aspecte, propoziția va fi împărțită în alte fragmente de text mai mici, în care se vorbește numai despre un singur aspect, urmând mai apoi a se efectua determinarea polarității pe aceste fragmente în mod individual.

3.1. Descrierea datelor

Am folosit seturile de date oferite de către SemEval în anul 2016 [45], ce conțin recenzii despre restaurante și laptop-uri. Fiecare propoziție este adnotată cu aspectele și polaritățile asociate. Polaritatea exprimă sentimentul transmis de către un utilizator care face referire la un aspect anume ce ține de obiectul descris. Părerea utilizatorilor poate fi pozitivă, negativă sau neutră, acestea trei fiind etichetele cu care sunt adnotate fiecare aspect în parte.

	Set date	Recenzii	Propoziții	Pozitive	Negative	Neutre	Null
Laptop	Antrenare	450	2500	1637	1084	188	461
	Testare	80	808	481	274	46	235
Restaurant	Antrenare	350	2000	1657	749	101	292
	Testare	90	676	611	204	44	89

Tabel 3.1: Date statistice cu referire la seturile de date: numărul de recenzii, numărul de propoziții, numărul de aspecte pozitive/negative/neutre, propoziții ne-etichetate

Așa cum se poate observa și în Tabelul 3.1, cele mai multe aspecte identificate în recenzii au asociate o părere pozitivă, urmate de cele cu părere negativă. Clasa propozițiilor neutre este aceea în care părerea este undeva între, adică nu se vorbește nici de bine dar nici de rău. Există și propoziții în care nu este exprimat niciun aspect, și deci nu sunt adnotate cu o polaritate. Ele există pentru a întregi contextul unei recenzii.

	Set de date	Nr.Pol = 1	Nr.Pol = 2	Nr.Pol > 2	Pol. diferite
Laptop	Antrenare	1352	534	153	151
	Testare	394	143	36	35
Restaurant	Antrenare	1151	381	176	143
	Testare	410	128	49	42

Tabel 3.2: Numărul de propoziții din seturile de date dare conțin una, două sau mai multe păreri adnotate și numărul de cazuri în care sunt exprimate păreri diferite

Problema pe care o ridică determinarea polarității în analiza sentimentelor bazată pe aspecte este aceea că în interiorul unei propoziții pot fi exprimate mai multe păreri. După cum se poate vedea și în Tabelul 3.2, avem un număr considerabil de propoziții în care sunt exprimate două sau mai multe păreri despre diferite entități (în unele cazuri există mai multe păreri exprimate despre aceeași entitate). Mai mult, există și cazuri când părerile pot să difere, ca în exemplul următor:

Ex: *The **food** was average to above-average; the **French Onion soup** filling yet not overly impressive, and the **desserts** not brilliant in any way.*
Aspect: **food**; Sentiment: *positive*
Aspect: **French Onion soup**; Sentiment: *positive*
Aspect: **French Onion soup**; Sentiment: *negative*
Aspect: **desserts**; Sentiment: *negative*

Se poate observa cum o singură recenzie poate să conțină păreri despre mai multe aspecte. Numărul de păreri exprimate nu coincide cu numărul de aspecte despre care se vorbește în recenzie, după cum este evidențiat în cazul aspectului *French Onion Soup*, pentru care există asociate o părere pozitivă și una negativă.

3.2. Separarea aspectelor

Propun o metodă euristică de separare a contextelor dintr-o propoziție, pentru a facilita determinarea polarității. Metoda nu garantează întotdeauna o delimitare precisă a contextelor,

însă rezultatele obținute în mod experimental pe setul de date au dovedit că în majoritatea cazurilor, delimitarea obținută coincide cu intuiția umană de delimitare. Numărul de cuvinte relativ scăzut care alcătuiesc propozițiile reprezintă un factor pentru care metoda se poate aplica, acestea nefiind complicate din punct de vedere semantic.

Este important pentru determinarea polarității unui aspect să se identifice expresia țintă care îl reprezintă și să se folosească contextul din jurul acesteia. În general, din punct de vedere al discursului și al structurii recenziilor, ideile sunt prezentate în mod secvențial, adică se prezintă o idee/părere/opinie despre un aspect, apoi se trece la alta, fără a mai reveni asupra punctului de vedere exprimat anterior, ca în exemplele următoare unde cuvintele evidențiate reprezintă expresia țintă la care se face referire: „*Nice **ambience**, but highly overrated **place**.” , „It has great **sushi** and even better **service**.”, „The **wine list** is incredible and extensive and diverse, the **food** is all incredible and the **staff** was all very nice, good at their jobs and cultured.”* etc.

Cel mai adesea, aceste contexte se pot delimita prin intermediul unor separatori: semne de punctuație sau anumite cuvinte (*and, so, but, though, while,* etc.). Am implementat o metodă care caută acești separatori și delimitează contextul unui aspect de restul propoziției, urmând apoi a efectua metode de determinare a polarității pe aceste secvențe de cuvinte noi obținute.

Există două moduri prin care un aspect poate fi exprimat într-o propoziție, așa cum am văzut și în capitolul 2: fie direct, prin identificarea unei expresii țintă, fie indirect, în cazul în care nu avem o expresie țintă, aceasta existând doar la nivel de sugestie. Cel mai ușor este să extragem un context pentru care există un cuvânt țintă, și de aceea, acestea vor fi primele extrase.

Se presupune că pentru această etapă a analizei sentimentelor bazată pe aspecte, se cunosc atât aspectele, pe care le putem obține prin metodele descrise în capitolul 1, cât și expresiile țintă, identificate în capitolul 2.

Notăm $sep = \{s_1, s_2, \dots, s_k\}$ lista predefinită de k separatori și $x = [x_1, x_2, \dots, x_n]$ propoziția exprimată sub forma unui șir de cuvinte pe care dorim să o prelucrăm, unde n reprezintă numărul de cuvinte. Presupunem că propoziția este adnotată cu m aspecte, pentru care există sau nu expresii țintă reprezentative. Un context reprezintă o secvență succesivă de cuvinte $x' = [x_i, x_{i+1}, \dots, x_j]$ provenind din x. Eliminând pe x' din x, propoziția se actualizează $x = [x_1, x_2, \dots, x_{i+1}, x_{j+1}, \dots, x_n]$ iar numărul de aspecte devine $m = m - 1$. Procesul de eliminare se repetă până când $m = 1$.

Presupunem că avem un cuvânt țintă x_{T1}, iar $m > 2$. Căutăm la stânga și la dreapta cuvântului separatori pentru a putea delimita contextul. Dacă, parcurgând propoziția la stânga nu găsim niciun separator și ajungem în capătul din stânga, vom considera că acesta reprezintă începutul contextului. Analog pentru parcurgerea la dreapta. Obținem contextul $x' = [x_i, x_{i+1}, \dots, x_{T1}, \dots, x_j]$ unde $1 \leq i < T1 < j \leq n$ și $x_i, x_j \in sep$.

Ex: $x =$ *I like the* ***somosas****,* ***chai****, and the* ***chole****, but the* ***dhosas*** *and* ***dhal*** *were kinda dissapointing.*

x_{T1}: *somosas*

Contextul lui x_{T1}: *I like the* ***somosas****,*

Actualizare: $x =$ ***chai****, and the* ***chole****, but the* ***dhosas*** *and* ***dhal*** *were kinda dissapointing.*

În acest exemplu, primul cuvânt țintă identificat este *somosas*. Căutând un separator la stânga cuvântului, epuizăm cuvintele fără a fi găsit unul. În acest caz, marginea din stânga a contextului va fi începutul propoziției. Căutând un separator la dreapta, găsim prima virgulă (faptul că virgula este un separator este prestabilit de la începutul rulării algoritmului), și deci aceasta va reprezenta marginea din dreapta a contextului cuvântului x_{T1}. Separăm contextul identificat de restul propoziției, iar procesul se repetă pentru cuvintele din propoziția rămasă.

În cazul în care $m = 2$, și există cuvintele țintă x_{T1} *și* x_{T2}, vom căuta un separator între cele două poziții și obținem contextul $x' = [x_1, x_2, \dots, x_{T1}, \dots, x_j]$ unde $T1 < j < T2$ și $x_j \in sep$.

Ex: *x = Best* ***Pastrami*** *I ever had and great* ***portion*** *without being ridiculous.*

x_{T1}: *Pastrami*; x_{T2}: *portion*

Contextul lui x_{T1}: *Best* ***Pastrami*** *I ever had and*

Actualizare:*x = great* ***portion*** *without being ridiculous.*

În această situație, vom căuta primul separator dintre cele două cuvinte țintă. Contextul va fi marcat de începutul propoziției și separatorul identificat. În urma actualizării, secvența de cuvinte x conține doar un singur aspect, și deci algoritmul ia sfârșit, deoarece nu avem alte aspecte de separat.

În cazul în care $m = 2$, și există cuvânt țintă x_{T1} dar al doilea aspect este exprimat în mod indirect, parcurgem propoziția la stânga din poziția $T1$. Dacă ajungem în capătul din stânga

fără a fi găsit un separator pe parcurs, știm sigur că acesta se află în partea dreaptă, și îl vom identifica. Analog pentru parcurgerea la dreapta a propoziției. Contextul în acest caz va fi $x' = [x_1, x_2, \dots, x_{T1}, \dots, x_i]$, unde $1 \leq T1 < i$ și $x_i \in sep$ sau $x' = [x_i, x_{i+1}, \dots, x_{T1}, \dots, x_n]$, unde $i < T1 \leq n$ și $x_i \in sep$.

Ex: *x = The **food** here is rather good, but only if you like to wait for it.*

x_{T1}: *food*

Contextul lui x_{T1}: *The **food** here is rather good,*

În această propoziție avem exprimate două opinii, una făcând referire la aspectul FOOD#QUALITY, identificată prin expresia țintă *food*, și a doua făcând referire la aspectul SERVICE#GENERAL, fără a avea o expresie țintă asociată. Separăm contextul aspectului pentru care există cuvântul țintă, uitându-ne la stânga și la dreapta acestuia pentru a identifica separatorii.

Acestea sunt toate cazurile în care extragem contextele, atunci când avem o țintă exprimată în mod direct. Mai rămân cazurile când țintele sunt exprimate indirect, fără a mai avea o poziție T în jurul căreia să ne uităm. În această situație, vom separa propoziția doar căutând separatori, pornind din capătul din stânga (sau dreapta) al propoziției. Pentru $m \geq 2$, contextul este de forma $x' = [x_1, \dots, x_j]$, cu $1 < j < n$ și $x_j \in sep$, sau $x' = [x_j, \dots, x_n]$ cu $j < n$.

Ex: *x = It is terrific, as is the value.*

Contextul: $x' =$ *It is terrific,*

În acest exemplu, avem două aspecte asociate acestei recenzii, ambele exprimate în mod indirect: FOOD#QUALITY și FOOD#PRICES. Neavând un cuvânt țintă în jurul căruia să delimităm contextul, vom considera că marginea din stânga a contextului va fi începutul propoziției, iar în partea dreaptă contextul va fi delimitat de primul separator identificat, în acest caz fiind virgula.

3.3. Metoda 1 – Regresie Logistică Multinomială

Regresia logistică multinomială reprezintă o metodă de clasificare supervizată care generalizează regresia logistică pentru problemele în care avem mai multe clase [16], și

reprezintă un model folosit pentru a prezice probabilitățile posibilităților de clasificare, dat fiind un set de variabile independente.

Vom defini regresia logistică din punct de vedere formal, conform Krishnapuram et al. [28]. Presupunem că avem n exemple de antrenare ce trebuiesc clasificate în m clase, pornind de la un vector de caracteristici sau observații x obținut din datele de intrare. Cele m pot fi codificate prin metoda „una din m", adică $y = \left[y^{(1)}, y^{(2)}, \dots, y^{(m)}\right]$, unde $y^{(i)}$ ia valoarea 1 dacă vectorul de caracteristici x corespunde unui exemplu aparținând clasei i și 0 altfel. Având aceste notații, datele de antrenare pot fi scrie sub forma $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$.

Regresia logistică multinomială folosește, ca și alte metode de regresie liniară, o funcție care va calcula probabilitatea ca observația i să aparțină clasei k și poate fi scrisă sub forma

$$f(k, i) = w_{0,k} + w_{1,k}x_{1,i} + \cdots + w_{n,k}x_{n,i} \quad (3.1)$$

iar w reprezintă parametrii ce vor fi obținuți în urma procesului de antrenare. Ne putem imagina regresia logistică multinomială ca efectuând regresie logistică de un număr de $m - 1$ ori, de fiecare dată separând o clasă de restul, iar restul claselor rămase vor forma o clasă mai mare între ele.

Folosind regresia logistică multinomială, probabilitatea ca x să aparțină clasei i poate fi scrisă sub forma

$$P\left(y^{(i)} = 1 \middle| x, w\right) = \frac{\exp(w^{(i)^T} x)}{\sum_{j=1}^{m} \exp(w^{j^T} x)}, \quad (3.2)$$

pentru $i = 1, \dots, m$, unde $w^{(i)}$ reprezintă vectorul de ponderi corespunzător clasei i, ce trebuie determinat în urma procesului de antrenare [28]. Pentru a obține estimarea maximă a probabilității vectorilor w, se maximizează funcția log-probabilitate (log-likelihood):

$$l(w) = \sum_{j=1}^{n} \log P\left(y_j \middle| x_j, w\right) = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} y_j^{(i)} w^{(i)^T} x_j - \log \sum_{i=1}^{m} \exp(w^{(i)^T} x_j) \right). \quad (3.3)$$

Pentru maximizarea formulei (3.3), se folosește cel mai adesea algoritmul celor mai mici pătrate calculate iterativ IRLS (iteratively reweighted least squares), rezultatul depășind din punct de vedere al performanței alte metode, conform Minka[37]. Când datele sunt liniar separabile, funcția $l(w)$ poate lua valori foarte mari, și de aceea este importantă alegerea unei estimări a-priori bune a lui w.

3.4. Rezultate

Pentru obținerea modelului regresiei logistice multinomiale, am folosit implementarea din platforma scikit_learn [35], folosind algoritmul lui Newton pentru problema de optimizare.

Am format datele de antrenare $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, cu y_i luând valori din mulțimea [0, 1, 2] corespunzătoare polarităților neutru, pozitiv respectiv negativ. Pentru antrenare, în cazurile când sunt etichetate mai multe aspecte, toate de aceeași polaritate, nu am mai efectuat separarea contextelor. Metoda s-a aplicat doar atunci când existau aspecte cu polarități diferit adnotate.

Pentru formarea caracteristicilor din datele de antrenare/testare, am experimentat mai multe metode descrise în articolul lui Pang et al. [40] [41]. Am împărțit datele în n-grame (uni/bi/trigrame) ținând cont de numărul de apariții a fiecărei n-gramă. Am încercat și o metodă în care țineam cont doar de prezența unigramelor, nu și de câte ori apare în cadrul unei propoziții. Nu există o diferență majoră între metode; în cel de-al doilea caz reușește să prezică corect un exemplu în plus față de prima metodă, lucru ce este firesc deoarece, spre deosebire de [40], lucrăm cu seturi de date de dimensiuni mai mici.

Am marcat cuvintele de după o expresie negativă până la primul semn de punctuație sau o a doua expresie negativă cu un sufix *_NOT* pentru a evita situații ca în următorul exemplu, unde cuvintele evidențiate, care în mod normal se găsesc în contexte pozitive, ar putea duce la o clasificare eronată a propoziției: The fish was not **great**, and we did not **like** the soup. Metoda nu a prezentat îmbunătățiri pe seturile noastre de date, crescând artificial numărul de caracteristici.

	Caracteristici	Nr. Caract.	Acuratețe
Restaurant	Unigrame Prezență	3501	**81.606%**
	Unigram Frecvență	3501	81.490%
	Bigrame	16606	79.511%
	Trigrame	31.424	79.161%
	Uni+NOT	3921	80.675%
	Bi+NOT	16773	80.558%
	Tri+NOT	32340	79.394%

Tabel 3.3: Rezultatele obținute pe setul de date Restaurant, cu metoda Regresiei Logistice Multinomiale

Ca și la convenția SemEval 2016[45], am folosit acuratețea ca măsură a determinării performanței modelului. Cele mai bun rezultat aplicând metoda regresiei logistice multinomiale s-a obținut folosind unigramele de prezență. Comparativ cu rezultatele înregistrate în anul 2016, metoda se clasează, ca scor, pe locul 3 dintr-un total de 12.

Au fost mai multe metode de implementare propuse folosind regresia logistică (Hercig et al. [18], Xenos et al. [60], însă aceștia au experimentat cu diferite metode de obținere a vectorilor de caracteristici: au considerat un context de 5 cuvinte înainte și după un verb, au considerat un context de 5 cuvinte la finalul, respectiv începutul unei propoziții, au ținut cont de părțile de vorbire ale cuvintelor, apariția unigramelor/bigramelor și a n-gramelor de caractere în interiorul unui cuvânt, fiind considerate doar acelea care apar de cel puțin 2 ori, și altele. În plus, în [60] a fost folosit și un lexicon al sentimentelor precompilat în care fiecare cuvânt avea asociat un scor, pozitiv sau negativ; aceste scoruri indicau cât de bine este exprimat un sentiment de către cuvintele respective. Cu aceste implementări, au obținut actratețea 81.83% la categoria *Constrained*, respectiv 83.23% la categoria *Unconstrained*.

Alte metode au folosit cu succes rețelele neuronale de convoluție (Khalil et al. [24]), în care au fost încorporate cunoștințe adiționale provenite din seturile de date cu recenzii Yelp și Amazon, folosite de alte implementări și în capitolele anterioare iar acuratețea obținută de aceștia a fost de 85.44%. O metodă folosind grafuri de dependență între domenii a fost propusă de Kumar et al. [4], calculând măsuri TF-IDF ale cuvintelor pentru fiecare aspect în parte și formând grafuri de dependență între cuvintele cu scorul cel mai mare. Această soluție a oferit o acuratețe bună, de 86.72%, fiind cea mai mare acuratețe înregistrată la categoria *Unconstrained* în anul 2016.

Concluzii

În concluzie, am implementat în limbajul de programare python metode pentru rezolvarea subproblemelor ce alcătuiesc analiza sentimentelor bazată pe aspecte, având ca punct de referință rezultatele obținute la convenția SemEval din 2016 și folosind seturile de date oferite în cadrul acesteia.

În primul capitol, am abordat problema identificării aspectelor folosind două metode diferite: mașini vector suport și rețele neuronale de convoluție. Am evidențiat faptul că SVM reprezintă o metodă competitivă de clasificare, folosind câte un model specializat pe identificarea fiecărui aspect existent. Dificultatea implementării acestei metode a constat în alegerea parametrilor, pentru care am folosit o metodă de căutare grilă. Pentru fiecare mod de obținere a vectorilor de trăsături, parametrii mașinilor trebuiau recalculați, iar acest lucru a descurajat o amplă experimentare, însă, folosind setul de date Laptop, am obținut cea mai mare măsură F1 comparativ cu restul soluțiilor trimise în anul 2016 în cadrul convenției. Rețelele neuronale de convoluție au fost mai simplu de folosit, acestea fiind implementate cu ajutorul platformei Keras. Ca și în cazul anterior, am specializat o rețea pentru identificarea unui singur aspect. În plus, am folosit vectorii de încorporare GloVe, pre-calculați, conținând informații cu privire la co-aparițiile cuvintelor în diferite contexte. Rezultatele au fost din nou foarte bune, obținând din nou cea mai mare măsură F1 folosind setul de date Restaurant. Vectorii de încorporare GloVe s-au dovedit a fi o alegere superioară față de alți astfel de vectori pre-calculați, cum ar fi setul de date cu recenzii provenite de la Amazon și Yelp, folosite în alte lucrări (Toh și Su [56]).

În capitolul doi, am rezolvat problema identificării expresiilor țintă prin metoda câmpurilor aleatorii condiționate. Problema s-a transformat în una de etichetare a cuvintelor prin intermediul sistemului IOB – în urma antrenării, un clasificator asociază eticheta I, O sau B unui cuvânt din cadrul unei propoziții. Am experimentat cu diverse moduri de a selecta trăsături pentru antrenarea modelului. Am folosit și vectori de încorporare pentru a adăuga informații adiționale, GloVe, amintiți anterior, și vectori provenind din recenzii obținute de pe site-ul Amazon. Am arătat că există numeroase moduri de a alege caracteristici, iar spațiul de experimentare nu este nici pe departe epuizat: de la modificarea dimensiunii ferestrei de context și până la trăsături la nivel de caracter.

În cel de-al treilea capitol, am aplicat metode clasice de efectuare a analizei sentimentelor, urmând metoda prezentată în lucrarea lui Pang și Lee [41] și implementând clasificarea cu ajutorul regresiei logistice multinomiale. Nou în această secțiune este modul în care recenziile au fost prelucrate, pentru a izola un aspect de restul propoziției, și efectuarea analizei sentimentelor pe aceste fragmente de text (în [41], se efectuează analiza sentimentelor la nivelul întregii recenzii). Am aplicat o metodă euristică de delimitare a contextelor, folosind o listă predefinită de separatori (cuvinte sau semne de punctuație) ce indică marginile contextelor.

Pe viitor, lucrarea poate fi reluată, experimentând cu diverse alte metode de clasificare. Pentru a îmbunătăți metoda SVM prezentată în primul capitol, se poate folosi o altă metodă de selectare a parametrilor, cum ar fi metoda prezentată de Cherkassky și Ma [5] prin care se calculează parametrul *C* pornind de la formulele matematice ale acestuia din cadrul teoriei mașinilor vector suport, aplicate pe setul de date oferite pentru antrenare. În cazul identificării expresiilor țintă, se poate experimenta cu diverse alte moduri de a obține caracteristici si se pot folosi diverși alți vectori de încorporare pentru îmbunătățirea rezultatelor. Nu în ultimul rând, în cazul analizei sentimentelor, putem încerca o metodă de a adăuga ponderi separatorilor, favorizându-i pe unii în locul altora – astfel, nu ne-am mai opri la primul identificator identificat, ci la cel mai bun care separă contextul. Pentru formarea unui astfel de set de date în care separatorii au asociate ponderi, este nevoie de ajutorul lingviștilor pentru un mod cât mai corect de acordare a valorilor. Pentru a evita separarea contextelor, se poate experimenta și cu rețelele neuronale de convoluție (Khalil et al. [24]) sau alte moduri de a obține trăsături.

Anexe

Anexa 1: Tabel cu valorile hiperparametrilor pentru modelul SVC

Aspect	Nucleu	C	γ	d
laptop	rbf	5.7	0.025	-
battery	sigmoid	24.627	0.0409	-
Cpu	sigmoid	64	0.0441	-
Graphics	Sigmoid	128	0.031	-
Hard Disc	rbf	64	0.007	-
OS	Poly	0.0002	4	3
Support	Rbf	38.054	0.011	-
Company	Rbf	26.908	0.022	-
Display	Rbf	2.378	0.022	-
Mouse	Rbf	5.656	0.007	-
Software	Sigmoid	6.727	0.0371	-
Keyboard	Poly	0.0011	1.1892	3
Optical Drives	Sigmoid	1024	0.0525	-
Warranty	Sigmoid	107.637	0.074	-
Multimedia	Sigmoid	304.437	0.0220	-
Ports	Linear	1.1892	-	-
Power Supply	Sigmoid	107.634	0.0185	-
Hardware	Sigmoid	32	0.0371	-
Shipping	Linear	1	-	-
Memory	Sigmoid	53.817	0.0312	-
Motherboard	Linear	0.594	-	-
Fans Cooling	Sigmoid	13.454	0.0312	-
General	Rbf	3.363	0.0625	-
Operation Perf.	Rbf	4	0.125	-
Design Features	Rbf	2	0.105	-
Usability	Rbf	13.454	0.018	-
Portability	Rbf	2.378	0.105	-
Price	Sigmoid	6.727	0.0371	-

Quality	Sigmoid	4	0.026	-
Miscellaneous	Rbf	1.414	0.105	-
Connectivity	Sigmoid	22.627	0.0525	-

Referințe bibliografice

[1]Abadi, Martín, Agarwal, Ashish; Barham, Paul; Brevdo, Eugene; Chen, Zhifeng; Citro, Craig; Corrado, Greg S.; David, Andy; Dean, Jeffrey; Devin, Matthieu; Ghemawat, Sanjay; Goodfellow, Ian; Harp, Andrew; Irving, Geoffrey, Isard, Michael, Jia, Yangqing; Jozefowicz, Rafal; Kaiser, Lukasz; Kudlur, Manjunath; Levenberg, Josh; Mane, Dan; Monga, Rajat; Moore, Sherry; Murray, Derek; Olah, Chris; Schuster, Mike; Shlens, Jonathon; Steiner, Benoit; Sutskever, Ilya; Talwar, Kunal; Tucker, Paul; Vanhoucke, Vincent; Vasudevan, Vijay; Viegas, Fernanda; Vinyals, Oriol; Warden, Pete; Wattenberg, Martin; Wicke, Martin; Yu, Yuan; Zheng, Xiaoqiang: *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015), http://download.tensorflow.org/paper/whitepaper2015.pdf

[2]Amazon Dataset, http://jmcauley.ucsd.edu/data/amazon/

[3]Boughorbel, Sabri; Tarel, Jean-Philippe; Boujemaa, Nozha: *Conditionally Positive Definite Kernels For SVM Based Image Recognition* (2005), IEEE International Conference on Multimedia and Expo, 2005

[4]Chang, Yin-Wen; Hsieh, Cho-Jui, Chang, Kai-Wei; Ringgaard, Michael; Lin, Chih-Jen: *Training and Testing Low-degree Polynomial Data Mappings via Linear SVM*, Journal of Machine Learning Research 11 (2010) 1471-1490

[5]Cherkassky, Vladimir; Ma, Yunqian: *Practical selection of SVM parameters and noise estimation for SVM regression* (2004), Neural Networks, Volume 17, Issue 1, January 204, Pages 113-126

[6]Chollet, Francois: *Deep Learning with Python* (2015), Manning;

[7]Collobert, Ronan; Weston, Jason; Bottou, Léon; Karlen, Michael; Kavukcuoglu, Koray; Kuksa, Pavel: *Natural Language Processing (Almost) from Scratch* (2011), Journal of Machine Learning Research 12 (2011) 2493-2537

[8]Cortes, Corinna; Vapnik, Vladimir N.: *Support-vector networks* (1995), Machine Learning, 20 (3): 273–297;

[9]Della Pietra, Stephen; Della Pietra, Vincent; Lafferty, John: *Inducing Features of Random Fields* (1997), IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 19, no. 4, april 1997

[10]Ganu, Gayatree; Elhadad, Noemie; Marian, Amelie: *Beyond the stars: Improving rating predictions using review text content* (2009), In Proceedings of WebDB, Providence, Rhode Island, USA;

[11]Gensim, https://radimrehurek.com/gensim/intro.html

[12]GloVe dataset, https://nlp.stanford.edu/projects/glove/

[13]Goldberg, Yoav; Elhadad, Michael: *splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications*, Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 237–240, Columbus, Ohio, USA, June 2008

[14]Goodfellow, Ian; Bengio, Yoshua; Courville, Aaaron : *Deep Learning*. MIT Press, 2016

[15]Graves, A.; Mohamed, A.; Hinton, G.: *Speech recognition with deep recurrent neural networks* (2013), In Proceedings of ICASSP 2013

[16]Greene, William H: *Econometric Analysis (Seventh Ed.)* (2012), Boston: Pearson Education. pp. 803–806. ISBN 978-0-273-75356-8.

[17]Hahnloser, Richard HR; Sarpeshkar, Rahul; Mahowald, Misha A.; Douglas, Rodney J.; Seung, Sebastian H.: *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit* (2010), Nature 405, 6789 (2000), 947.

[18]Hercig, Tomas; Brychcin, Tomas; Svoboda, Lukas; Konkol, Michal: *UWB at SemEval-2016 Task 5: Aspect Based Sentiment Analysis* (2016), Proceedings of SemEval-2016, pages 342–349, San Diego, California, June 16-17, 2016.

[19]Horrigan, John B.: *Online shopping* (2008) Pew Internet & American Life Project Report, https://www.pewinternet.org/2008/02/13/online-shopping/;

[20]Hsu, Chih-Wei; Chang, Chih-Chung; Lin, Chih-Jen: *A Practical Guide to Support Vector Classification* (2016), Department of Computer Science, National Taiwan University, Taipei 106, Taiwan, http://www.csie.ntu.edu.tw/~cjlin

[21]Hu, Minqing; Liu, Bing: *Mining and summarizing customer reviews* (2004a), In Proceedings of KDD, pages 168–177, Seattle, WA, USA;

[22]Kalchbrenner, N.; Grefenstette, E.; Blunsom, P.: *A Convolutional Neural Network for Modelling Sentences* (2014), In Proceedings of ACL 2014

[23]Karampatsis, Rafael Michael; Pavlopoulos, John; Malakasiotis, Prodromos: *AUEB: Two Stage Sentiment Analysis of Social Network Messages* (2014), Proceedings of the 8th

International Workshop on Semantic Evaluation (SemEval 2014), pages 114–118, Dublin, Ireland, August 23-24, 2014;

[24]Khalil, Talaat; El-Beltagy, Samhaa R.: *NileTMRG at SemEval-2016 Task 5: Deep Convolutional Neural Networks for Aspect Category and Sentiment Extraction* (2016), Proceedings of SemEval-2016, pages 271–276, San Diego, California, June 16-17, 2016.

[25]Kim, Joshua: *Understanding how Convolutional Neural Network (CNN) perform text classification with word embeddings* (2017), http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/

[26]Kim, Yoon; *Convolutional Neural Networks for Sentence Classification* (2014), New York University, https://www.aclweb.org/anthology/D14-1181

[27]Kingma, Diederik P.; Ba, Jimmy Lei: *Adam: A Method for Stochastic Optimization* (2014), ICLR 2015, https://arxiv.org/pdf/1412.6980.pdf

[28]Krishnapuram, Balaji; Carin, Lawrence; Figueiredo, Mario A.T.; Hartemink, Alexander J.: *Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds* (2005), IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 6, June 2005

[29]Krizhevsky, A.; Sutskever, I.; Hinton, G.: *ImageNet Classification with Deep Convolutional Neural Networks* (2012), In Proceedings of NIPS 2012

[30]Kumar, Ayush; Kohail, Sarah; Kumar, Amit; Ekbal, Asif; Biemann, Chris: *IIT-TUDA at SemEval-2016 Task 5: Beyond Sentiment Lexicon: Combining Domain Dependency and Distributional Semantics Features for Aspect Based Sentiment Analysis* (2016), pages 1129–1135, San Diego, California, June 16-17, 2016

[31]Lafferty, John, McCallum, Andrew; Pereira, Fernando C.N.: *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data* (2001), Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001), pages 282-289

[32]LeCun, Y.; Bengio, Y.; Hinton, G.: *Deep learning,* Nature, 521(7553):436, 2015.

[33]Lin, Hsuan-Tien; Lin, Chih-Jeh: *A study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods,* Neural Computation, 2003

[34]Liu, Pengfei; Joty, Shafiq; Menh, Helen: *Finegrained Opinion Mining with Recurrent Neural Networks and Word Embeddings* (2015), In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, September.

[35]Logistic Regression, scikit API, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression

[36]Lu, Lu; Shin, Yeonjong; Su, Yanhui; Karniadakis, George Em: *Dying ReLU and*

[37]Minka, T: *A Comparison of Numerical Optimizers for Logistic Regression* (2003), technical report, Dept. of Statistics, Carnegie Mellon Univ., 2003

[38]Nair, Vinod; Hinton, Geoffrey E.: *Rectified linear units improve restricted boltzmann machines* (2010), In ICML, pages 807–814.

[39]NLTK platform ,WordnetLemmatizer, https://www.nltk.org/_modules/nltk/stem/wordnet.html

[40]Pang, Bo; Lee, Lillian; Vaithyanathan, Shivakumar: *Thumbs up? sentiment classification using machine learning techniques* (2002), in proceedings of EMNLP, pages 79–86, Philadelphia, Pennsylvania, USA;

[41]Pang, Bo; Lee, Lillian; *A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts* (2004), Proceedings of the ACL, 2004

[42]Pennington, Jeffrey; Socher, Richard; Manning, Christopher D.: *Global Vectors for Word Representation* (2014), Computer Science Department, Stanford University, Stanford, CA 94305

[43]Pontiki, Maria; Galanis, Dimitrios; Pavlopoulos, John; Papageorgiou, Harris; Androutsopoulos, Ion; Manandhar, Suresh: *SemEval-2014 Task 4: Aspect Based Sentiment Analysis* (2014), In Proceedings of the 8th International Workshop on Semantic Evaluation, Dublin, Ireland;

[44]Pontiki, Maria; Galanis, Dimitrios; Papageorgiou, Harris; Manandhar, Suresh; Androutsopoulos, Ion: *SemEval-2015 Task 12: Aspect Based Sentiment Analysis* (2015), In Proceedings of the 9th International Workshop on Semantic Evaluation, Denver, Colorado;

[45]Pontiki, Maria; Galanis, Dimitrios; Papageorgiou, Harris; Androutsopoulos, Ion; Manandhar, Suresh; AL-Smadi, Mohammad; Al-Ayyoub, Mahmoud; Zhao, Yanyan; Qin, Bing; De Clercq, Orphée; Hoste, Véronique; Apidianaki, Marianna; Tannier, Xavier;

Loukachevitch, Natalia; Kotelnikov, Evgeny; Bel, Nuria; Jiménez-Zafra, Salud María; Eryiğit, Gülşen: *Semeval-2016 task 5: Aspect based sentiment analysis* (2016), In Proceedings of SemEval 2016, San Diego, California;

[46]Popescu, Ana-Maria; Etzioni, Oren*: Extracting product features and opinions from reviews* (2005), In Proceedings of HLT/EMNLP, pages 339–346, Vancouver, British Columbia, Canada;

[47]PorterStemmer tool, https://www.nltk.org/_modules/nltk/stem/porter.html ;

[48]Pystruct models, ChainCRF API, https://pystruct.github.io/generated/pystruct.models.ChainCRF.html

[49]Pystruct models, FrankWolfeSSVM, https://pystruct.github.io/generated/pystruct.learners.FrankWolfeSSVM.html

[50]Sarkar, Kanchan: *ReLU : Not a Differentiable Function: Why used in Gradient Based Optimization? and Other Generalizations of ReLU* (2018), https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec

[51]Schmidhuber, Jürgen: *Deep learning in neural networks: An overview* (2015), Neural Networks, p.85-117, Elsevier, 2015.

[52]Severyn, Aliaksei; Moschitti, Alessandro: *UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification* (2015), Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 464–469, Denver, Colorado, June 4-5, 2015.

[53]Sharma, Sagar; *Activation Functions in Neural Networks* (2017), https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[54]SVM API, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[55]Thet, Tun Thura; Na, Jin-Cheon, Khoo, Christopher S. G.: *Aspect-based sentiment analysis of movie reviews on discussion boards* (2010), J. Information Science, 36(6):823–848;

[56]Toh, Zhiqiang; Su, Jian: *NLANGP at SemEval-2016 Task 5: Improving Aspect Based Sentiment Analysis using Neural Network Features* (2016), Proceedings of SemEval-2016, pages 282–288, San Diego, California, June 16-17, 2016.

[57]Turney, Peter: *Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews* (2002), In Proceedings of ACL, pages 417–424, Philadelphia, Pennsylvania, USA;

[58]Vapnik, Vladimir: *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, 1982.

[59]Wilson, Theresa; Wiebe, Jaynce; Hoffmann, Paul: *Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis* (2005), Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing Pages 347-354;

[60]Xenos, Dionysios; Theodorakakos, Panagiotis; Pavlopoulos, John; Malakasiotis, Prodromos; Androutsopoulos, Ion: *AUEB-ABSA at SemEval-2016 Task 5: Ensembles of Classifiers and Embeddings for Aspect Based Sentiment Analysis* (2016), Proceedings of SemEval-2016, pages 312–317, San Diego, California, June 16-17, 2016;

[61]Yelp Dataset, https://www.yelp.com/dataset/challenge

[62]Zeiler, Matthew D.: *ADADELTA: An Adaptive Learning Rate Method* (2012), New York University, USA, https://arxiv.org/pdf/1212.5701.pdf