MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

SISTEM DE CLASIFICARE A OPINIILOR DESPRE DISPOZITIVE MOBILE UTILIZAND TEHNOLOGII APACHE

LUCRARE DE LICENŢĂ

Absolvent: **Alina-Denisa Ilieș**

Coordonator ştiinţific: **As. Ing. Cosmina IVAN**

2018

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Alina-Denisa Ilieș**

SISTEM DE CLASIFICARE A OPINIILOR DESPRE DISPOZITIVE MOBILE UTILIZAND TEHNOLOGII APACHE

1. **Enunţul temei:***Proiectul își propune realizarea unui sistem care să analizeze și clasifice în timp real mesajele postate pe rețeaua de socializare Twitter despre dispozitive mobile utilizând tehnologii Apache. Analiza și clasificarea vor fi aplicate unor date de intrare din lumea reale și vor fi implementate utilizând tehnologiile : Apache Spark (Apache Spark Streaming, Apache Spark Mllib), Apache Zookeeper, Apache Kafka, Apache Zeppelin.*

2. **Conţinutul lucrării:***Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalare și Utilizare, Concluzii și Dezvoltări Ulterioare, Bibliografie.*

3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare

4. **Consultanţi**:

5. **Data emiterii temei:** 1 februarie 2018

6. **Data predării:** 9iulie 2018

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind
autenticitatea lucrării de licenţă

Subsemnatul(a)_____
_____,
legitimat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării
_____
_____
_____elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea _____ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

_____

Nume, Prenume

_____

Semnătura

Cuprins

Capitolul 1. Introducere – Contextul proiectului

Scopul acestui proiect este construirea unui sistem capabil să analizeze și să clasifice în timp real recenziile despre dispozitive mobile postate pe rețeaua de socializare Twitter.

1.1. Contextul proiectului

Scopul proiectului poate deservii interprinderilor mici și mijloci pentru a-și analiza business-ul. În zilele noastre, analiza business-ului, Business Inteligence[1]în limba engleză este o parte esențială în succesul unei afaceri. Acest termen se referă la aplicații realizate în cadrul companiilor care au rolul de a extrage și analiza datele disponibile într-o companie cu scopul de a oferi suport în luarea unor decizii legate de afacere. Așadar, acest sistem va oferi utilizatorilor posibilitatea de a afla părerile altor oameni despre anumite brand-uri, lucru care va fi util în luarea unei decizii.

Termenul“Big data” reprezină seturi de date care depășesc capacitatea de procesare a sistemelor de baze de date tradiționale.Fenomenul care stă în spatele acestuia reprezintă un domeniu de cercetare popular in știința calculatoarelor, dar apare și în alte domenii precum: cercetări biologice, meteorologie, finanțe si afaceri, asistență medicală. Acesta este stâns legat și de Internet, astfel datele stocate in acesta din toate domeniile enunțate vor cunoaște o creștere semnificativă. Un raport în acest sens a fost elaborate de International Data Corporation (IDC) care afirmăîn [8] ca până în în intevalul 2005- 2020 universul digital va crește cu un factor de 300, de la 130 exabytes la 44 000 exabytes.

Nu toate cantitățile mari de dat generate conțin informații valoroase, iar o încercare de a procesa toate aceste date poate fi fără folos. În raportul menționat anterior se definesc 5 criterii pentru ca datele sa aducă valoare : să fie ușor de accesat, să fie date generate în timp real, să își pună amprenta asupra unui număr mare de indivizi dintr-o populație, să fie benefice companiilor sau utilizatorilor după procesare și să conțina cel puțin un criteriu din cele prezentate.

Creșterea amintită anterior este datorată faptului ca numarul dispozitivelor conectate la Internet sunt din ce in ce mai multe, iar acestea generează în mod constant date prin : senzorii atașați acestora care colectează informații din jurul lor, tranzacțiile online care sunt mai accesibile datorită dispozitivelor mobile, gps-uri. O altă sursă importantă pentru generarea datelor o reprezină rețelele sociale deoarece oamenii preferă comunicarea online in detrimentul celei fizice, iar astfel sunt distribuite foarte ușor informații depre vaste domenii într-un mod foarte rapid.

Oamenii își exprimă opiniile și experiențele pe rețele de socializare și este important ca acestea să fie analizate deoarece societatea umană se lasă influențată de p-ărerile celor din jur in ceea ce privește luarea unor decizii noi. In cadrul acestui proiect aceste date sunt generatee pe baza rețelei sociale Twitter. Extragereainformațiilor din date, termen numit in engleza data-mining reprezină procesul de analiză a unei cantități mari de date și preluarea informațiilor necesare și relevante, utilizând diferite metode.

Datorită mărimii și complexității acestora au devenit greu de gestionat utilizând instrumente clasice. Astfel a apărut nevoia realizării si utilizării unor tool-uri actuale ,

[1]https://ro.wikipedia.org/wiki/Business_intelligence

precum Apache Hadoop pentru a satisface aceste nevoi. In cadrul lucrării se vor prezenta arhitecturi si framework-uri pentru procesarea big-data

1.2. Conținutul lucrării

Lucrarea este structurată în 8 capitole, urmând ca în continuare sa fie prezentat succint conținutul fiecăruia :

Primul capitol - Introducere : este motivată alegerea temei și plasarea acesteia în contextul specific. De asemenea, se prezintă complexitatea domeniului "Big Data" și necesitatea stocării și procesării datelor din acest domeniu.

Al doilea capitol – Obiectivele Proiectului : sunt enumerate obiectivele urmărite ale proiectului, atât cele principale cat și cele secundare. Obiectivul principal este impărțit in pași exacți, care, îndepliniți vor conduce la realizarea unui sistem funcțional.

Al treilea capitol – Studiu Bibliografic : oferă o explicație mai clară a conceptului "Big Data" si se descriul metodele de clasificare a textului. De asemenea, se prezintă modele de procesare a datelor : modelul MapReduce, procesarea streaming și arhitectura Lambda. In continuare se prezintă informații teoretice despre cloud și alte sisteme care se bazează pe ideile propuse.

Al patrulea capitol – Analiză si Fundamentare Teoretică : se prezină perspectiva tehnologică bazată pe tehnologiile Apache. Se realizează o descriere teoretică a tehnologiilor și API-urilor folosite și modul de funcționare a acestora și un studiu al principalului framework folosit : Apache Spark. Acest capitol își propane să pună în evidența atât avantajele cât și dezavantajele acestor tehnologii. Totodată se prezintă conceptual modul de funcționare al sistemului și scenariile care există.

Al cincilea capitol – Proiectare de Detaliu și Implementare : descrie cu amănunte arhitectura si scenariile care au fost implementate utilizând Apache Spark pentru a clasifica părerile utilizatorilor rețelei Twitter despre dispozitive mobile.Tweet-urile au fost colectate , preprocesate si analizate utilizând algoritmi de învătare pentru a obține o clasificare a acestora.

Al șaselea capitol – Testare și validare : prezintă modul prin care aplicația poate fi testată pentru a valida corectitudinea rezultatelor obținute.

Al șaptelea capitol – Manual de instalare si utilizare : surprinde etapele și pașii necesari pentru a instala tool-urile folosite și pentru a rula aplicația, dar și modul in care aceasta poate fi utilizată.

Ultimul capitol – Concluzii : prezintă studiul si starea sistemului implementat, precizează în ce măsură au fost îndeplinite obiectivele si propune posibilele dezvoltări ulterioare.

Capitolul 2. Obiectivele Proiectului

În acest capitol se prezintă obiectivul principal al proiectului implementat , dar și o serie de obiective secundare, care îndeplinite vor conduce spre un sistem funcțional. Lucrarea va realiza o descriere detaliata a modelelor si arhitecturilor pentru procesarea datelor în timp real, dar si o descriere a framework-urilor și tehnologiilor folosite pentru creearea sistemului.

2.1. Obiectivul principal

Scopul constă în realizarea unei aplicații reale care va clasifica opiniile utilizatorilor rețelei sociale Twitter despre anumite dispozitive mobile și afișsarea rezultatelor clasificării într-un mod ușor de înțeles pentru utilizatori. Obiectivul principal este colectarea în timp real a tweet-urilor și procesarea acestora utilizând tehnologii Apache.

2.2. Obiective secundare

Este necesar ca răspunsul sistemului să fie unul valoros pentru utilizatorii care încearcă sa găsească alte opinii despre dispozitivele pe care încearcă să le achiziționeze, așadar datele cu care acesta lucrează trebuie să facă parte din lumea reală.

Așadar, datele colectate sunt tweet-uri postate, bineînțeles, pe rețeaua socială Twitter. Astfel, *primul obiectiv este colectarea tweet-urilor folosind cuvinte cheie actuale despre fiecare dispozitiv mobil* în parte, iar acestea vor fi colectate pentru o perioadă mai mare de timp astfel încât fișierul rezultat să ofere o antrenare cât mai bună a sistemului.

Un alt obiectiv este documentarea, identificarea si utilizarea unei metode pentru clasificarea opiniilor colectate. Scopul acestui obiectiv este alegerea celui mai potrivit algoritm pentru necesitățile sistemului din multitudinea de algoritmi existenți în domeniul machine learning.În urma acestei documentări, s-a stabilit ca metoda folosită să fie un clasificator de tip Naive Bayes.

Singurele filtre aplicate streaming-ului de date sunt doar cuvintele cheie după care se selctează datele importante pentru domeniul de interes și limba în care tweet-urile sunt postate pentru a obține rezultate relevante în urma clasificării. Alte filtre precum profilul utilizatorului care postează sau locația unde este identificat acesta nu erau relevante deoarece, în zilele noastre , oamenii se bazează și pe opiniile unor străini, din orice colț al lumii. Așadar *unul din obiectivele propuse este clasificarea sentimentelor extrase din recenziile unor brand-uri de dispozitive mobile, clasificare bazată pe algoritmii de machine learning.*

În urma selectării metodei de clasificare prezentate anterior, aceasta se va aplica datelor colectate deoarece un *alt obiect este antrenarea sistemului.* După ce sistemul este antrenat acesta va ști pentru un set nou de date să determine polaritatea sentimentelor extrase din acestea.Reazultatul testării acestei funcționaliăți va fi vizibil pentru utilziatorii sistemului și îi va îndruma, probabil, în alegerea unui dispozitiv mobil.

Pentru ca utilizatorul sa primească un răspuns prompt, în timp real, *se va introduce sistemul de mesagerie Apache Kafka* care va face legatura intre streaming-ul de date si sistemul antrenat. Acesta poate rula în mai multe noduri, iar pentru gestionarea acestora *este necesară intorducerea tehnologiei Apache Zookeper*.

Un alt obiectiv este introducerea tehnologiei Apache Zeppelin pentru a analiza si a vizualiza rezultatele într-un mod simplu și ușor de înțeles.

Pentru a asigura scalabilitatea sistemului în fața unei creșteri mari a datelor si accesibilitatea acestuia pentru a fi disponibil oricând utilizatorului, *se dorește rularea în cloud al sistemului implementat.*

Capitolul 3. Studiu Bibliografic

In acest capitol este prezentat stadiul actual al domeniului si al subdomeniilor în care este plasată tema acestei lucrări.

Astfel, inițial se vor prezenta metode de clasificare utilizate in diferite sisteme pentru clasificare de text,sentimente, etc. În continuare se vor prezenta arhitecturi pentru procesarea datelor, urmate de o introducere teoretică în arhitectura cloud. În final se vor prezenta alte sisteme similare cu cel propus pentru implementare.

3.1. Metode de clasificare

Datele sunt într-o continuă expansiune în mediul online, iar oamenii scriu din ce în ce mai multe păreri despre diferite subiecte. Utilizând metodele de clasificare, aceste păreri pot fi utile pentru alți utilizatori pentru a avea o prezentare generală despre anumite produse. Utilitatea metodelor de clasificare constă în faptul că în urma clasificării acestea primesc o etichetă : pozitiv, negativ sau neutru. Astfel utilizatorul va ști nuanța opiniilor altor utilizatori despre un anumit subiect. O mare parte din eforturile îndreptate spre cercetare din ultimii ani au fost dedicate clasificării automate a textului. În continuare se vor prezenta tehnicile care ajută la îndeplinirea acestui obiectiv : tehnicile bazate pe machine learning si tehnicile bazate pe lexicon.

Există două abordări pentru clasificarea automată : tehnicile bazate pe machine learning si cele bazate pe lexicon.

Abordarea bazată pe machine learning utilizează un set de date etichetate cu ajutorul căruia clasificatorul este antrenat. În urma antrenării clasificatorul va putea prezice rezultatul unei probleme noi apărute.

Abordarea bazată pe lexicon împarte datele de intrare în cuvinte individuale care sunt verificate cu un lexicon pentru sentimente care conține valori de polaritate pentru cuvintele individuale. Suma acestor polarități este transmisă ca date de intrare unui algoritm care determină polaritatea unei întregi propoziții.

În general, toate metodele de clasificarea supravegheate care vor fi prezentate vor urma un șablon standard care conține cinci pași : creearea datelor de intrare, pre-procesarea acestora prin extragerea caracteristicilor, transformarea lor în vectori, instruirea clasificatorului cu ajutorul unuia dintre algoritmii ce urmează să fie prezentați și testarea clasificatorului. Primii patru pași fac parte din etapa de antrenare, iar toți cei cinci pași pot fi incluși în etapa de prezicere. Acești pași sunt prezentați grafic în figura 3.1

Crearea datelor de intrare presupune colectarea datelor bazate pe categorii, în cazul sistemului tweet-uri despre diferite brand-uri. Fiecare text aparține unei categorii și primește eticheta corespunzătoare. De obicei aceste date sunt împărțite în date pentru antrenare și date pentru testare.

Pre-procesarea presupune eliminarea tuturor caracterelor și cuvintelor care nu sunt necesare în tweet cum ar fi stop-word-urile sau semnele de punctuație. Acest pas

este foarte important deoarece în funcție de zgomotul existent în sensul de date, antrenarea sistemului poate fi afectata.

Vectorizarea textului transformă textul într-o vector cu valori reale pentru a putea fi recunoscut de către sistem, adică de către clasificator. Fiecare tweet va fi reprezentat ca un vector de caracteristici.

Antrenarea clasificatorului presupune alegerea unui algoritm de clasificarea și transmite setul de date pentru antrenare spre clasificator în scopul învățării acestuia.

Testarea clasificatorului presupune transmiterea setului de date spre clasificatorul deja antrenat care va returna o etichetă pentru fiecare tweet.

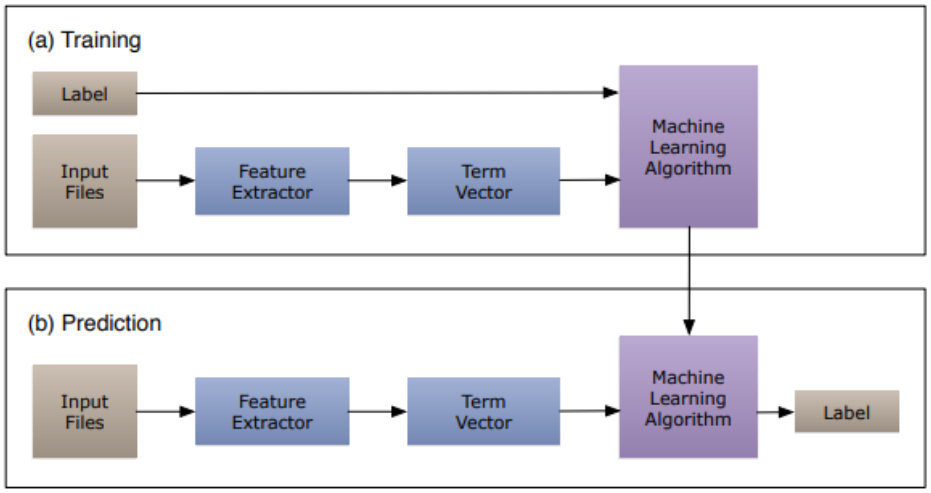

Figura 3.1 Pași necesari pentru a crea un sistem de clasificare (Sursa : [16])

3.1.1. Tehnici Machine Learning – ML

În urma aplicării algoritmilor de machine learning, datele primesc o etichetă : pozitivă, negativă sau neutra , iar acești algoritmi sunt repartizați în două categori, în funcție de modul de învățare : supravegheată sau nesupravegheată. Figura 3.2 ilustrează cei mai utilizați algoritmi pe baza celor două metode de învățare

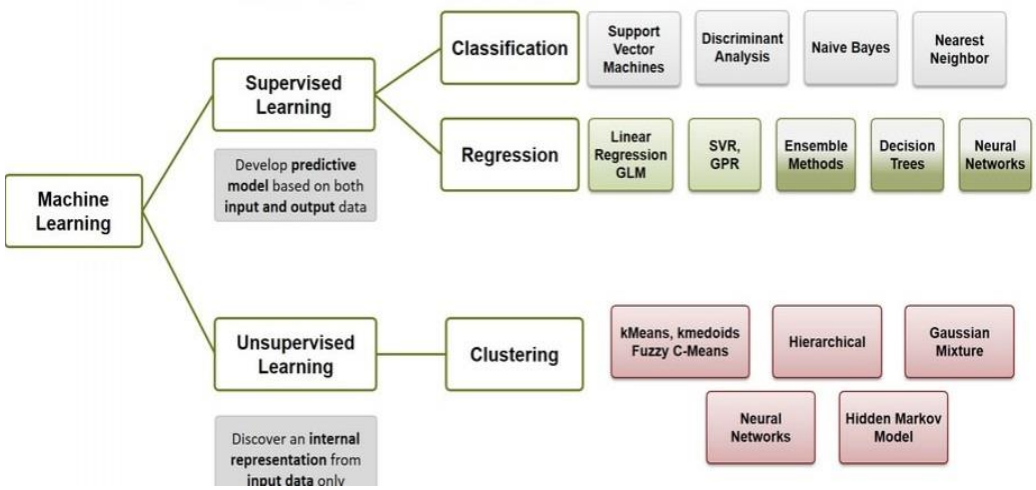

Figura 3.2 Algoritmi machine learning [2]

Învățarea supravegheată sugerează, prin numele ei, prezenta unui supraveghetor care instruiește sistemul de învățare să asocieze etichetele datelor cu unele exemple, lucru sugerat în [1]. Această învățare presupune o mapare între un set de variabile de intrare și o variabilă de ieșire și aplicarea mapării pentru a prezice rezultatul pe un set de date neetichetate. Această metodă reprezintă cea mai important metodologie în tehnicile de machine learning și este importantă și în procesarea datelor multimedia.

Învățarea nesupravegheată nu necesită un set de date etichetate. Metode din această categorie precum Clustering și Topic Modeling sunt utilizate în cazuri în care datele necesare pentru antrenarea sistemului sunt indisponibile sau greu de găsit. Aceste metode propun gruparea automată a grupurilor similare de date sau obiecte în anumite colecții.

Există unele carateristici ale problemelor de clasficare a sentimentelor, care luate în considerare și aplicate vor crește acuratețea sistemului.Prima astfel de caracteristică este **etichetarea părților de vorbire** și este foarte utilizată pentru analiza sentimentelor. Un exemplu pentru implementarea acestei caracteristici sunt adjectivele, care sunt considerate a fi indicatori puternici pentru exprimarea sentimentelor. Primele studii în acest domeniu au încercat identificarea caracteristicilor adjectivelor și corelarea lor cu contextul propoziției pentru a sugera subiectivitatea acesteia. Pe lângă adjective există și alte părți de vorbire care pot indica subiectivitatea : verbe – a plăcea, substantive – comoară.

Modificatorii de sentimente sunt o altă caracteristică importantă în cazul analizei sentimentelor deoarece aceștia schimbă polaritățile cuvintelor.Cel mai întâlnit exemplu în acest sens sunt cuvintele de negare care trebuie atașate unei caracteristici pentru a nu

[2] http://www.embedded-computing.com/embedded-computing-design/analytics-driven-embedded-systems-part-2-developing-analytics-and-prescriptive-controls

schimba polaritaea. În exemple de tipul “a nu plăcea” este necesară atașarea cuvântului nu de caracteristica plăcea, în caz contrar sintagma ar putea fi interpretată drept pozitivă.

A treia caracteristică importantă este **frecvența sau prezența termenilor** deoarece numărul frecvenței termenilor este importană în clasificarea textului. Termenii individuali se numesc unigrame, dar în unele cazuri se folosesc și n-grame (bigrame sau trigrame). Nu se poate specifica exact dacă unigramele sunt mai eficiente decât n-gramele, acest lucru depinzând de contextul clasificării. Pentru clasificarea clasică a textului este mai importană caracteristica de prezență a termenului decât frecvența acestuia.

Sentiment lexicon este a patra caracteristică din această categorie și utilzează frecvent cuvinte pentru sentimente pentru a exprima polaritățile propozițiilor. Sunt utilizare cuvinte precum “bun”, “iubire”, “încântare” pentru a exprima pozitivitate și cuvinte precum “trist”, “rău” pentru a exprima negativitate. Majoritatea cuvintelor întâlnite ca exemple în acet sens sunt adjective, însă pot fi utilizate și alte părți de vorbire: substantive, verbe, locuțiuni. Toate cuvintele și expresiile au rolul de a forma un lexicon care este ulterior utilizat în analiza și clasificarea sentimentelor.

A cincea caracteristică este **regula pentru opinii** și reprezintă regulile dintr-un anumit context sau domeniu în funcție de care se identifică polaritatea textelor. Aceste reguli sunt utilizate și pentru a obține o acuratețe mai mare a rezultatelor clasificatorului.

Emoticoanele reprezină simboluri puternice prezente în comunicarea din era digital, fiind în ultimul timp un mod mai accesibil pentru indivizi de a-și exprima sentimentele.Analizarea lor este foartă utilă în determinarea polarității mesajelor, astfel încât identificarea unei inimi sau a unei feței care zâmbește poate indica un sentiment pozitiv, iar identificarea unei fețe nervoase sau întristate poate indica un sentiment negativ.

3.1.1.1. Naive Bayes

Clasificatorul Naive Bayes prezentat în [5] este cel mai simplu și cel mai utilizat clasificator. În ceea ce privește clasificarea sentimentelor, acesta calculează probabilitatea unei polarități în funcție de cuvintele din tweet. Acesta folosește teorema lui Bayes pentru a prezice probabilitatea ca un anumit set de caracteristici să aparțină unei polarități particulare :

$$P(\text{etichetă}|\text{caracteristică}) = \frac{\mathrm{P}(\text{etichetă}) * \mathrm{P}(\text{caracteristică}|\text{etichetă})}{\mathrm{P}(\text{caracteristică})}$$

unde P(etichetă) este probabilitatea unei etichete sau polaritați să fie setata de o caracteristică întâmplătoare, iar P(caracteristică|etichetă) este probabilitatea ca un set dat de caracteristici să fie etichetat cu o polaritate. P(caracteristică) este probabilitatea să

existe un set de caracteristici. Astfel, dacă aceste date se cunosc se poate calcula probabilitatea atribuirii unei polarități în funcție de caracteristicile extrase din tweet.

Deoarece acest clasificator se bazează pe idea că toate caracteristicile sunt independente, formula anterioare se poate rescrie în felul următor:

$$P(\text{etichetă}|\text{caracteristică}) = \frac{\text{P(etichetă)} * \text{P(caracteristică1}|\text{etichetă}) * \ldots * \text{P(caracteristicăn} \mid \text{etichetă})}{\text{P(caracteristică)}}$$

3.1.1.2. Entropia Maximă

Clasificatorul Maximum Entropy introdus în [2] este un algoritm probabilistic supervizat din categoria machine learning. Spre deosebire de Naive Bayes, acesta presupune caracteristicile ca fiind dependente unele de altele. În unele scenarii clasificatorul Maximum Entropy surclasează clasificatorul prezentat anterior , dar nu în toate cazurile. Acest clasificator este folosit pentru a estima distribuția probabilistică a datelor. El se bazează pe principiul următor : dacă nu există cunoștințe prioritate despre anumite date, atunci acestea ar trebui distribuite întamplător sau uniform. De exemplu, presupunem că avem răspunsurile unor studenți la o examinare a limbii engleze, răspunsuri care pot veni în patru variante : scrisoare, eseu descriptiv, eseu argumentativ sau poveste. Presupunând ca avem 40 % șanse ca răspunsul să fie "argument" dacă acesta conține cuvantul opinie. Conform principiului de distribuție uniform, există doar 20 % șanse pentru celelalte trei tipuri de răspunsuri. Deoarece nu există informații suplimentare despre răspunsul studentului, probabilitatea este distribuită uniform pentru toate răspunsurile, fiecare având șansa de 25 % în cazul în care cuvântul din răspuns este "argument".

Metoda distribuției uniforme a probabilităților este aplicată în multe probleme de clasificare a textului cum ar fi : indentificarea limbii în care textul este scris, analiza sentimentelor sau clasificarea topicurilor. Există o strânsă relație între distribuția datelor și entropie : cu cât distribuția uniform este mai mare, cu atât entropia va fi mai mare. Scopul este maximizarea entropiei, dar asigurarea consistenței cu privire la constrângerile legate de date. În cazul clasificării utilizând entropia maximă, primul pas este alegerea unui set de caracteristici care sunt necesare clasificării documentului sau sentimentelor. Pasul următor este calcularea valorii așteptate pentru fiecare caracteristică din setul de date, iar din acest lucru derivă constrângerea modelului de distribuție. În urma antrenării clasificatorului cu constrângerea dată, se poate incerca predicția pe un nou document sau text.

3.1.1.3. Suport Vector Machines - SVM

Suport Vector Machines este un algoritm de clasificare supravegheată care are nevoie de date pentru antrenament, după cum este specificat în [3]. Spre deosebire de cele două clasificatoare prezentate anterior, care erau probabilistice, acesta este unul liniar. Pentru a aplica SVM, primul pas este reprezentarea tututor punctelor de date într-un graf de dimensiune n, unde n este numărul total de caracteristici. Vectorul suport este

reprezentat de coordonatele fiecărui punct de date, iar scopul SVM este găsirea unui hiperplan care separă o clasă de alta. Acest lucru este reprezentat în figura 3.3

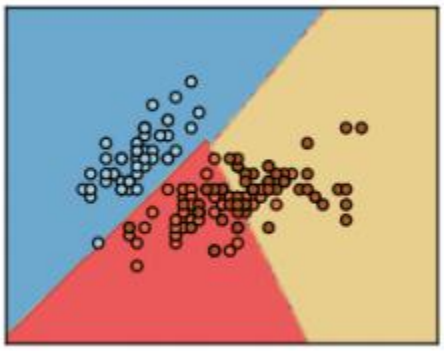

Figura 3.3 Clase separate de un hiperplan (Sursa : [4])

Separarea se numește margine și ar trebui să fie cât mai mare. Este posibilă calcularea mai multor hiperplane care separă cu succes setul de dare pentru antrenament în clase distincte. Hiperplanul devine optimal atunci când marginea între datele de antrenament este maximă. Dacă separarea nu este una optimală clasificarea datelor neseparate ar putea introduce erori în sistem.

SVM poate fi aplicat pe mai mult de două dimensiuni (sau caracteristici). După ce hiperplanul este identificat, un nou răspuns poate fi etichetat. În cazul în care punctele nu pot fi separate liniar, se adaugă o nouă caracterisitcă ‘z’. Z se calculează ca fiind suma pătratelor dintre x și y și se treasează un grafic între z și x. În urma acestei modificări datele vor fi liniar separate. Această modificare se numește kernel, iar SVM folosește funcții numite kernel pentru a transforma automat spații dimensionale mici în spații dimensionale mai mari când datele nu pot fi separate liniar.

3.1.2. Tehnici bazate pe lexicon

Un clasificator realizat cu tehnici bazate pe lexicon prezentat în [6] are nevoie doar de un document de polaritate care să conțină cuvinte și orientarea lor semantică. Acesta nu are nevoie de antrenament sau alte preprocesări înainte de a fi utilizat. Reazultatul clasificării este reprezentat binar ca un scor pozitiv sau negativ, iar uneori poate fi inclus și un scor neutru. Un asemenea clasificator este mai simplu de creat și implementat față de un clasificator bazat pe învățare deoarece nu este necesară existența unui set de date etichetate pentru antrenament. Deși perofmanțele unui clasificator realizat cu tehnicile bazate pe lexicon sunt mai slabe decât perfomanțele unui clasificator bazat pe învățare, acesta are unele beneficii. Clasificatorul bazat pe lexicon nu are un domeniu specific de cunoștințe, iar perofmanțele unui clasificator bazat pe învățare pot

scădea considerabil daca aceste folosește date dintr-un alt domeniu față de datele cu care a fost antrenat.

Există două tipuri de cuvinte care ar trebui evitate în aceste tipuri de clasificare : cuvintele negative și cuvintele care au rolul de a intensifica sentimentele. Problema o repezintă faptul că nu există conceptul de negație sau intensificare pentru clasificator.

Pentru a implementa un asemenea clasificator se consideră două abordări : bazată pe dicționar și bazată pe corpus.

Clasificatorul bazat pe dicționar

În cazul acestui clasificator există un proces iterativ pentru a crea lexiconul de opinie. În primul pas se alege un set de cuvinte și polaritatea acestora, iar apoi sunt căutate și selecționate sinonimele și antonimele pentru cuvintele alese, repetându-se acești pași până când nu se găsesc alte cuvinte noi. Lista în care aceste cuvinte sunt adăugate se numește lista de seed, iar aceasta este verificată de obicei manual pentru a elimina erorile ce pot apărea.

Clasificatorul bazat pe corpus

Acest clasificator presupune selectarea unor cuvinte de opinie inițiale. În pasul următor sunt utilizate șabloane similare care apar în lista seed de cuvinte pentru a determina orientarea altor cuvinte. Un corpus de dimensiuni mari va reuși să identifice dacă două cuvinte asociate reprezintă același sentiment. Rezultatul legăturilor dintre cuvinte formează un grafic. Dezavantajul acestei abordări este faptul că trebuie să existe un corpus suficient de mare pentru a fi utilizat.

3.1.3. Alte tehnici

Toate tehnicile prezentate anterior sunt tehnici supravegheate. Există posibilitatea utilizării tehnicilor nesupravegheate pentru clasificarea sentimentelor. Algoritmii nesupravegheați compară caracteristicile unui text țină cu un lexicon, iar polaritățile cuvintelor sunt predeterminate. Numărul cuvintelor pozitive și negative sunt numărate, iar prezența unui număr mare de cuvinte pozitive indică fapul că tweet-ul este unul pozitiv. CRF – Conditional random field – este un modul probabilistic grafic pentru a clasifica sentimelete textelor, conform [7]. Unii cercetători au abordat metodele CRF pentru a clasifica datele.

Există tehnici hibride pentru clasificare care combină mai multe metode prezentate anterior și tehnici care nu pot fi catalogate ca fiind bazate pe machine learning sau lexicon. Un exemplu pentru acest caz este FCA (Formal Concept Analysis).

3.2. Arhitecturi pentru procesarea datelor

3.2.1. Introducere

Paradigma “Big Data” a cunoscut o expansiune a popularității acestia în ultimul timp. Termenul “Big Data” este folosit pentru seturi de date cafre sunt atât de mari încât nu pot fi procesate și administrate utilizând metode clasice cum ar fi Sistemele de Baze de Date Relationale. Pe lângă volumul mare al datelor, viteza și varietatea sunt alte provocări ridicate de big data.

Dacă în trecut analiza BigData era potrivită și aplicată in corporații de dimensiuni mari precum Google, apariția tehnologiilor precum Hadoop a adus această procesare la un nivel accesibil și cu costuri reduse și pentru companii si persoane fizice cu resurse limitate.

În continuare se va descrie procesarea MapReduce, în maniera batch, în capitolul 3.2.2 și procesarea de tip streaming, în timp real, în capitolul 3.2.3. Capitolul 3.2.4 prezintă o descriere a arhitecturii Lambda.

3.2.2. Modelul MapReduce

Printre primele soluții de administrare și procesare a cantităților mari de date a fost framework-ul MapReduce care a fost introdus și folosit de Google. Acesta avea 3 mari componente : motorul de execuție MapReduce, un sistem distribuit de fișiere numit GFS(Goolge File System) și o bază de date NoSql numită BigTable.
Dupa apariția acestuia, Apache implementează un framework MapReduce similar care avea ca motoare de execuție Hadoop MapReduce și Hadoop Yarn, iar ca sistem distribuit de fișiere apare HDFS(Hadoop Distributed File System). BigTable a fost în perspectiva Apache cu HBase.

Framework-ul MapReduce duce procesarea cantităților mai de date la un nivel mai rapid și mai sigur. Deși are acest avantaj, el a fost proiectat pentru procesarea batch și nu este potrivit pentru procesarea în timp real.
MapReduce prezentat în [9] este un model de programare și execuție creat inițial de Google pentru a rezolva problema de indexare a paginilor web. Acesta împarte sarcinile masive în sarcini mai mici, iar mai apoi le procesează în paralel pe mai multe noduri. Acesta permite disitribuția unei cantități mari de date pentru care procesarea pe o singură mașină nu este potrivită.

La baza acestui program stau principiile programării funcționale și conține două funcții : map și reduce. Nodul master creează numărul de noduri slave care vor executa funcțiile specificate. În acest framework, utilizatorul furnizează un bloc de date spre funcția map. Nodul master împarte aceste date spre nodurile slave care execută funcția map. Această funcție are ca rezultat o perechi cheie-valoare a datelor de intrare și grupează aceste perechi după cheie. Rezultatul acestei funcții arată în felul următor : (cheie, (listă de valori), iar acesta este pasat mai departe nodurilor care execută funcția

reduce. Această funcție procesează datele primite si produce rezultatul final. În figura 3.4 este prezentat modul de funcționare a MapReduce.

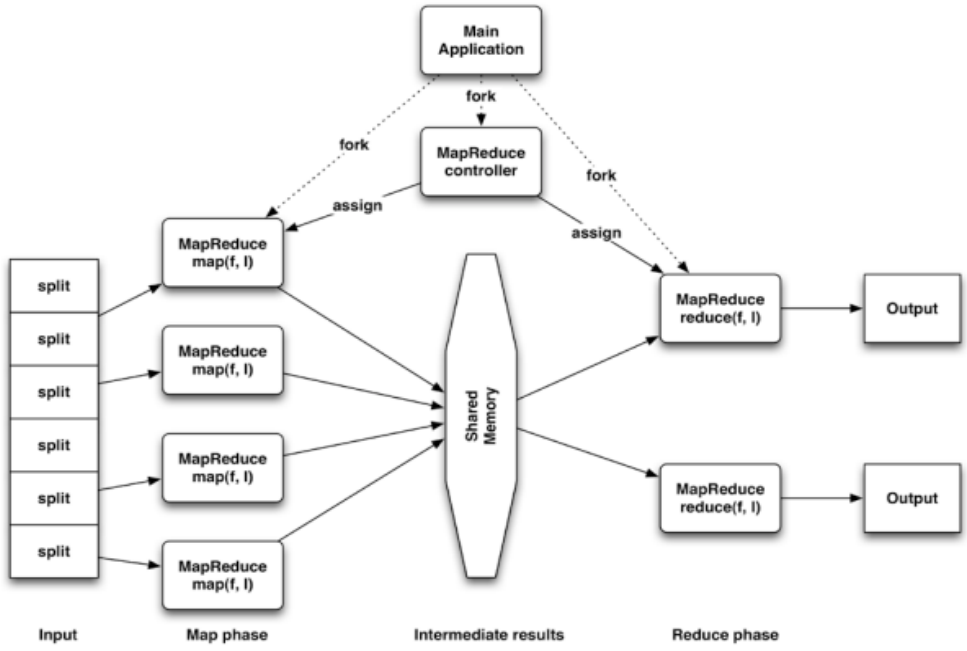

Figura 3.4 Modul de funcționare MapReduce[3]

3.2.3. Procesare Streaming

Sistemele de procesare streaming prezentate în [10] au rolul de a elimina întârzierea cu care rezultatele apar în urma procesării cu frameworkul prezentat anterior. Necesitatea implementării unor astfel de sisteme apare în aplicații precum detecția fraudei sau aplicații financiare, în care răspunsul sistemului trebuie să fie unul în timp real.

Pentru a reduce latența aceste sisteme trebuie să proceseze mesajele și să elimine costurile suplimentare de memorare. Pentru multe aplicații nu e nici necesară nici acceptabilă operația de memorare a datelor înainte de faza de procesare deoarece mesajele trebuie procesate unul după altul, în ordinea în care apar în stream. Latența poate fi dată si de pasivitatea sistemelor. Un sistem pasiv trebuie să aștepte să i se spună cum să reacționeze, lucru care va întarzia raspunsul. Așadar, un sistem de procesare streaming trebuie să elimine operația de memorare a datelor și ideal ar fi să fie un sistem activ.

O altă caracteristică a sistemelor de procesare streaming este mecanismul de interogare care se bazează pe limbajul SQL. Pentru a asigura necesitățile procesării în

[3]https://static.dzone.com/dz1/dz-files/refcardz/rc117-010d-hadoop_0.pdf

timp real, a fost conceput o variantă a limbajului SQL numită StreamSQL. Dacă un sistem SQL tradițional știe să finalizeze calculele când ajunge la finalul tabelului, în procesarea streaming a fost necesară introducerea a doi noi termeni pentru a sfârși calculele și a afișa rezultatul. Așadar s-a introdus conceptul de fereastra care definește numărul de mesaje care trebuie procesate și conceptul de slide care arată cum procesul iterează. În figura 3.5 se prezină definirea unei asemenea ferestre cu un slide 1 asupra un stream de date.

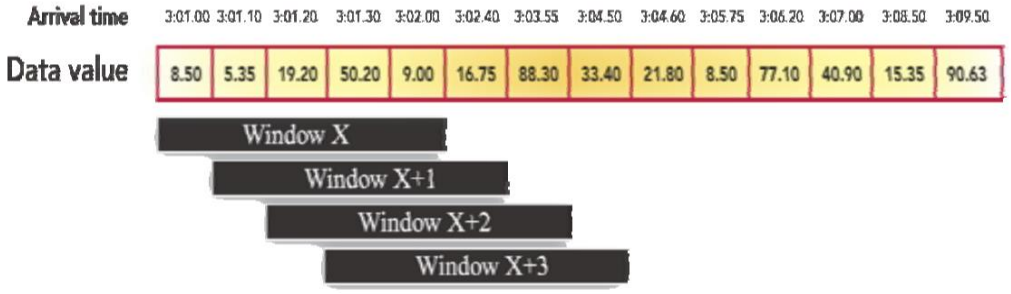

Figura 3.5 Definirea unei ferestre pentru procesare unui stream de date (Sursa : [10])

Deoarece în sistemele convenționale de baze de date datele sunt prezente în sistem înaintea procesării, în sistemele de procesare streaming trebuie să existe un mecanism care să împiedice sistemul să oprească execuția în cazul în care datele nu sunt prezente. Astfel, aceste sisteme trebuie să suporte datele cu imperfecțiuni și cele care apar în altă ordine. Datorită întarzierii cu care pot apărea datele, în cazul în care apar date cu un timestamp mai mic decât al ferestrei curente în care se realizează procesarea, aceste sisteme trebuie să fie capabile să ofere un mecanism de adăugare a unui timp adițional ferestrei ca datele să fie procesate si sistemul să ofere un răspuns relevant.

O altă provocare a acestor sisteme este capabilitatea lor de a integra date deja memorate cu date din streaming. De exemplu într-o aplicație de detecție a cardurilor sau a activităților anormale în mediul online, este necesară utilizarea datelor deja memorate despre aceste activități pentru a compara șabloanele existente cu aceste activități în timp real. Deci, un sistem de procesare streaming trebuie să aibă capabilitatea de a stoca, accesa și modifica informații în mod eficient și de a le combina cu date din stream.

Aceste sisteme trebuie să asigure atât faptul că aplicațiile sunt disponibile cât și integritatea datelor, indiferent de eșecurile prin care sistemul trece.

Operația de distribuție a datelor este din ce în ce mai împortantă datorită volumului mare de date, așadar aceste sisteme oferă mecanisme de distribuție automata a datelor pe mai multe mașini pentru a asigura scalabilitate. Această distribuție trebuie să fie realizată în mod automat și transparent, iar în mod ideal ar trebui eliminata necesitatea intervenției unui programator.

Toate aceste caracteristici ale sistemelor de procesare streaming sunt completate de necesitatea procesării cantităților de date din ce în ce mai mari cu o latență cât mai scăzută. Execuția trebuie optimizată si eliminate posibilele supraîncărcări pentru a livra un răspuns real time într-un timp cât mai scurt.

3.2.4. Arhitectura Lambda

Arhitectura Lambda prezentată în [10] este un șablon arhitectural care combină procesul lent bazat pe procesarea in batch-uri cu elementele ale procesării streaming pentru a depăși provocările BigData legate de velocitatea datelor și volumul lor.
Aceasta este împarțită în trei layere : layerul de persistență asigură stocarea datelor într-un layer precum HDFS care este preluat si prelucrat de un layer batch periodic, la intervale de timp stabilite. Layerul pentru viteză preia partițiile de date care nu sunt procesate încă de layerul batch, iar layerul de servire combină raspunsul layerului batch și a celui de viteză. În figura 3.6 este prezentată realizarea arhitecturii Lambda.

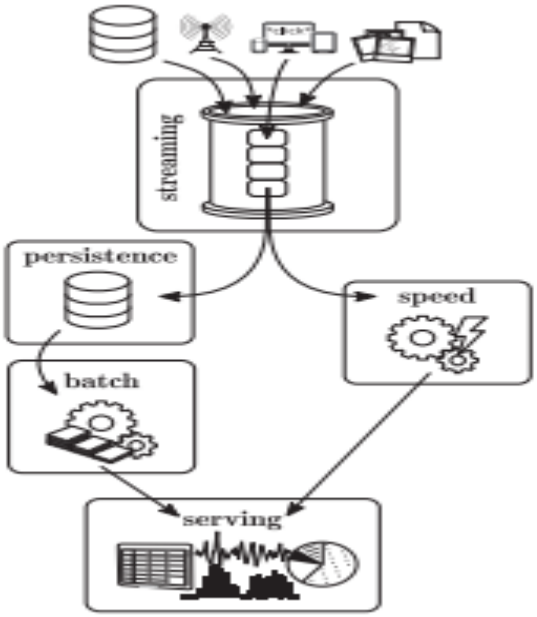

Fig 3.6 Arhitectura Lambda (Sursa : [11])

3.2.5. Procesarea Streaming vs Modelul MapReduce

Astfel, atât procesarea streaming cât și cea în bacth-uri au atât avantaje cât și dezavantaje. Dacă procesarea în timp real reduce latența de prelucarea a datelor, acestea sunt disponibile sistemului doar pe perioada unei ferestre de timp predefinte. După ce perioada de timp este consumată, datele nu vor fi disponibile utilizatorului. Dacă se dorește păstrarea datelor procesate o perioadă mai mare de timp, procesarea batch oferă acest lucru și analizarea unei cantități mare de date, însa latența va fi una mai mare. Aceste avantaje și dezavantaje sunt încorporate în arhitectura Lambda care implementează cele două metode.

Astfel, alegerea unei metode este datorată necesității sistemului, iar în cazul în care utilizatorii au nevoie de un răspuns prompt, imediat, se va alege procesarea streaming. În cazul în care utilizatorii au nevoie de răspunsul procesării datelor la intervale de timp, cum ar fi zilnic, saptămânal, se va alege procesarea batch.

Capitolul 4. Analiză şi Fundamentare Teoretică

Acest capitol are rolul de a prezenta scenariile existente în sistem care duc la îndeplinirea obiectivelor. Totodată se prezintă și tehnologiile și framework-urile Apache care fac posibila implementarea acestora.

4.1. Scenariile sistemului

Pentru a pune în valoare analiza inteligenței business sistemul va prezice părerile utilizatorilor rețelei sociale Twitter despre dispozitive mobile care aparțin de 4 brand-uri cunoscute: Apple, Samsung, Huawei și LG. S-a decis alegerea acestora deoarece fac parte din primele 6 locuri ale Top Mobile Brands in World 2017 [4]. Scenariile existente se mapează pe pașii necesari în realizarea unui sistem de clasificare prezentați anterior.

Astfel, primul pas este antrenarea sistemului. Pentru a realiza acest lucru, după ce setul de date de intrare este creat, acesta va fi pre-procesat și transmis mai departe spre un clasificator Naive Bayes care fac parte din librările Apache Spark Mllib. Acest scenariu care reprezintă etapa de antrenare este prezentat în Figura 4.1. După cum se poate observa în figura, înainte ca datele să fie pre-procesate acestea sunt date text, string-uri, iar în urma pre-procesării ele vor deveni un vector de valori reale pentru a putea fi înțelese de către clasificator.

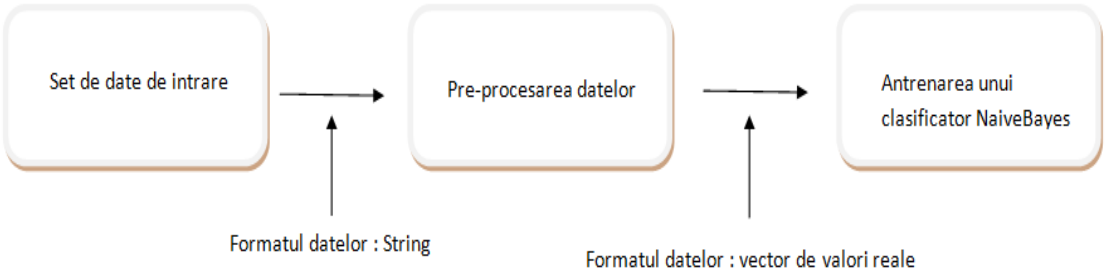

Figura 4.1 Scenariul pentru etapa de antrenare a sistemului

După ce streaming-ul de date este creat pentru fiecare brand în parte, tweet-urile sunt publicate pe o coadă de mesagerie Apache Kafka. Pentru a putea gestiona aceste cozi de mesage este necesară instalarea și pornirea unui server Apache Zookeper. După postarea lor pe coadă cu ajutorul unui producer Kafka, un consumer Kafka le va consuma și le va trimite mai departe spre pre-procesare. O dată pre-proceaste datele sunt transmise unui clasificator Naive Bayes care fac parte din librările Apache Spark Mllib astfel că sistemul va fi antrenat.

[4] https://www.mbaskool.com/fun-corner/top-brand-lists/17188-top-10-global-mobile-phone-brands-in-2017.html

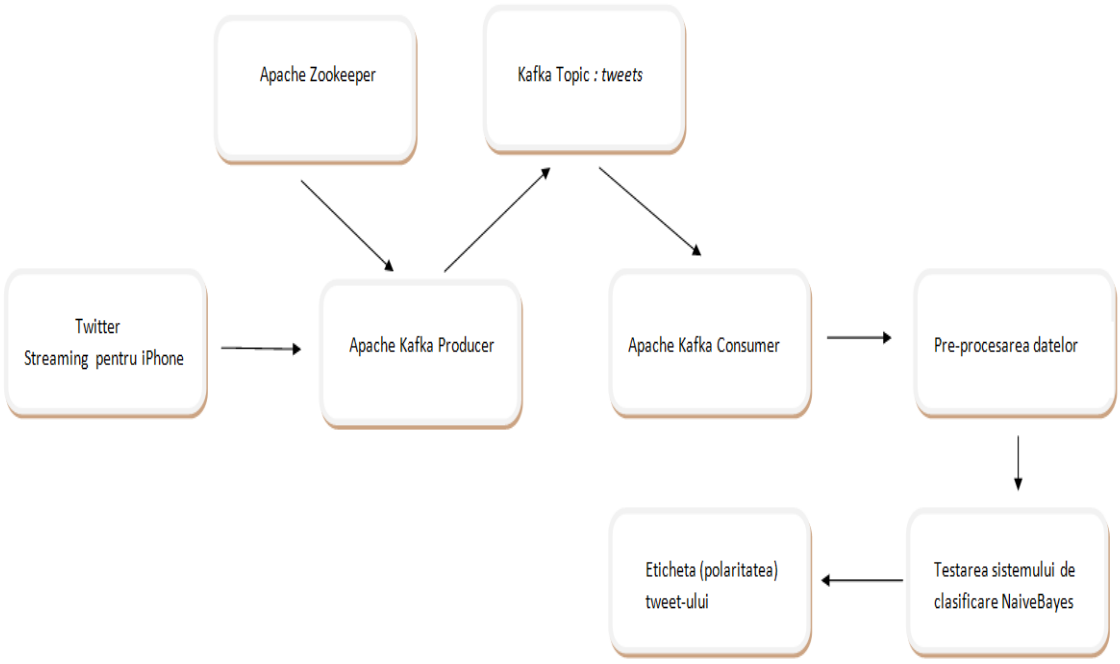

Figura 4.2 Scenariul sistemului pentru testarea brand-ului Apple

În Figura 4.2 se prezintă scenariul de testare pentru dispozitivle mobile care aparțin brand-ului Apple, iar acest scenariu este sistem și pentru celelalte brand-uri : Samsung, Huawei și LG.

Pentru ca acest sistem să fie util unor interprinderi de dimensiuni mici și mijlocii este necesară introducerea în sistem a unui nou scenariu, în care utilizatorul sistemului introduce un nou set de cuvinte după care se va face filtrarea tweet-urilor. Astfel, cu costuri mai reduse fiecare companie poate beneficia de avantajele acestui sistem prin introducerea cuvintelor cheie specifice lor și oferă posibilitatea viitorilor clienți să își facă o părere despre brand-ul pe care îl oferă. Acest scenariu este prezentat în figura 4.3

Cele șase scenarii prezentate sunt construite în jurul unu cluster Kafka format dintr-un singur produce și un singur consumer. Acest lucru este configurat prin intermediul Apache Zookeeper care se poate identifica în scenarii. În cazul unui flux foarte mare de date, Zookeeper oferă posibilitatea de a configura cluster-ul astfel încât să existe mai mulți consumatori, lucru care va evita o posibilă supra-încărcare a sistemului.

Dacă sistemul se va confrunta cu un flux prea mare de date și va fi configurat un singur consumer, datele nu vor fi pierdute, însă vor aștepta pe coada de mesaje până când vor fi consumate, lucru care va introduce o latență în sistem și răspunsul nu va fi unul în timp real. În figura 4.3 va fi prezentat un scenariu în care există trei consumeri Kafka pentru a face față unui flux mare de date.

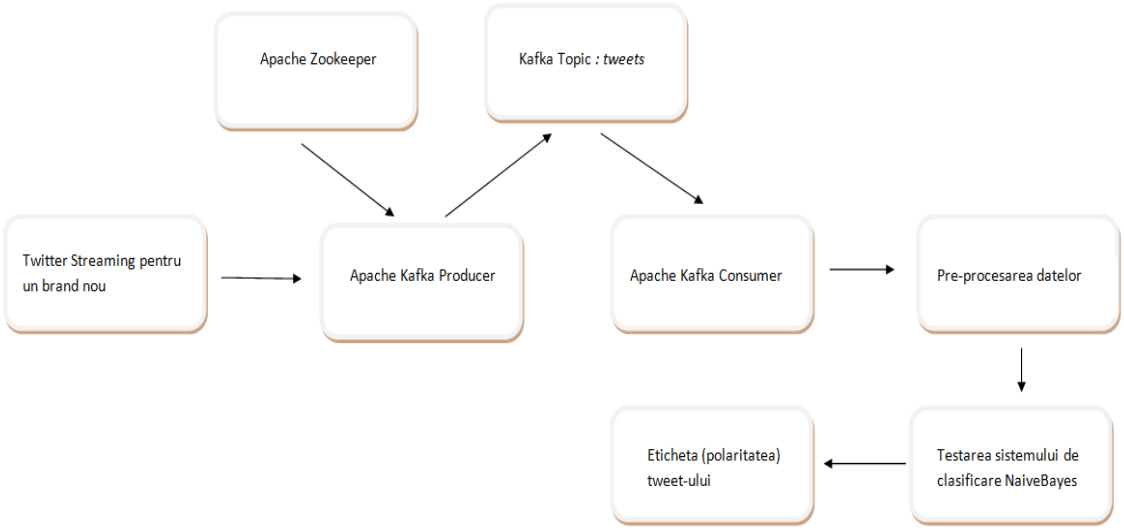

Figura 4.3 Scenariul sistemului pentru testarea unui brand nou

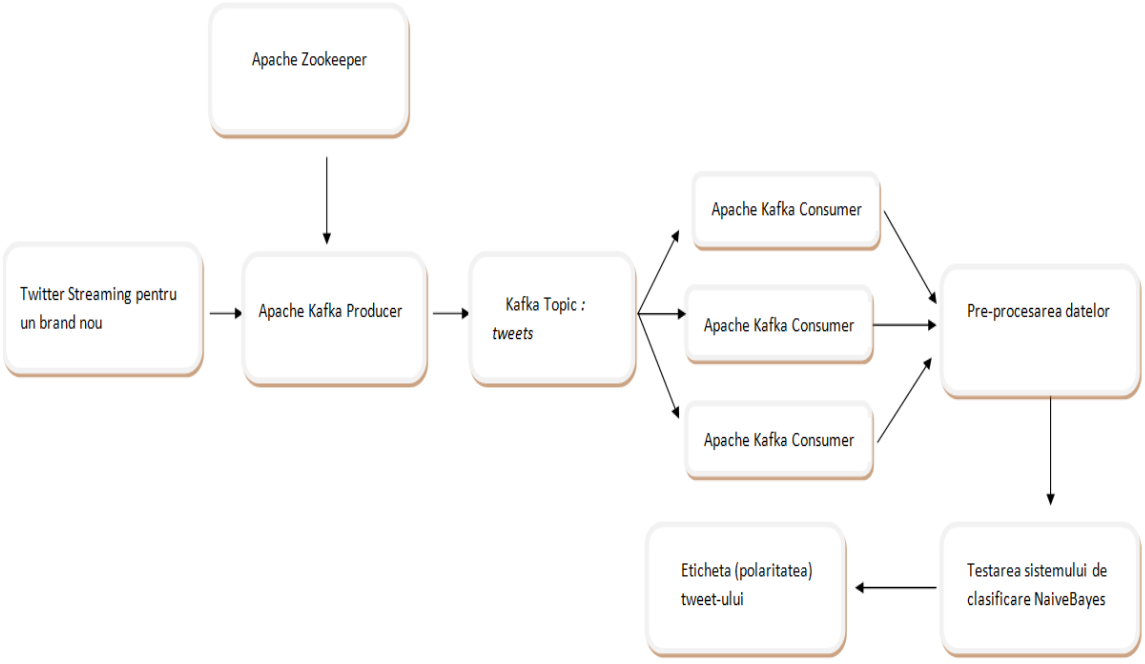

Figura 4.4 Scenariul sistemului pentru testarea unui brand nou cu trei consumeri configurați

4.2. Perspectivă tehnologică

4.2.1. Apache Spark

Apache Spark prezentat în [12] a fost conceput pentru procesarea de date pe scară largă. Este un framework care a apărut ca o îmbunătățire a Hadoop MapReduce, iar autorii acestuia pretind ca depășește performanțele Hadoop de 100 de ori în memorie și de 10 ori pe disc. Acesta extinde modelul popular MapReduce pentru a oferi suport mult mai eficient în diferite tipuri de calcul, interogări interactive și procesare streaming.
A fost proiectat pentru a fi ușor accesibil și oferă API-uri simple în limbaje de programare precum Java, Python,Scala si SQL și poate fi ușor integrat cu alte unelte din zona BigData.

Spre deosebire de implementările MapReduce din framework-urile precedente, Apache Spark aduce în plus cinci beneficii importante :

1. **Viteza foarte mare de calcul** deoarece datele sunt încărcate în memoria distribuită RAM peste un cluster de mașini fizice. Ulterior, datele sunt transformate rapid pentru a fi iterate și sunt cache-uite pentru a fi utilizate ulterior. După cum s-a precizat, Apache Spark procesează datele de 100 de ori mai repede decât Hadoop MapReduce în cazul în care datele se regăsesc în memorie. În cazul în care datele sunt stocate pe disc din lipsa memoriei RAM perfomanța Apache Spark este de 10 ori mai mare. Această caracteristică este evidențită în Figura 4.5

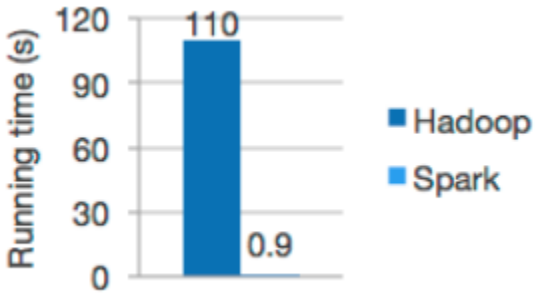

Figura 4.5 Viteza Spark vs viteza Hadoop[5]

2. **Accesibilitatea** data de API-urile standard puse la dispoziție utilizatorilor care sunt construite în diverse limbaje de programare cum ar fi : Java, Scala, Python, SQL, R. Totodată, Spark oferă un set bogat de librării pentru dezvoltarea

[5] https://spark.apache.org/

aplicațiilor de tip Machine Learning, avantaj relevant pentru contextul acestui proiect.

3. **Compatibilitatea** cu ecosistemele existente deja pe piață cum ar fi Hadoop v1 și 2.x YARN, lucru care oferă posibilitatea companiilor de a-și valorifica infrastructura deja implementată.

4. Apache Spark oferă un proces de descărcare și instalare **convenabil**, astfel existând un shell interctiv pentru a învăta funcționalitățile acestui framework.

5. **Creșterea productivității** deoarece este construit în așa manieră încât se menține atenția asupra conținutului și puterii de calcul.

Există cateva cuvinte cheie care sunt necesare pentru a întelege funcționale Apache Spark :

- **Job** : este un calcul paralel care citește datele de intrare și execută unele calcule asupra datelor

- **Task** : fiecare etapă are câteva task-uri, câte unul pentru fiecare partiție. Un task este executat pe o partiție pe un executor (mașină)

- **Driver** : procesul resonsabil pentru a rula un job pe motorul Spark

- **Master** : mașina pe care se rulează programul driver.

- **Slave** : mașina pe care se rulează programele executor.

- **Stages** : fiecare job este împarțit în etape. Acestea sunt divizate în etape map sau reduce.

- **Executor** : procesul responsabil pentru executarea task-ului. Numărul de executori folosiți este direct proporțional cu cantitatea de timp necesară unui job să fie executat. Folosirea corectă a numărului de executori este prezentată în Figura 4.6

Proiectul Spark conține mai multe componente strâns integrate. În esență Spark este un mecanism computațional care este responsabil pentru programarea, distribuția și monitorizarea aplicațiilor intens computaționale de pe mai multe mașini slave sau dintr-un cluster. Componentele sunt : Spark Core, Spark SQL, Spark Streaming, Mlib și GraphX. Aceste componente sunt descrise în figura 4.7 și vor fi prezentate în continuare.

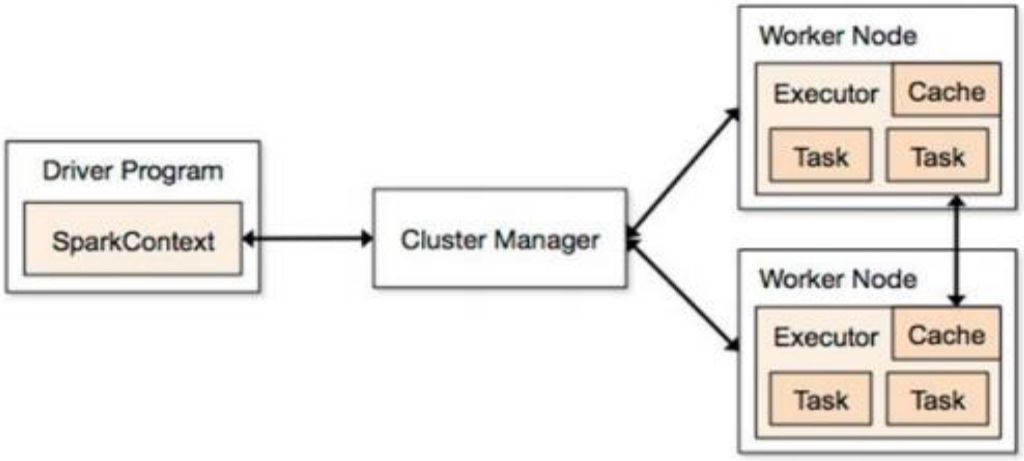

Figura 4.6 Prezentarea unui cluster Spark[6]

Spark Core conține funcționalitățile de bază ale Spark și include componente pentru planificarea sarcinilor, managmentul memoriei, recuperarea la eșec, interacțiunea cu sisteme de stocare. Totodată acesta conține API-ul care **definește *Resilient Distributed Datasets* (RDDs)** care reprezintă o colecție de elemente disitribuite pe diferite noduri de calcul care pot fi manipulate în paralel. Spark Core oferă API-uri care definesc, construiesc și manipulează colecțiile RDD.

Toată munca realizată de Spark este în jurul RDD-urilor fie că este vorba despre crearea unor RDD-uri noi, transformarea celor existente sau apelul operațiilor pe acestea pentru a calcula un rezultat. Spark distribuie automat datele conținute în RDD-uri în cluster și paralelizează operațiile care sunt executate asupra lor.

Fiecare RDD este divizat în multiple partiții care pot fi calculate pe noduri diferite în cluster. O dată create ele oferă două tipuri de operații : transformări și acțiuni. Transformările sunt operații care returnează un nou RDD cum ar fi *map()* și *filter()*, iar

acțiunile sunt operații care returnează un rezultat cum ar fi *count()* și *first()*. Transformările pe RDD-uri sunt evaluate "leneș" ceea ce înseamnă că Spark nu le va executa până când va vedea și o acțiune. Cu alte cuvinte, când se apelează o transformare pe un RDD, aceasta nu este imediat executată decât atunci când aceasta este utilizată.

În esență RDD reprezintă o memorie abstract care reduce căutarea pe disk si tolerează erorile. Cu ajutorul acestora, datele sunt stocate în memorie și sunt mai ușor accesibile în cazul reutilizării acestora. Deoarece sunt tolerante la erori, în cazul în care acestea apar RDD-urile au capabilitata de a reconstrui datele. Datorită unor restricții privind folosirea memoriei partajate, au fost concepute cu două proprietăți: sunt partiționate și read-only.

[6] http://spark.apache.org/docs/1.3.0/cluster-overview.html

Pot exista situații în care se cere păstrarea efectivă a unor partiții, iar atunci motorul Spark stochează implicit în memorie. Totuși, în cazul în care utilizatorii socilită memorarea pe hard-disk sau nu mai este suficient spațiu în memorie Spark va stoca RDD-uriile pe hard-disk.

Fiecare RDD are mai multe elemente: un set de partiții, un set de dependințe spre RDD-ul părinte, o funcție pentru calcul și metadata despre schema de partiționare. Funcția pentru calcul definește modul în care un RDD rezultă din RDD-ul părinte, iar metadatele conțin informații despre locație si număr de blocuri ale fișierului HDFS. Există două moduri de a crea un RDD și anume din alte RDD-uri existene sau printr-un set de date existene în spațiul de stocare stabil. Primul mod de creare prezentat facilitează reconstrucția RDD-urilor pierdute deoarece fiecare RDD conține suficiente informații despre alte RDD-uri și le pot reconstrui. Dependințele către părinte sunt de două feluri : narrow (înguste) și wide (largi). Dependințele sunt înguste atunci când partițiile copilului depind de un număr fix de partiții din parinte, iar un exemplu în acest sens sunt transformările de tip map. Dependințele sunt largi atunci când fiecare partiție poate depinde de toate partițiile din părintele RDD, exemplu poate fi transformarea groupByKey. Această clasificare este utilă în îmbunătățirea procesului de execuție și recuperare. În cazul unui eșec este de preferat ca dependența să fie îngustă deoarece Spark care capabilitatea de a recompune un număr fix de partiții eșuate, iar recompunerea lor are loc în paralel pe noduri diferite.

Apache Spark prin planificatorul de sarcini folosește strucurile RDD-urilor pentru a optimiza execuția prin construirea unui graf aciclic direct pentru etapele de calcul pentru fiecare RDD. Planificarea este realizată în așa fel încât fiecare etapă să conțină cât mai multe informații posibile cu dependințe înguste. Sarcinile sunt plasate de către planificator în funcție de localizarea datelor astfel încât comunicarea în rețea să fie minimizată.

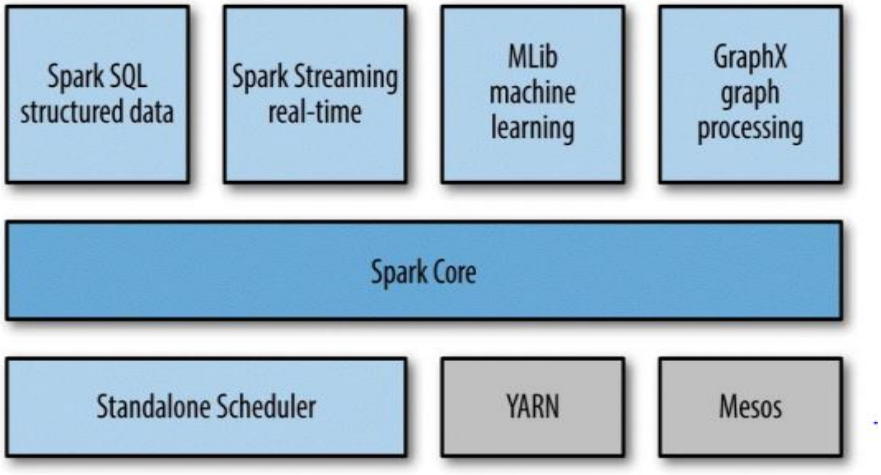

Figura 4.7 Arhitecutra Spark (Sursa : [12])

Spark SQL este pachetul integrat in Spark pentru a lucra cu datele structurate. Acesta ofera o interfață pentru a interacționa cu Spark prin limbaje precum SQL sau HQL(Hive Query Language)

Spark Streaming permite procesarea stream-urilor în timp real. Permite aceleași funcționalități precum Spark Core în ceea ce privește toleranța la erori și scalabilitatea. Datele pot fi inserate în Spark Streaming din diferite surse cum ar fi : Kafka, Flume, HDFS, Kinesis, Twitter. Acesta permite încărcarea în sistem a unor cantități mari de date care vor fi apoi distribuite pe mai multe noduri și procesate în paralel. Datele o dată intrate în sistem sunt prelucrate utilizând funcții de nivel înalt precum : *map()*, *reduce()*, *window()* sau *join()*. Stream-urile primite sunt împărțite în mai multe secțiuni sau batch-uri, care apoi sunt transmise la motorul Apache Spark.

Principalele avantaje ale Spark Streaming sunt : este scalabil, ajunge la latențe la scară mai mică, este integrat cu procesarea batch și interactive , are un model simplu de programare și este foarte eficient în ceea ce privește toleranța la eșec. În mod intern, Spark Streaming primește streamuri de date pe care le împarte în batch-uri. Dupa împărțire, datele sunt procesate de către Spark Core care va genera stream-ul final în batch-uri. Acest mod de procesare a datelor este prezentat în figura 4.8.

Figura 4.8 Modul de procesare a datelor de către Spark Streaming[7]

Apache Spark Streaming are diferite caracteristici : datorită RDD-urilor datele pot fi recalculate în cazul unui eșec. Caracteristica throuhput asigură o perfomanță ridicată deoarece oferă posibilitatea de a primi stream-uri în paralel. Scalabilitatea poate fi oferită deoarece Spark execută sarcinile pe mai multe noduri care sunt conectate între ele și lucrează în paralel pentru a echilibra procesarea.

Apache Spark Streaming consideră fiecare batch de date ca fiind un RDD. Configurarea acestui interval de timp este foarte importantă deoarece dacă intervalul este prea mic datele nu vor fi suficiente, iar rezultatele nu vor fi relevante pentru utilizator. Fiecare batch, la intervalul de timp specificat, va creea un nou Resilient Distributed Dataset care este procesat, iar rezultatul procesării va fi un nou RDD. Această arhitectura a streaming-ului de date este prezentată în Figura 4.9

[7] https://spark.apache.org/docs/1.3.0/streaming-programming-guide.html#linking

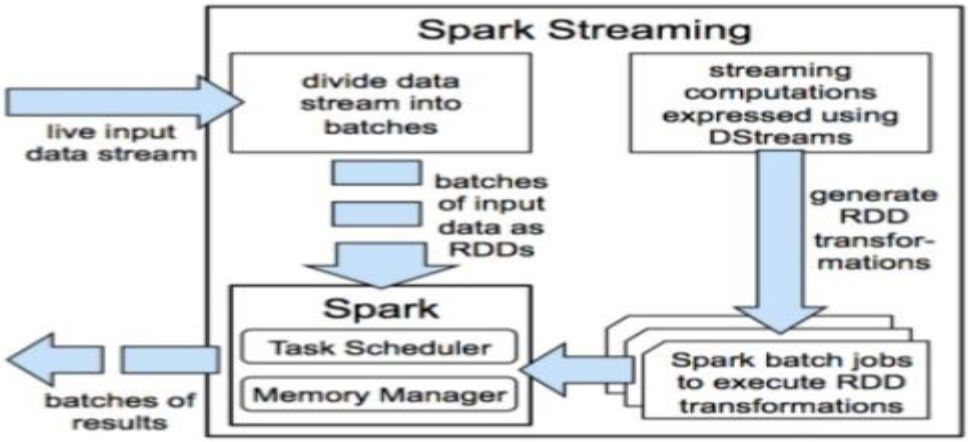

Figura 4.9 Arhitectura Spark Streaming

Spark Streaming oferă o abstracție de nivel înalt numită discritezed stream sau Dstream care repezină streamurile continue de date. Aceastea pot fi create fie din datele de intrare care provin din diferite surse (Kafka, Flume, Kinesis, etc.) sau din alte Dstreams care intern sunt reprezentate ca o secvență de RDD-uri care au fost prezentate anterior. Datorită faptului ca Dstream-urile conțin RDD-uri face posibilă ca Spark Streaming să aibă disponibile o serie de operații asupra Dstream-urilor. Aceste operații sunt similare celor care se aplică asupra RDD-urilor.

Tehnica abordată de Spark pentru a lucra cu aceste stream-uri este combinarea mai multor înregistrări în mini-batch-uri. Acestea pot fi delimitatea fie prin intervalul de timp fie prin numărul de înregistrări dint-un batch. Pnetru a crea un Dstream este necesar să se definească un timp pentru mini-batch-uri. În acest interval datele de intrare din Dstream vor deveni RDD-uri. Dacă nu există date în intervalul specificat , RDD-ul va fi gol.

Pentru a recupera datele se verifică period starea RDD-urilor, iar apoi punctele de control sunt replicate către noduri diferite. În cazul în care un nod eșuează, recuperarea detectează partițiile pierdute și lansează sarcina de recuperare de la ultimul punct de control.

Spark MLlib Libraria machine leaning este unul dintre cele mai importante unelte ale Apache Spark, iar aceasta conține algoritmii pentru învățarea asistată de calculator pe un cluster. Această librărie poate fi folosită și din alte librării. Acesta conține capabilități comune din zona de machine learning precum : regresie, clasificare binară, filtre colaborative. Acesta are un design și o arhitectură simplă în sensul în care permite invocarea algortimilor pe seturi de date distribuite , reprezentate de RDD-uri.

GraphXeste o librărie distribuită care extinde API-ul pentru Spark RDD. Acesta oferă un API pentru a manipula grafurile și de a exmprima diferitele calcule pe aceste grafuri dar oferă și o librărie care conține algoritmi comuni pentru grafuri.

Transformările existene în Spark sunt funcții care produc un RDD nou din RDD-uri existente. Fiecare funcție primește ca intrare unul sau mai multe RDD-uri și returnează unul sau mai multe RDD-uri. În general, aceste transformări sunt lente și executate atunci când se face apel la o acțiune, dar nu este o regulă ca acestea să fie executate imediat. După transformare este creat un nou RDD care poate avea dimensiuni diferite față de cel inițial în funcție de ce funcții sunt aplicate. În cazul în care sunt aplicate transformări precum *filter(), count(), distinct(), sample(),* RDD-ul rezultat va fi mai mic, iar în cazul în care sunt aplicate transformări precum *flatMap(), union(), cartesian(),* RDD-ul rezultat va fi mai mare sau de aceeași dimensiune. Există două tipuri de transfromări :

- **Transformări narrow** : toate elementele necesare în calculul înregistrărilor în partiția unica există în partiția unuică a RDD-ului părinte. Doar un subset limitat de partiție este folosit pentru a calcula rezultatul. Din cadrul acestor transformări fac parte funcțiile *map() și filter().*
- **Transformări wide** : toate elementele necesare pentru a cacula înregistrările din partiția unică pot trăi în mai multe partiții ale RDD-ului părinte. Din cadrul acestor transformări fac parte funcțiile *groupByKey()*și *reduceByKey().*

Acțiunile sunt utile deoarece prin executarea lor se declanșează calculul transformărilor prezentate anterior. Acțiunea instruiește Spark să calculeze rezultatul dintr-o serie de transformări. Printre exemplele existente în Spark amintim : *count(), collect(), take(n), top(), reduce(), fold(), aggregate(), foreach().*

Extragerea caracteristicilor TF-IDF (Term frequency – inverse document frequency) este o metodă implementată în Apache Spark care returnează un vector de caracteristici pentru a indica importanța unui termen în corpusul unui dicționar. Aceasta este extrem de relevantă în contextul acestui proiect deoarece fiecare tweet trebuie transformat într-un vector de valori double, reprezentând datele de test pentru clasificator, pentru a putea fi prezisă polaritatea acestuia. Pentru a prezenta beneficiile acestei metode considerăm că avem un termen t, un document d și un corpus D. Frecvența termenilor (Term frequency) TF(t,d) reprezintă de câte ori termenul t apare în documentul d, iar frecvența documentelor(Document frequency) DF(t,D) reprezină numărul documentelor care conțin termenul t.

În cazul în care am utiliza doar frecvența termenilor pentru a măsura importanța unor termeni, se ajunge ușor într-o zonă de eroare deoarece exisă mulți termeni precum a , the sau of care apar foarte des în documente, dar care nu conțin informație relevantă. O soluție pentru această problemă este frecvența inversului unui document (inverse document frequency) care este o unitate de măsură numerică care indică nivelul de informație pe care un termen o oferă. Aceasta se notează cu IDF(t,D) și se calculează utilizând formula :

$$IDF(t,D) = \log \frac{|D|+1}{DF(t,D)+1}$$ [8],

unde D este numărul total de documente care sunt conținute în corpus. Deoarece în calcularea acestui termen se utilizează funcția algoritm, în cazul în care un termen apare în toate documentele, valoarea IDF va deveni 0. Masurarea TF-IDF este calculată simplu, fiind rezultatul produsului dintre TF și IDF, astfel :

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D).$$

4.2.2. Apache Kafka

Kafka este un serviciu pentru distribuire, partiționare și replicare. El oferta utilizatorului funcționalitățile unui sistem de mesagerie. Kafka este utilizat ca o soluție pentru sistemele care au o mulțime de surse pentru date si sistemul devine foarte impractic și ineficient. Pentru a elimina aceste complicații , Kafka va utiliza un sistem producer-cosnumer de mesagerie.

Există câteva cuvinte cheie în sistemele de mesagerie :

- Topic : reprezintă categorii unde Kafka păstrează mesajele.
- Producer : procesul care postează mesajele pe topicul Kafka.
- Subscrieber : procesul care preia mesajele de pe topicul Ksfka.
- Broker : serverul care cuprinde clusterul Kafka.

Arhitectura Kafka care include aceste elemente este prezentată în figura 4.10

Topicurile reprezintă categoria în care mesajele sunt publicate. Dacă am luat în considerarea contextul lucrării putem considera un topic precum un hashtag de pe Twitter și toate tweet-urile cu anumite hastag-uri să fie incluse într-un topic. Subscrieberii se pot abona la aceste topicuri și vor primi mesajele publicate doar în topicurile de interes pentru aceștia.

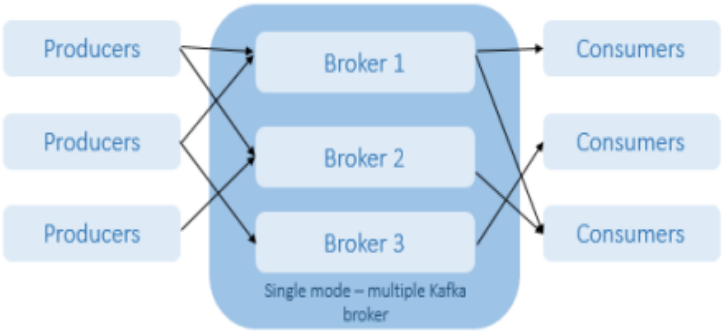

Figura 4.10 Arhitectura Kafka [9]

[8] https://spark.apache.org/docs/latest/ml-features.html#tf-idf-hashingtf-and-idf

Pentru fiecare topic Kafka mențione un log de partiții. Fiecare partiție e ca o secvență de mesaje ordonată și imutabilă. Fiecare mesaj din partiție este asignat unui id asignat în mod secvențial care se numește offset și identifică în mod unic fiecare mesaj din ce partiție face parte. Toate aceste mesaje sunt reținute într-un cluster Kafka pentru o perioadă de timp care poate fi configurată, chiar dacă ele nu sunt consumate. Concepul nou care apare lucrând cu aceste tipuri de partiții este offsetul. Aceasta este singura metadată salvată pentru cosumator și îi oferă acestuia posibilitatea de a găsi poziția consumatorlui în logul de mesaje. Offsetul este constrolat de către consumator, iar în acest fel el poate consuma mesajele în orice ordine prin resetarea la un offset mai vechi sau trecerea peste offsetul curent. Acest control oferit utilizatorului aduce valoare prin flexibilitatea pe care acesta o are de a reprocesa date din trecut sau pentru a trece peste înregistrările curente.

Producers sunt responsabili pentru a publica datele în topicurile la care sunt asignați. Aceștia sunt responsabili și de asignarea mesajelor în diferite partiții din diferite topicuri.

Consumers : în sistemele tradiționale de mesagerie un grup de consumatori pot citi mesajele de pe un server și fiecare mesaj merge la unul dintre consumatori. In sistemele publish-subscriebe mesajele sunt transmise tuturor consumatorilor. Kafka oferă o singură abstractizare pentru consumer care generalizează și reprezintă întreg grupul de consumeri. Aceștia se etichetează singuri cu un nume al grupului și fiecare mesaj publicat într-un topic este livrat unei instanțe de consumer din interiorul grupului. Aceste instanțe care formează un grup pot face parte din procese separate sau chiar din mașini separate.

Partițiile în Kafka sunt distribuite pe servere într-un cluser în care fiecare server preia cererile pentru mai multe partiții. Fiecare partiție este replicată pe diferite servere pentru a asigura că un cluster va funcționa chiar și în cazul unui eșec. Fiecare partiție are un server lider și mai multe servere care urmează instrucțiunile serverului lider. Liderul recepționează cererile de tip citire scriere pentru partiții, iar în cazul în care eșuează un server urmăritor ales în mod aleator va deveni noul sever lider.

4.2.3. Apache Zookeeper

Pentru a administra sistemul în aceste situații de eșec este folosit Apache Zookeper. Acesta este o unealtă devoltată de Apache și specializată în administra configurările pentru distributia sincronă. Acesta este utilizat de către Kafka pentru a realiza alegerea liderilor. Zookeeper trimite și ține la curent despre schimbările în topologia Kafka, așadar fiecare nod din cluster știe când un nou broker a fost adăugat, când un broker a fost eliminat, când un topic a fost șters sau adăugat.În figura 4.11 este reprezentat Zookeper integrat cu Kafka.

[9]https://kafka.apache.org/

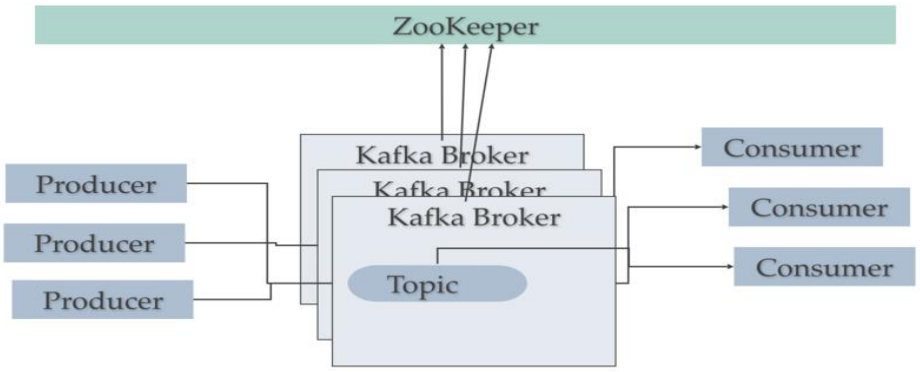

Figura 4.11 Zookeeper integrat cu Kafka [10]

4.2.4. Twitter APIs

Aplicația Twitter a fost creată în anul 2006 iar una dintre principalele caracteristici ale acesteia este faptul că utilizatorii pot posta și citi mesaje care au o lungime maximă de 140 de caractere conform [13].

Interfața aplicației Twitter este mijlocul prin care putem prelua datele dorite ale utilizatorilor de pe rețeaua de socializare. Aceasta oferă dezvoltatorilor de aplicații un acces progrmatic la un set de funcționalități ale rețelei Twitter. API-ul permite integrarea ușoară a funcționalităților Twitter în aplicații persoanlizate.

Conceptul original al API-ului împarte resursele în 3 categorii după funcționalitățile oferite :

- **REST** : oferă acces la datele de pe Twitter, actualizează timeline-ul, statusurile și informațiile despre user.
- **SEARCH** : oferă posibilitatea dezolvatorilor de a interacționa cu motorul de căutare
- **STREAM** : oferă accesul la date în timp real și într-o canitate mare

Începând cu versiunea 1.1 toate aceste servicii sunt unificare într-un singur serviciu sub header-ul REST. Principala diferență dintre API-ul REST si Streaming este faptul că cel streaming oferă acces cu o latență scăzută și necesită menținerea unei conexiuni HTTP. Figurile 4.12 și 4.13 exemplifică modul în care cele două API-uri reacționează în mod diferit la o cerere de tip HTTP.

În cazul API-ului REST cererea HTTP realizează o conexiune la API-ul Twitter, iar în cazul API-ului Streaming procesul HTTP și procesul de streaming vor rula separat. Procesul streaming colectează datele în timp real, le filtrează , le pasează și în final le stochează, urmând ca procesul HTTP sa utilizeze aceste date cand are loc o cerere din partea unui utilizator.

[10] http://cloudurable.com/blog/kafka-architecture/index.html

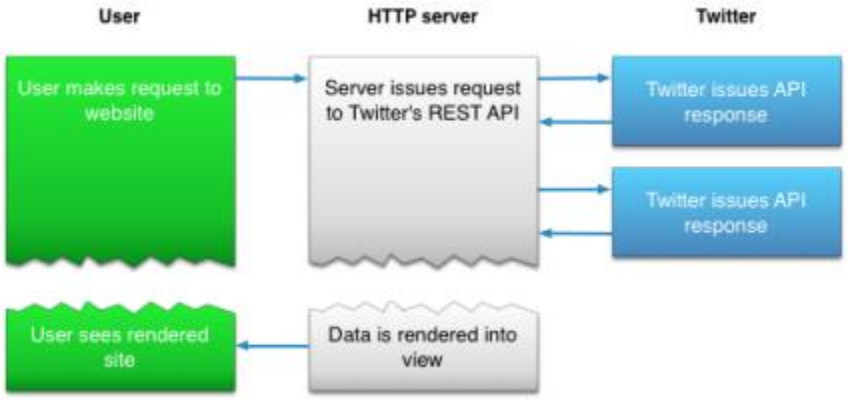

Figura 4.12 Twitter REST API[11]

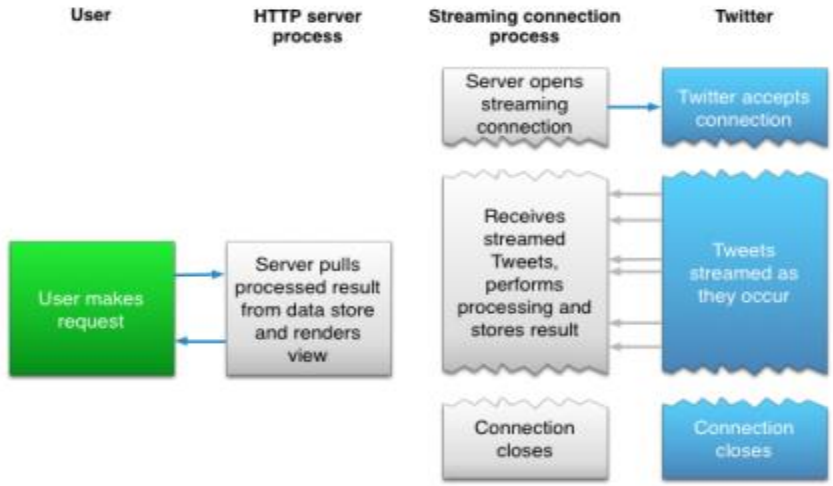

Figura 4.13 Twitter Streaming API [12]

Pentru a prelua doar postările care reprezintă interes pentru aplicația în care Twiter API este utilizat, acesta oferă utilizatorilor posibiliateta ca tweet-urile să fie filtrate după anumiți parametrii [13]. Aceștia sunt în numar de 11 si pot fi utlizați în diferite combinații :

- **Delimited** : setarea acestui parametru indică faptul că statusurile ar trebui să aibă o anumită lungime, astfel acestea vor fi delimitate în stream. Clienții vor ști câți biți să citească înainte ca un mesaj să ajungă la final.

[11] https://dev.twitter.com/rest/public
[12] https://dev.twitter.com/streaming/
[13] https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters.html

Statusurile sunt reprezentate printr-o lungime exprimată în bytes, un caracter pentru linie nouă și textul statusului.

- **Stall_warnings** : prin setarea acestui parametru se vor trimite periodic mesaje de avertizare către client în cazul în care acesta este în pericol să fie deconectat. Acest parametru este mai potrivit și se utilizează în general pentru clienții care au conexiuni cu o lungime mare de bandă.

- **Filter_level** : setarea acestui parametru la “one” sau ”none” , la ”low” sau ”medium” va seta nivelul de filtrare al statusurilor. Implicit acesta este setat la ”none” ceea ce înseamnă că vor fi incluse toate toate tweet-urile disponibile.

- **Language** : acest parametru se poate seta la o listă de limbi după care tweet-urile să fie filtrate.

- **Follow** : reprezintă o listă de id-uri ale utilizatorilor ale căror tweet-uri vor apărea în stream. Răspunsul va conține tweet-uri create de utilizatorii respectivi, tweet-uri repostate de către aceștia, răspunsurile la oricare tweet create de user, dar și răspunsurile manual, adică acelea care au fost create fără a apăsa butonul de reply. Stream-ul nu va conține tweet-urile în care userii sunt menționați sau tweet-urile postate de user protejați.

- **Track** : acest parametru constă într-o listă de fraze care vor determina care Tweet-uri vor apărea în răspuns. O frază poate conține unul sau mai mulți termeni separați prin spațiu, iar o frază va fi valabilă dacă toți termenii din aceșia apar în tweet.

- **Location** : reprezintă o listă de perechi longitudine-latitudine care setează limitele unui tweet în funcție de poziția geografică.

- **Count** : acest parametru poate fi utilizat de utilizatorii cu mai multe drepturi și permite , în cazul unei reconectări, preluarea mesajelor pierdute cât timp conexiunea nu era disponibilă.

4.2.5. Apache Zeppelin

Apache Zeppelin este o unealtă open-source din mediul online care permite analizarea interactivă si vizualizarea datelor din sisteme de procesare a datelor distribuite precum Apache Spark sau Apache Flink [14]. Prima versiune stabilă a acestui proiect a apărut in 2016. Interpretorul Apache Zeppelin [15] permite integrarea cu orice limbaj sau procesare de date. În stadiul current, Zepllein oferă interpretoare precum Scala, Spark,

[14] https://flink.apache.org/
[15] https://zeppelin.apache.org/docs/latest/manual/interpreters.html

Python , Apache HBase[16]. Datorită integrării deja existente cu Apache Spark nu este necesară construirea unor librări suplimentare în cazul în care se dorește utilizarea celor 2 tehnologii. Oferă diferite unelte și diagrame care pot fi utilizate pentru a vizualiza și analiza datele.

[16]https://hbase.apache.org/

Capitolul 5. Proiectare de Detaliu si Implementare

În capitolul curent se va descrie în detaliu soluția pentru implementarea sistemului de analiză a tweet-urilor despre dispozitive mobile, implementare ce are ca fundamente studiul bibliografic cuprins în capitolul trei și fundamentarea teoretică prezentată în capitolul 4. Sistemul implementat procesează tweet-urile despre dispozitive mobile în timp real și analizează conținutul acestora pentru a extrage polaritatea lor.

Obiectivul acestui capitol este descrierea generală a arhitecturii sistemului, dar și a fiecărui modul specific în parte.

5.1. Arhitectura sistemului

5.1.1. Descriere generală

Deoarece sistemul este o aplicație care procesează fluxul de date în timp real, folosind Apache Spark se vor folosi și următoarele componente : SparkSQL, Spark Streaming, Spark Mllib. Între componentele Spark Streaming și Spark Mllib se interpune Apache Kafka prin creearea unui producer care preia fluxul de date din Apache Spark și îl postează pe o coadă,pe topicul numit tweets. Consumerul Kafka preia aceste mesaje și le transmite mai departe spre componenta Apache Mllib pentru a fi procesate. În figura 5.1 se prezintă aceste componente, care integrate împreună redau arhitectura sistemului.

Primul pas este crearea unui cont pentru dezvoltator pe reteaua socială Twitter pentru a se putea realiza autentificare. După autentificare, fluxul de date începe, iar datele sunt împărțite în batch-uri, fiecare batch conținând twitt-urile dintr-un interval de timp configurabil. Apache Spark Streaming consideră fiecare batch de date ca fiind un RDD. Configurarea acestui interval de timp este foarte importantă deoarece dacă intervalul este prea mic datele nu vor fi suficiente, iar rezultatele nu vor fi relevante pentru utilizator. Fiecare batch, la intervalul de timp specificat, va creea un nou Resilient Distributed Dataset care este procesat, iar rezultatul procesării va fi un nou RDD.

Următorul pas este pornirea unui server Zookeeper pentru a gestiona sistemul de mesagerie Apache Kafka. O dată pornit serverul Zookeeper și creat topicul pentru tweet-uri poate începe publicarea datelor provenite din streaming-ul de date pe o coadă de mesaje Kafka de către un producer, iar apoi un consumer le va prelua și le va transmite modulului de pre-procesare. După acest pas tweet-ul nu va mai fi de tip String ci va fi de tipul vector , pregătit pentru a fi testat de clasificator. Înainte de testare, un clasificator NaiveBayes este antrenat cu un set de date predefinit. Implementarea clasificatorului se realizează utilizând libraria Apache Spark Mllib.

Toți acești pași sunt ilustrați în figura 5.1 care conține arhitectura generală a sistemului.

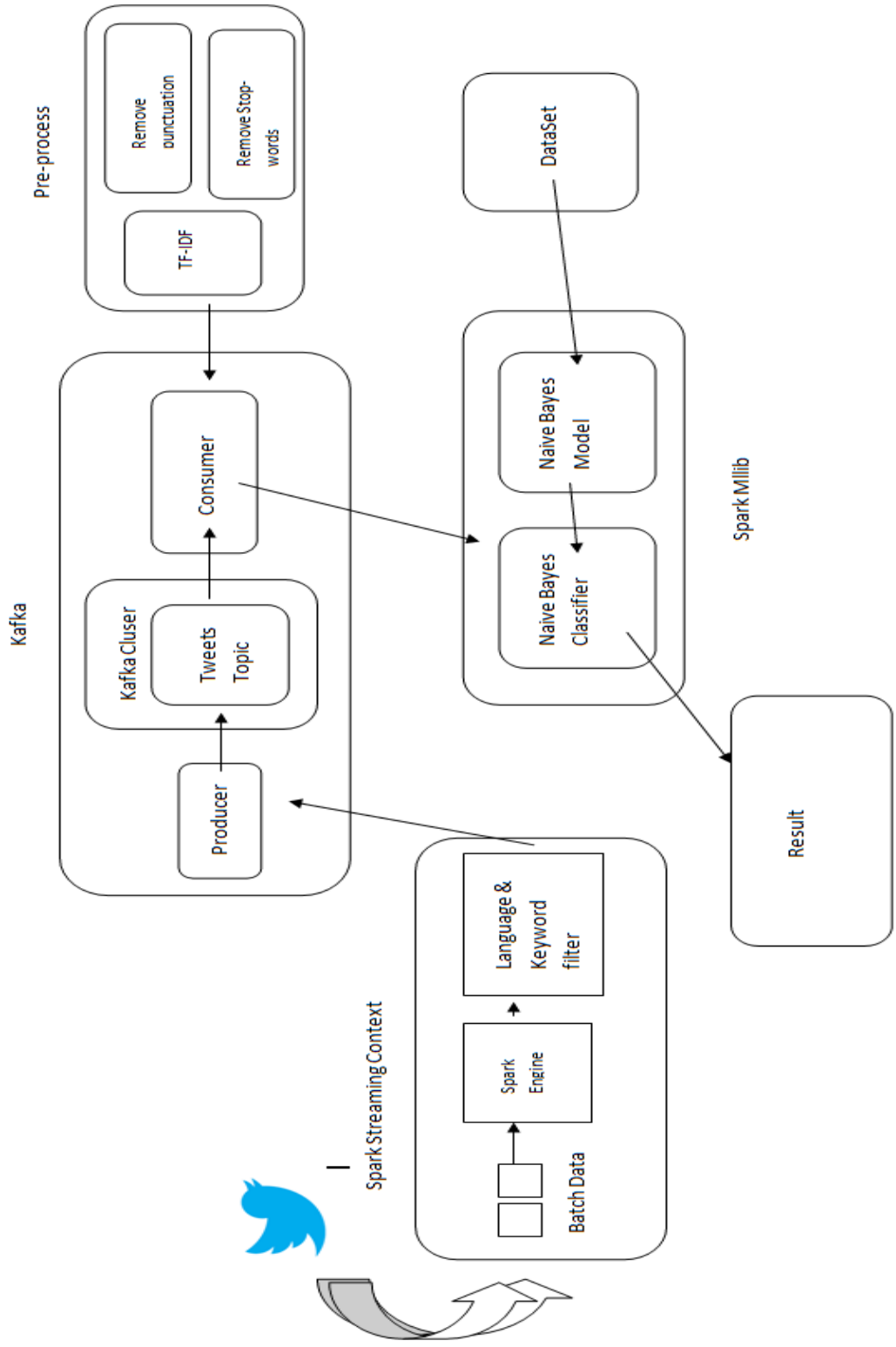

Figura 5.1 Arhitectura generală a sistemului

Toate modulele existente în arhitecutră sunt prezente în implementarea sistemului și sunt reprezentate de câte un pachet Java. Astfel, sistemul conține patru astfel de pachete și sunt prezentate în figura 5.2 :

- Pachetul streaming
- Pachetul kafka
- Pachetul preprocesare
- Pachetul classifier

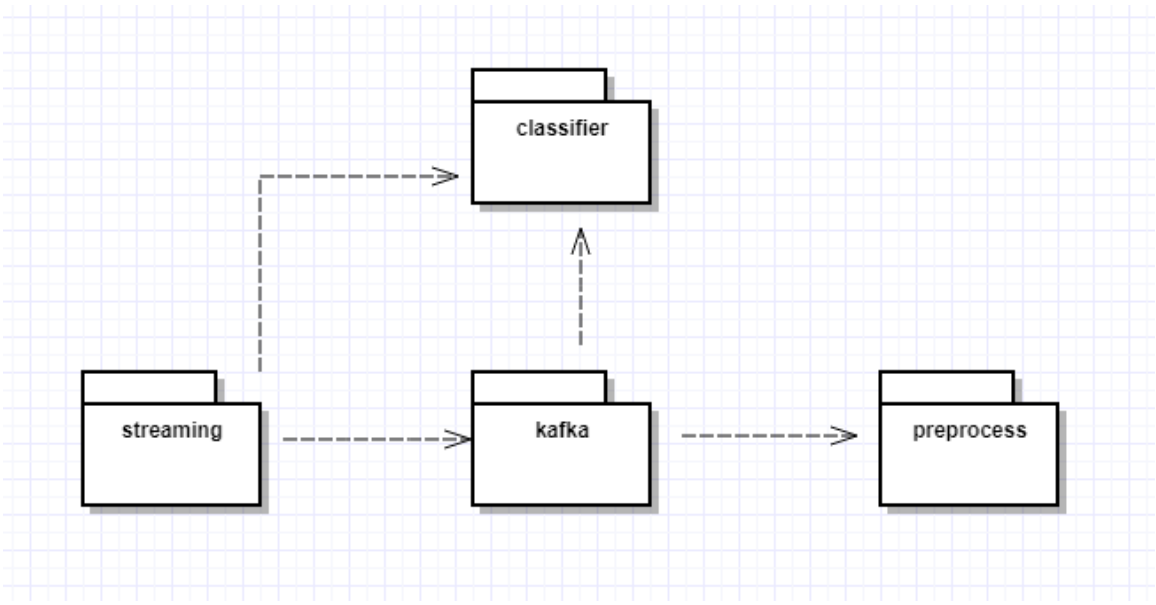

Figura 5.2 Diagrama de pachete a sistemului

Arhitectura Spark Streaming, care este o componentă a arhitecturii generale, în care Apache Spark împarte datele de intrare în RDD-uri care mai apoi sunt salvate în format DStream. Datele DStream sunt formate dintr-o secvență de RDD-uri primite ca intrare în sistemul implementat. În cazul acestui sistem, fluxul de date primit de pe rețeaua de socializare Twitter este transformat în Dstream-uri. Asupra acestora este aplicată funcția filter() care filtrează aceste date în funcție de limbă și cuvinte cheie. Astfel, vor fi repezentate ca RDD-uri într-un rezultat final doare tweet-urile care sunt scrise în limba engleză și care vor conține cuvinte cheie pentru fiecare dispozitiv mobil în parte. Acest lucru este prezentat și în figura 5.3

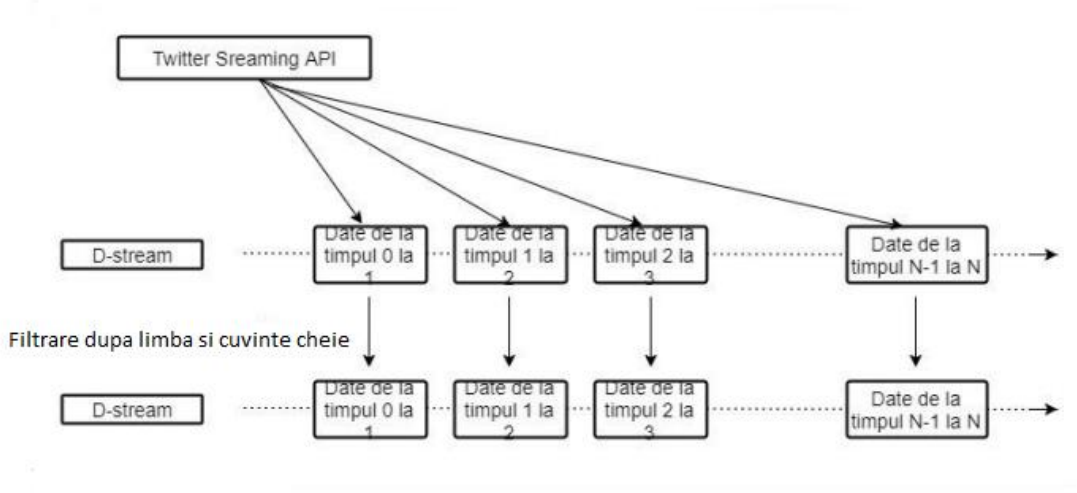

Figura 5.3 Creearea unui RDD dintr-un D-Stream după aplicarea cuvintelor cheie

Pentru a utiliza toate funcționalitățile puse la dispoziție de către Apache Spark, primul pas este creearea unui Spark Streaming Context care permite prin metodele sale de a primi datele din streaming-ul Twitter.

Există două tipuri de transformări care se pot aplica D-Stream-urilor : transformări stateless și transformări stateful. În cazul primei categori de transformări prelucarea batch-urilor individuale nu depinde de rezultatele procesării datelor din batch-urile anterioare. Numeroare funcții fac parte din această categorie printre care : map(), filter(), reduceByKey(). În cazul celei de-a doua categorii de transformări, rezultatele procesării datelor din batch-urile anterioare vor influența calculul rezultatului din batch-ul curent. În cazul implementării acestui sistem se folosesc transformările stateless deoarece funcția filter() este folosită pentru a prelua din întreg fluxul de date doar cele relevante pentru utilizator. După etapa de filtrare este folosită și funcția map() care permite prelucrarea individuală a fiecărui element din D-Stream.

După preluarea datelor în sistem acestea sunt publicate pe o coadă de mesagerie Kafka, pe un topic specific numit tweets. Acestea vor fi apoi consumate și transmise mai departe spre preprocesare. Preprocesarea presupune, în primul rând eliminarea spațiilor, link-urilor, semnelor de punctuație, iar în al doilea rând se vor elimina cuvintele stop-words cum ar fi I,we, me, at , he, is , etc. Dupa această preprocesare tweet-urile vor conține doare cuvintele relevante care vor fi transformate într-un vector de valori double pentru a fi acceptate ca date de intrare pentru clasificator. Astfel, datele sunt pasate spre clasificatorul Naive Bayes care, pentru fiecare tweet va prezice polaritatea acestuia.

5.2. Implementarea arhitecturii

5.2.1. *Modulul Streaming*

Acest modul are rolul de a porni streaming-ul de date și de a publica tweet-urile pe o coadă de mesaje de tipul Apache Kafka, pe un topic creat de utilizatorul, în cazul sistemului cesta numindu-se tweets. Aceast modul are rolul de a oferi sistemului datele de intrare, astfel că tweet-urile reale sunt preluare de pe rețeaua de socializare utilizând API-ul Twitter Stream API.

Twitter oferă utilizatorilor prin API-ul Twitter Stream accesul la datele postate pe rețeaua de socializare. În această manieră se poate prelua ce se postează pe această rețea, însă doar postările cu caracter public.Dintre toate informațiile pe care librările le pune la dispoziție despre un tweet, cum ar fi : id-ul tweet-ului, screenName-ul, text, geoLocation, originalProfilImageUrl, createdAt, în cazul acestui sistem este relevant doar conținutul text al acestuia.

Primul pas în realizarea conecțiunii este creearea unui cont de dezvoltator de aplicații. O dată creat contul se obține datele necesare pentru autentificare :

- **Consumer Key** : cheia consumatorului
- **ConsumerSecret** : cheia secretă a consumatorlui
- **Access Token** : token-ul de acces și parola
- **Secret Access Token** : token-ul de acces secret

Aceste date obținute se vor utiliza ulterior în aplicația Java pentru a stabili conectivitatea dintre sistemul implementat și rețeaua socială Twitter. După stabilirea conexiunii aceasta se va menține deschisă cat timp nu intervin erori la nivelul sistemului, la nivelul rețelei sau încercări multiple de conectare cu aceleași date de autentificare. API-ul oferă o interfață web pentru utilizatori prin intermediul căreia aceștia pot simula diferitele interogări necesare în sistemele pe care le implementează.

În ceea ce privește implementarea sistemului, din punct de vedere al limbajului de programare Java, se utilizează librăria Twitter4J care permite autentificarea la API-ul Twitter dar și alte metode care fac mai ușoare interacționarea cu stream-ul de date. În cazul acestui modul pentru a deservi statusul de date se folosesc următoarele clase din această librările :

- **ConfigurationBuilder** : acestă clasă este folosită pentru a seta conectivitatea cu rețeaua Twitter. Pentru a realiza acest lucru se folosesc metodele : setOAuthConsumerKey(setează cheia consumatorului), setOAuthConsumerSecret(setează cheia secretă a consumatorului), setOAuthAccessToken(setează token-ul de acces) și setOAuthAccessTokenSecret(setează token-ul secret de acces)

- **TwitterStreamFactory** : se creează o clasă de tip Factory utilizând configurațiile anterioare setate în ConfigurationBuilder.

- **FilterQuery** : oferă metode pentru a urmări doar tweet-urile care conțin anumite cuvinte cheie.

- **TwitterStream** : se creează stream-ul de date prin utilizarea metodei getInstance() apelată asupra obiectului TwitterStreamFactory. Asupra twitterStream-ului se va adăuga un listener care va prelua tweet-urile și le va introduce în sistem. Din această clasă se folosește metoda filter() pentru ca rezultul streaming-ului să fie tweet-urile filtrate.

- **StatusListener** : din această clasă se folosește metoda onStatus(Status status) care permite ca fluxul de date să fie introdus în sistem. În mometul în care tweet-urile sunt preluate se verifică ca metoda getLang a obiectului status să fie egală cu "en", astfel vor fi preluate doar statusurile în limba engleză. Această metodă apelează și metoda runProducer() care va fi prezentată și care va posta pe o coadă de mesaje Kakfa tweet-urile.

Deoarece sunt relevante doar mesajele despre dispozitivele mobile care fac parte din brand-urile Iphone, Samsung, Huawei și Lg, în continuare se vor prezenta cuvintele cheie care sunt utilizate pentru filtrarea acestora. Astfel, în Figura 5.4 se vor prezenta cuvintele cheie utilizate pentru Iphone, în Figura 5.5 cele utilizate pentru Samsung, în Figura 5.6 cele utilizate pentru Huawei, iar în Figura 5.7 cele utilizate pentru LG.

```
String[] keywordsArray = { "#iphone", "iph", "iphonex", "iphone x",
        "iphone 8", "#iphone8", "#iphone8plus", "#iphone7",
        "#iphone7plus","#iphoneaccessories", "#iphoneX", "iphone 7",
        "iph 7", "iph8", "iphone 6s", "#iphone6plus", "#iphone7"};
```

Figura 5.4 Cuvinte cheie pentru filtrarea brand-ului Apple

```
String[] keywordsArray = { "#samsung", "samsung", "#samsunggalaxy",
        "#galaxys8", "#galaxys7", "#galaxys7edge","#samsunggalaxys8",
        "#s7edge", "#s8", "#s8plus", "#galaxynote8", "#galaxynote7",
        "#note7", "#note8" };
```

Figura 5.5 Cuvinte cheie pentru filtrarea brand-ului Samsung

```
String[] keywordsArray = { "#huawei", "huaweihonor", "huaweimate",
        "huaweimate10", "p10", "#huaweip10", "#p10lite",
        "#huaweip10lite", "#huaweip8", "#huaweip8lite",
        "#p8lite", "huaweip9", "huaweip9plus", "p9plus",
        "p9lite","#honor9lite", "#nova2s"};
```

Figura 5.6 Cuvinte cheie pentru filtrarea brand-ului Huawei

```
String[] keywordsArray = { "#lg", "#lgphone", "#lgv30",
        "#v30", "#lgq8", "#lgq6", "#lgxadventure","#lgg6",
        "#lgstylues3", "#stylus3", "#lgk10", "lgk8", "lgk4"};
```

Figura 5.7 Cuvinte cheie pentru filtrarea brand-ului LG

Acest modul conține cinci clase care au rolul de a porni streaming-ul pentru fiecare brand în parte și comunică cu clasa KafkaProducerForTweets care va fi prezentată în modulul Kafka. Aceste clase sunt :

- **SearchTweetsIphone**
- **SearchTweetsSamsung**
- **SearchTweetsHuawei**
- **SearchTweetsLg**
- **SearchTweetNewBrand**

5.2.2. Modulul de preprocesare

Deși tweet-urile au avantajul că lungimea lor este una relativ scurtă, de aproximativ 140 de caractere, printre acestea ar putea exista și informație irelevantă pentru sistemul de clasificare cum ar fi semnele de punctuație, hashtag-ul, diferite link-uri. Astfel, modulul de preprocesare are rolul de a elimina din tweet-uri infromația irelavanta și stop-words-urile prin aplicarea unei expresii regulate. Pe langa această atribuție, tweet-urile rezultate din acest modul sunt reprezentate de un vector de valori duble obținut în urma calculării valorii TF-IDF (Term frequency – Inverse document frequency).

Metoda prin care se aplică primul pas al preprocesării este aplicarea unei expresii regulate care să inlocuiască link-uri,hastag-uri, semne de punctuații cu un spațiu alb.Pentru a reduce și mai mult zgomotul din tweet-uri se aplică și procedura eliminării stop-words-urilor. Acest lucru se realizează prin utilizarea unor liste de expresii considerate fără valoare pentru determinarea polarităților în sistemele de clasificare sau de analiză a sentimentelor.

Clasa**TwitterUtilsForPreprocess** conține :

- **Metoda removeSpacesAndPunctuation(String tweet)** care primeste ca parametru conținutul text al tweet-ului, il transformă intr-un string cu caractere mici și elimină toată informația nedorită prin inlocuirea ei cu un spațiu gol.

- **Metoda removeStopWords(String tweet)** primește ca parametru conținutul text al tweet-ului și împarte tweet-ul în mai multe cuvinte prin separarea acestuia când întâlnește un spațiu gol, adică finalul unui cuvânt. Lista de cuvinte din tweet obținută este iterată și se verifică dacă lista de

cuvinte stopWords conține elementele listei de cuvinte. În caz afirmativ, cuvântul respectiv este înlocuit la rândul său cu un spațiu gol. În figura 5.8 este prezentată lista de stop-words-uri care vor fi eliminate

```
public static String[] stopwords = { "a", "as", "able", "about", "above", "according", "accordingly", "across",
        "actually", "after", "afterwards", "again", "against", "aint", "all", "allow", "allows", "almost", "alone",
        "along", "already", "also", "although", "always", "am", "among", "amongst", "an", "and", "another", "any",
        "anybody", "anyhow", "anyone", "anything", "anyway", "anyways", "anywhere", "apart", "appear", "appreciate",
        "appropriate", "are", "arent", "around", "as", "aside", "ask", "asking", "associated", "at", "available",
        "away", "awfully", "be", "became", "because", "become", "becomes", "becoming", "been", "before",
        "beforehand", "behind", "being", "believe", "below", "beside", "besides", "best", "better", "between",
        "beyond", "both", "brief", "but", "by", "cmon", "cs", "came", "can", "cant", "cannot", "cant", "cause",
        "causes", "certain", "certainly", "changes", "clearly", "co", "com", "come", "comes", "concerning",
        "consequently", "consider", "considering", "contain", "containing", "contains", "corresponding", "could",
        "couldnt", "course", "currently", "definitely", "described", "despite", "did", "didnt", "different", "do",
        "does", "doesnt", "doing", "dont", "done", "down", "downwards", "during", "each", "edu", "eg", "eight",
        "either", "else", "elsewhere", "enough", "entirely", "especially", "et", "etc", "even", "ever", "every",
        "everybody", "everyone", "everything", "everywhere", "ex", "exactly", "example", "except", "far", "few",
        "ff", "fifth", "first", "five", "followed", "following", "follows", "for", "former", "formerly", "forth",
        "four", "from", "further", "furthermore", "get", "gets", "getting", "given", "gives", "go", "goes", "going",
        "gone", "got", "gotten", "greetings", "had", "hadnt", "happens", "hardly", "has", "hasnt", "have", "havent",
        "having", "he", "hes", "hello", "help", "hence", "her", "here", "heres", "hereafter", "hereby", "herein",
        "hereupon", "hers", "herself", "hi", "him", "himself", "his", "hither", "hopefully", "how", "howbeit",
        "however", "i", "id", "ill", "im", "ive", "ie", "if", "ignored", "immediate", "in", "inasmuch", "inc",
        "indeed", "indicate", "indicated", "indicates", "inner", "insofar", "instead", "into", "inward", "is",
        "isnt", "it", "itd", "itll", "its", "its", "itself", "just", "keep", "keeps", "kept", "know", "knows",
        "known", "last", "lately", "later", "latter", "latterly", "least", "less", "lest", "let", "lets", "like",
        "liked", "likely", "little", "look", "looking", "looks", "ltd", "mainly", "many", "may", "maybe", "me",
        "mean", "meanwhile", "merely", "might", "more", "moreover", "most", "mostly", "much", "must", "my",
        "myself", "name", "namely", "nd", "near", "nearly", "necessary", "need", "needs", "neither", "never",
```

Figura 5.8 Exemple de cuvinte stop-words

Clasa **HashingTDIDF**conține o singură metodă **tfIdf(SparkSession spark, String tweet)** care primește ca parametru o sesiune Spark care este inițială o singură dată în metoda main pentru rularea proiectului și pasată ca parametru spre toate clasele care o folosesc, pentru a se evita instanțierea multiplă a unei sesiuni. Această metodă primește ca parametru și conținutul text al tweet-ului , care prima dată este transformat într-un obiect List<Row> pentru a putea fi procesat în continuare. Pasul următor este creearea unei scheme care va fi utilizată de tipul StructType care conține două câmpuri – label (pentru polaritatea tweet-ului) și sentence (pentru conținutul tweet-ului) , câmpuri care sunt de tipul StructField.

O dată creată schema se creează un set de date prin apelarea metodei createDataFrame conținută în clasa SparkSession. În aceeași metodă se instanțiază un obiect de tip Tokenizer pentru care se setează coloana de intrare conținutul tweet-ului și coloana de ieșire cuvintele care se extrag din acesta. După creearea obiectului, pentru obținerea efectivă a cuvintelor din tweet se apelează metoda transform de pe obiectul tokenizer care va returna un set de date de tipul DataSet<Row>.

Pasul următor este creearea unui obiect de tipul HashingTF care oferă posibilitatea apelării metodei transform prin intermediul căreia datele obținute în urma aplicării tokenizer-ului sunt transformate în caracteristici. Având la dispoziție datele în formatul corespunzător, trebuie aplicate formulele pentru a afla valoarea numerică a importanței cuvintelor în tweet. Pentru a realiza acest lucru se creează un obiect de tipul

IDF care oferă metoda fit prin intermediul căreia se creează vectorul de valori double necesat testării clasificatorului.

5.2.3. Modulul Kafka

Acest modul are rolul de a introduce în aplicație un sistem de mesagerie astfel încât sa fie asigurat un streaming de date în timp real și oferă posibilitatea configurării numărului de consumer astfel încât să se evite o posibilă supra-încărcare a sistemului datorită unui flux abundent de date.

Înainte de implementarea propriu-zisă a codului, unele configurări pentru funcționalitatea sistemului trebuie rulate din linia de comandă. Din această categorie fac parte următoarele operații :

- Pornirea unui server Zookeeper
- Pornirea unui broker Kafka
- Creearea unui topic Kafka, în cazul sistemului numit tweets

Acest modul conține KafkaProducerForTweets și KafkaConsumerForTweets, două clase care au rolul de a crea și rula producer-ul și consumer-ul sistemului. În cazul în care la pornirea broker-ului Kafka se alege varianta în care se configurează mai mulți consumeri, clasa KafkaConsumerForTweets va fi instanțiată de mai multe ori, nefiind necesară crearea unei clase pentru fiecare consumer configurat.

Clasa **KafkaProducerForTweets** conține :

- O variabilă **TOPIC** care definește topicul utilizat: tweets
- O Variabilă care declară adresa **server-ului boostrap** : localhost:9092
- Metoda **createProducer()** : creează un obiect Properties nou cu ajutorul căruia se configurează detaliile necesare ca producer-ul să poată rula
- Metoda **runProducer(final int sendMessageCount, String message)**: această metodă are rolul de a publica pe topicul tweets mesajele pe care le primește. Mesajele sunt primite spre a fi publicate prin intermediul parametrului message. Acest parametru este transmis în momentul în care metoda este apelată în clasa care creează streaming-ul de date, astfel fiecare mesaj din fluxul de date este transmis spre această metodă

Clasa **KafkaConsumerForTweets** conține :

- O variabilă **TOPIC** care definește topicul utilizat: tweets.

- O Variabilă care declară adresa **server-ului boostrap** : localhost:9092.

- Metoda **createConsumer()** : creează un obiect Properties nou cu ajutorul căruia se configurează detaliile necesare ca producer-ul să poată rula.

- Metoda **runConsumer()** : scopul principal al acestei metode este de a porni un consumer care preia datele de pe topic. La începutul metodei se creează un obiect twitterUtil de tipul TwitterUtilsForPreprocess prezentat anterior, un obiect hashingTDIDF de tipul HashingTDIDF prezentat anterior și un obiect naiveBayesModel de tipul NaiveBayesC care urmează a fi prezentat. După ce consumer-ul este creat se obțin toate înregistrările preluate de pe topic, în limita unui parametru care poate fi setat. Pentru fiecare înregistrare este pre-procesată prin eliminarea spațiilor și semnelor de punctuație, eliminarea stop-words-urilor și transoformarea tweet-urilor într-un vector de valori double. După aceste transformări se creează un obiect de tipul JavaRDD LabeledPoint pentru a putea fi transmis clasifiatorului. Ultimul pas este apelarea metodei predic a obiectului naiveBayesModul pentru a prezice polaritatea tweet-ului. Aceste funcționalități sunt prezentate în figura 5.9.

```
consumerRecords.forEach(record -> {
    String tweetFaraSpatiiSiPunctuatie = twitterUtils.removeSpacesAndPunctuation(record.value().toString());
    String tweetFinal = twitterUtils.removeStopWords(tweetFaraSpatiiSiPunctuatie);
    Dataset<Row> labeledTweet = hashingTDIDF.tfIdf(spark, tweetFinal);
    JavaRDD<Row> rows = labeledTweet.rdd().toJavaRDD();

    labeledTweet.select("features").show();

    JavaRDD<LabeledPoint> data = rows.map(f -> new LabeledPoint(f.getDouble(0), f.getAs(4)));

    JavaPairRDD<Double, Double> predictionAndLabel = data
            .mapToPair(p -> new Tuple2<>(model.predict(p.features()), p.label()));
```

Figura 5.9 Metode pentru prelucrarea tweet-ului și prezicerea polarității

5.2.4. Modulul pentru clasificare

Înainte de începerea efectivă a clasificării trebuie ales un set de date cu ajutorul căruia clasificatorul să fie antrenat. Acest pas este foarte important, deoarece acuratețea sistemului nu depinde doar de modul în care algoritmul este dezvoltat, dar depinde și de alegerea unui set de date relevant și potrivit pentru antrenare.

Astfel, este pus la dispoziția utilizatorilor în mod gratuit setul de date Sentiment 140 pus la dispoziție de către Stanford [17]. Acesta conține aproximativ 1,6 milioane de tweet-uri gata etichetate. Acesta conține informații stocate în mai multe câmpuri :

- Polaritatea tweet-ului
- Id-ul tweet-ului
- Data când acesta a fost creat
- Interogarea utilizată pentru obținerea acestuia
- User-ul care a postat tweet-ul
- Textul efectiv al tweet-ului

Structura acestui fișier cu câteva exemple este prezentată în figura 5.10

0	1467810369	Mon Apr 06 22:19:45	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	1467810672	Mon Apr 06 22:19:49	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
0	1467810917	Mon Apr 06 22:19:53	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds
0	1467811184	Mon Apr 06 22:19:57	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
0	1467811193	Mon Apr 06 22:19:57	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
0	1467811372	Mon Apr 06 22:20:00	NO_QUERY	joy_wolf	@Kwesidei not the whole crew
0	1467811592	Mon Apr 06 22:20:03	NO_QUERY	mybirch	Need a hug
0	1467811594	Mon Apr 06 22:20:03	NO_QUERY	coZZ	@LOLTrish hey long time no see! Yes.. Rains a bit ,only a bit LOL , I'm fine thanks , how's you ?
0	1467811795	Mon Apr 06 22:20:05	NO_QUERY	2Hood4Hollywood	@Tatiana_K nope they didn't have it
0	1467812025	Mon Apr 06 22:20:09	NO_QUERY	mimismo	@twittera que me muera ?
0	1467812416	Mon Apr 06 22:20:16	NO_QUERY	erinx3leannexo	spring break in plain city... it's snowing
0	1467812579	Mon Apr 06 22:20:17	NO_QUERY	pardonlauren	I just re-pierced my ears
0	1467812723	Mon Apr 06 22:20:19	NO_QUERY	TLeC	@caregiving I couldn't bear to watch it. And I thought the UA loss was embarrassing . . . . .
0	1467812771	Mon Apr 06 22:20:19	NO_QUERY	robrobbierobert	@octolinz16 It it counts, idk why I did either. you never talk to me anymore
0	1467812784	Mon Apr 06 22:20:20	NO_QUERY	bayofwolves	@smarrison i would've been the first, but i didn't have a gun. not really though, zac snyder's just a doucheclown.
0	1467812799	Mon Apr 06 22:20:20	NO_QUERY	HairByJess	@iamjazzyfizzle I wish I got to watch it with you!! I miss you and @iamlilnicki how was the premiere?!
0	1467812964	Mon Apr 06 22:20:22	NO_QUERY	lovesongwriter	Hollis' death scene will hurt me severely to watch on film wry is directors cut not out now?

Figura 5.10 Structura setului de date

În contextul acestui proiect sunt relevante doar datele referitoare la polaritatea tweet-ului și mesajul text conținut în acesta, toate celelalte câmpuri fiind irelevante în realizarea unui clasificator de sentimente.

Modulul de clasificare folosește librăria Apache Spark Mllib pentru a implementa algoritmul Naive Bayes în vederea realizării clasificatorului. S-a ales acest algoritm deoarece funcționează eficient în multe sisteme de clasificare a datelor. Acest algoritm este unul probabilistic, adică probabilitatea de atribuire a unei etichete este determinată în funcție de etichetele preluate de la un set de date clasificat.

Setul de date pentru antrenarea sistemului este încărcat într-un obiect de tipul DataFrame astfel încât să poată fi recunoscut de către sistem. În acest timp de obiect datele sunt stocate și privite ca o colecție organizată de date. Cu o singură parcurgere a setului de date, modelul calculează probabilitatea ditribuită pentru fiecare etichetă.

[17] https://www.kaggle.com/kazanova/sentiment140

Metoda train poate avea un parametru opțional care – bernoulli sau multinomial- care să indice care tip de model Naive Bayes este folosit. În cazul implementării acestui sistem acest parametru nu s-a setat, lucru care face ca algoritmul să își aleagă metoda implicită, multinomială. Totodată ca și parametru în metoda train se setează si valoare lambda pentru netezire care are de obicei valoarea 1.

Clasificatorul tweet-urilor are 2 clase, pozitiv și negativ și nu este lipsit de provocări. Cea mai mare provocare este faptul că unele cuvinte pot avea polarități diferite în funcție de contextul din care fac parte. Astfel, polaritatea poate fi identificată greșit mai ales datorită faptului ca nu exisă încă nicio modalitate de înțelegere exactă a contextului unui tweet. Clasificarea datelor din surse cum ar fi rețelele sociale este cu atât mai dificilă cu cât mesajele publicate aici pot conține adesea cuvinte prescurtate sau cuvinte de tip jargon. Textul fiind informal, ideile sunt postate rapid, scrise prescurtat, cuvintele adeseara pierzându-și sensul.

Clasa **NaiveBayesC** conține:

- Metoda **runClassifier()** care are rolul de a prelua un set de date de intrare pe care îl transformă în obiecte de genul JavaRDD LabeledPoint și cu ajutorul cărora antrenează clasificatorul. Metoda returnează un obiect de tipul NaiveBayesModel care este folosit în clasa KafkaConsumerForTweets pentru a testa clasificatorul cu tweet-uri din streamul de date.

Clasa **Classifier** conține :

- **SparkConf** : un obiect care reprezintă configurarea pentru Spark
- **SparkSession** : un obiect care reprezintă sesiunea Spark
- **JavaSparkContext** : un obiect care reprezintă Contextul Java pentru Spark
- **Metoda main()** pentru rularea proiectului în care se apelează una din clasele pentru începerea streaming-ului de date (SearchTweetsIphone, SearchTweetsSamsung, SearchTweetsHuawei, SearchTweetsLG, SearchTweetsNewBrand) și clasa KafkaConsumerForTweets pentru consumarea datelor. Declararea și inițializarea SparkConf, SparkSession, JavaSparkContext sunt prezentate în figura 5.11

```
SparkConf sparkConf = new SparkConf()
        .setAppName("JavaNaiveBayesExample")
        .setMaster("local[2]")
        .set("spark.executor.memory", "1g");

System.setProperty("hadoop.home.dir", "C:\\winutils\\");
SparkSession spark = SparkSession.builder().
        appName("TFIDF Example")
        .master("local[2]")
        .getOrCreate();
JavaSparkContext jsc = new JavaSparkContext(spark.sparkContext());
```

Figura 5.11 Crearea sesiunii Spark

Toate aceste clase prezentate sunt cuprinse în figura 5.12 care reprezintă diagrama de clase a sistemului implementat.

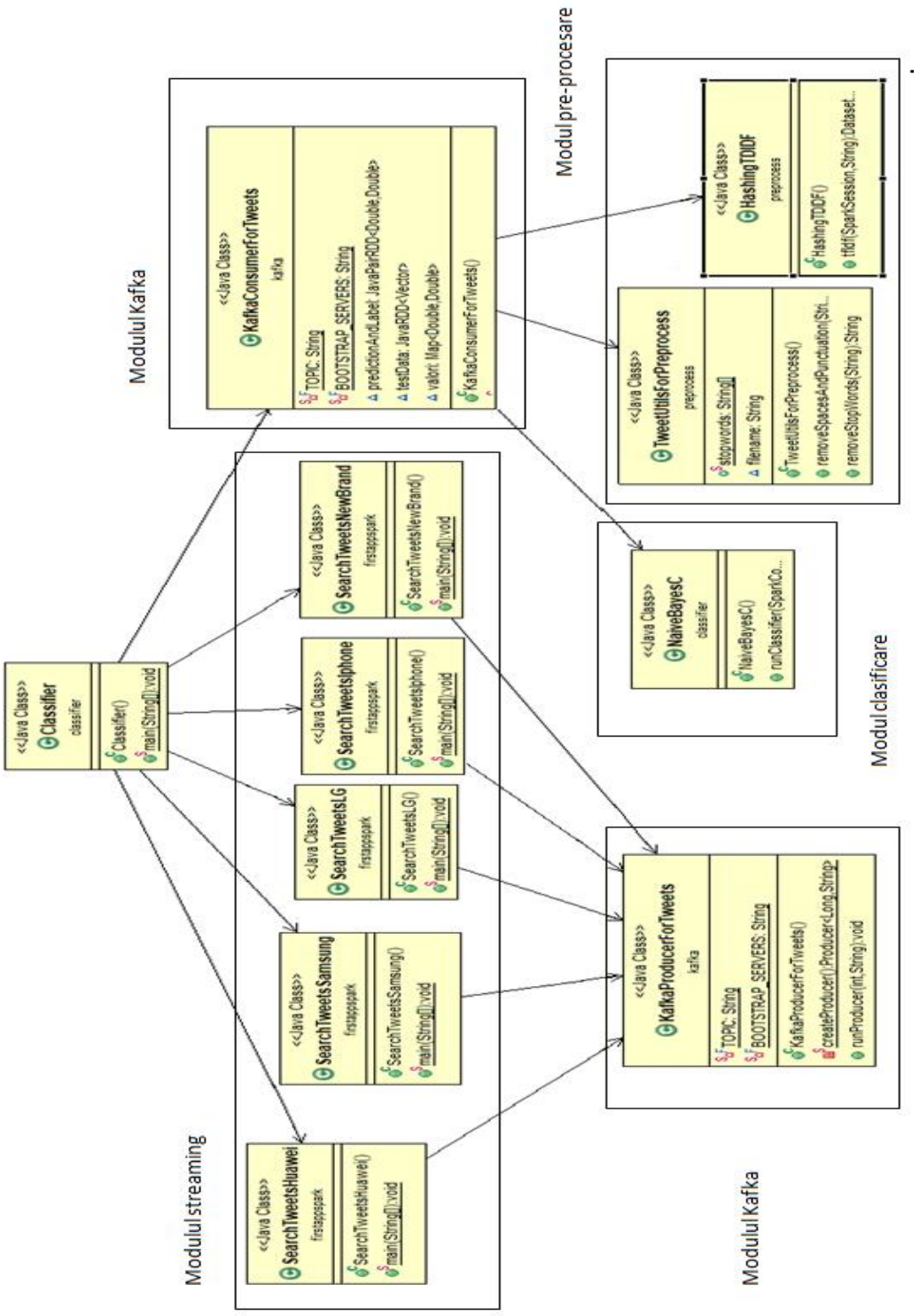

Figura 5.12 Diagrama de clase

5.3. Apache Zeppelin

Apache Zeppelin este un notebook cu care se poate interacționa prin Internet și oferă utilizatorilor posibilitatea de a vizualiza grafic și de a analiza datele. În cazul acestui sistem, acesta este utilizat pentru a analiza datele despre dispozitive mobile. Astfel, după ce un utilizator se decide asupra unu brand prin utilizarea clasificatorului prezentat, el va putea utiliza Zeppelin pentru a alege un anumite dispozitiv din acel brand.

Astfel, pe o interfață web accesată prin *localhost:8080/* utilizatorii pot vedea pentru fiecare brand în parte despre care dispozitiv s-a discutat cel mai mult pe rețeaua socială Twitter. Pentru realizarea acestui lucru se va folosi limbajul Scala, deoarce în acest moment Zeppelin nu suportă un interpretor și pentru Java.

Pentru realizarea acestui lucru, streaming-ul de date despre fiecare brand, pe lângă că este publicat pe topicul Kafka, este salvat și într-un fișier csv. În urma salvării, vor exista 4 astfel de fișiere. Următorul pas este accesarea Apache Zeppelin și crearea unui notebook nou. Un notebook este format din mai multe paragrafe și fiecare paragraf poate fi rulat separat.

Pentru obținerea graficelor în urma analizelor datelor, este necesar ca primul paragraf să conțină următoarele comenzi pentru a fi aduse local două librării necesare :

```
%dep
z.reset
z.load("org.apache.hadoop:hadoop-common:jar:2.8.2")
z.load("org.apache.zeppelin:zeppelin-file:0.7.3")
```

Următoarele comenzi au rolul de a încărca în contexul Spark fișierele care sunt necesare pentru analiză într-o variabilă. După încărcare se parcurge fișierul și se creează o listă de cuvinte prin split-urilea fișierului după fiecare spațiu. Lista obținută se convertește într-un obiect de tipul DataFrame și se salvează ca un tabel temporar.

O dată obținut un astfel de tabel, un interpretor sql poate fi utilizat pentru a rula anumite interogări asupra acestui tabel, în cazul sistemului se interoghează tabelul pentru a vedea cât de căutate sunt dispozitivele mobile. Un exemplu de interogare este :

```
%sql select value, count(1) from iphone where value like 'iphone7' or value like 'iphone8' or value like 'iphoneX'
                    or value like 'iphone6'  or value like 'iphone8plus' group by value
```

Capitolul 6. Testare şi Validare

Pentru testarea sistemului se vor urmări streaming-ul de date și rezultatele clasificării. În scopul realizării acestui lucru se va rula metoda main din clasa Classifier și în consolă vor apărea informațiile necesare. La rularea aplicației va apărea meniul prezentat în figura 6.1

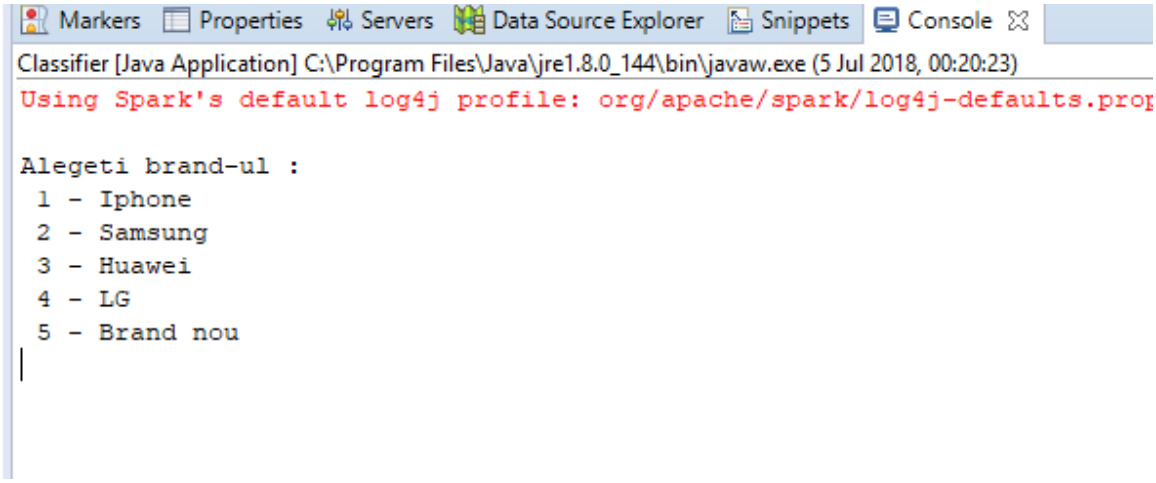

Figura 6.1 Meniul aplicației

În funcție de brand-ul care se dorește clasificat se alege una dintre opțiuni. Pentru a pune în valoare inteligența business în analiza datelor disponibile ale unei compani există și posibilitatea de a introduce noi cuvinte cheie pentru filtrarea tweet-urilor, în funcție de necesitățile companiei.

După alegerea opțiunii în consolă vor apărea informațiile : texul tweet-ului, vectorul de valori reale format din text și valoarea predicției : 0 – negativ , 1- pozitiv. Astfel, în figura 6.2 este prezentat un tweet etichetat ca fiind pozitiv, iar în figura 6.3 se prezintă un tweet etichetat ca fiind negativ.

```
TWEET : #Best iPhone X Deals for July 2018 https://t.co/5uMNCeQmoY Visit to path in additional #science #message1530736815699
VECTOR :
+--------------------------------------------------------------------+
|features                                                            |
+--------------------------------------------------------------------+
|(20,[5,7,8,9,10,12,16,17,18],[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0])|
+--------------------------------------------------------------------+

PREDICTION : 1
```

Figura 6.2 Tweet cu etichetă pozitivă

```
TWEET : RT @chribani: If #CristianoRonaldo goes to #Juventus I will donate an iPhone X to anyone who will retweet this.
VECTOR :
+--------------------------------------------------------------------------+
|features                                                                  |
+--------------------------------------------------------------------------+
|(20,[0,6,7,8,9,10,12,16,17,19],[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0])|
+--------------------------------------------------------------------------+

PREDICTION : 0
```

Figura 6.3 Tweet cu etichetă negativă

În figura 6.4 este evidențiată testarea funcționalității pentru un brand nou : Google care pune la dispoziția utilizatorilor telefoane mobile marca Google Pixel.

```
Alegeti brand-ul :  1 - Iphone 2 - Samsung 3 - Huawei 4 - LG 5 - Brand nou
5
Ati ales un brand nou
Dati noile cuvinte cheie
#googlepixel #pixelxl googlepixel googlepixelxl #googlepixelxl #googlepixel2xl
am ajuns aici

Okay but this #googlepixel camera is SO good. I'm excited to start instagraming again. ??? #parfait #preworkout… https://t.co/pUaE5ViKUJ
```

Figura 6.4 Streaming pentru un nou brand : Google Pixel

Pentru testarea Apache Zepplin există mai multe tipuri de grafice care pot fi utilizate pentru vizualizare. În continuare se prezintă rezultatele obținute pentru fiecare brand, fiecare fiind reprezentat de un alt tip de grafic.

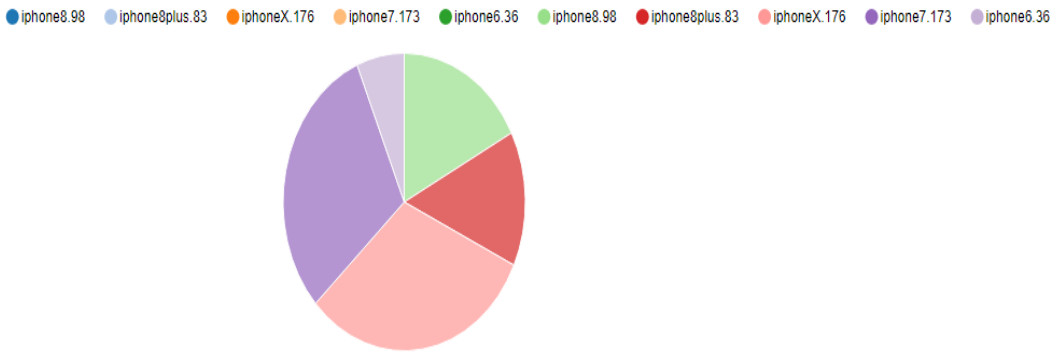

Figura 6.5 Grafic obținut pentru brand-ul Apple

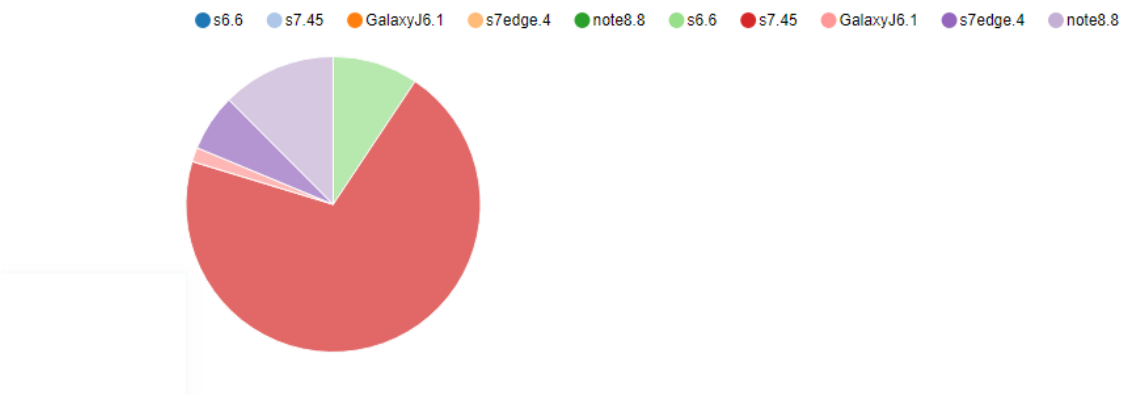

Figura 6.6 Grafic obținut pentru brand-ul Samsung

Pe baza graficelor obținute se poate observa că în cazul brandului Apple, dispozitiviul mobil iPhone X a fost cel mai apreciat de către utilizatori, iar dispozitivul iPhone 8 este următorul în ordinea preferințelor utilizatorilor. Acest lucru poate fi important pentru luarea unei decizii despre cumpărarea unui dispozitiv mobil. În cazul graficelor pentru Samsung, se poate observa că cel mai utilizat dispozitiv este S7, urmat de note8.

Pentru a analiza cele două brand-uri se poate observa că în cazul primului brand, Apple, dispozitivul cel mai căutat, iPhone X, apare în 176 de tweet-uri, în timp ce în cazul brandului Samsung dispozitivul s7 apare doar în 45 de tweet-uri. Această analiză s-a bazat pe același interval de timp. Pentru a compara celelalte două dispozitive utilizate foarte des, se poate observa ca iphone8 a fost utilizat în 98 de tweet-uri, în timp ce note8 a fost utilizat doar în 8.

Capitolul 7. Manual de Instalare si Utilizare

Pentru a rula cu succes aplicația, mașina fizică de pe care se rulează trebuie să aibă pregătite următoarele tehnologii :

- **JDK1.8.0_144**:http://www.oracle.com/technetwork/java/javase/downloads/index.html
- **Eclipse Mars** :http://www.eclipse.org/mars/
- **Apache Maven 3.5.0**: https://maven.apache.org/download.cgi
- **Apache Spark 2.2.0**: https://spark.apache.org/releases/spark-release-2-2-0.html
- **Apache Zookeeper 3.4.10**: https://zookeeper.apache.org/releases.html
- **Apache Kafka 2.12-1.1.0**: https://kafka.apache.org/downloads
- **Apache Zeppelin 0.8.0**: https://zeppelin.apache.org/download.html

După instalarea instrumentelor necesare, înainte de rularea proiectului din IDE-ul Elipse, este necesară configurarea Apache Zookeeper și Apache Kafka din linia de comandă.

Pentru rularea **Apache Zookeeper** sunt necesari următori pași :

1. Accesarea directorului conf din Zookeeper. De exemplu *C:\zookeeper-3.4.10\conf*
2. Se redenumește fișierul "zoo_sample.cfg" în "*zoo.cfg*"
3. Se deschide fișierul zoo.cfg într-un editor text și se editează *dataDir=/tmp/zookeeper* în *:\zookeeper-3.4.10\data*
4. Se adaugă variablia de sistem ZOOKEEPER_HOME = *C:\zookeeper-3.4.10*
5. Rularea zookeeper prin deschiderea unei linii de comandă și introducerea comenzii zkserver. Această comandă este prezentată în figura 7.1

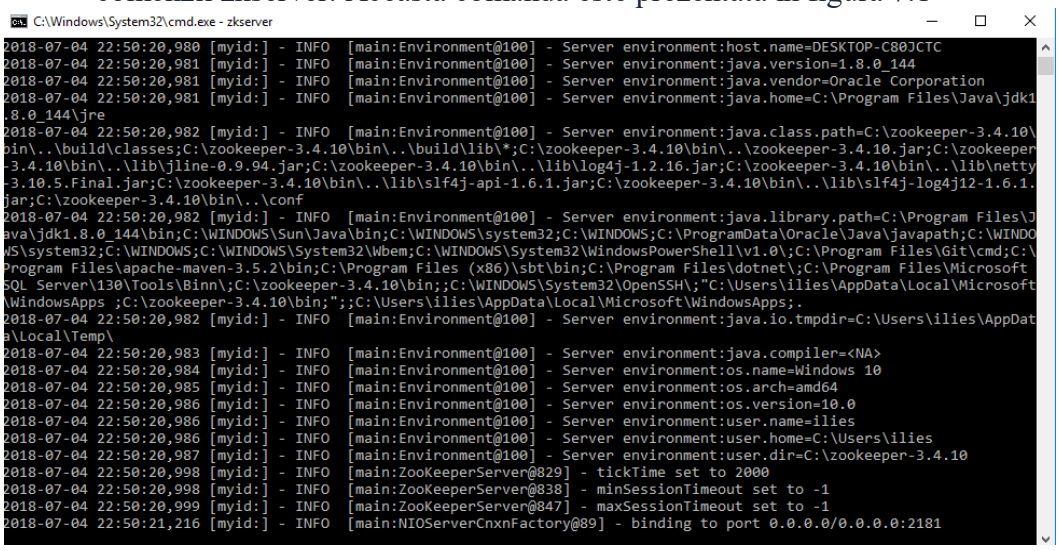

Figura 7.1 Instalarea Apache Zookeeper

În cazul în care serverul Zookeeper a pornit cu success, ar trebui să apară în consolă că a fost atribuit portului 2181.

Pentru rularea **Apache Kafka** sunt necesari următorii pași :

1. Se accesează directorul config al Kafka C:\kafka_2.12-1.1.0\config și se deschide fișiserul server.properties
2. În fișierul deschis de editează linia *log.dirs=/tmp/kafka-logsîn log.dir= C:\kafka_2.11-0.9.0.0\kafka-logs*
3. Kafka este setat în mod automat să ruleze pe portul 9092 și se contează la portul implicit Zookeeper 2181.
4. Pentru rularea Apache Kafka se accesează folderul *C:\kafka_2.12-1.1.0* și se introduce comanda *.\bin\windows\kafka-server-start.bat .\config\server.properties*
5. Dacă totul funcționează în parametrii normali, consola ar trebui să arate precum în figura 7.2

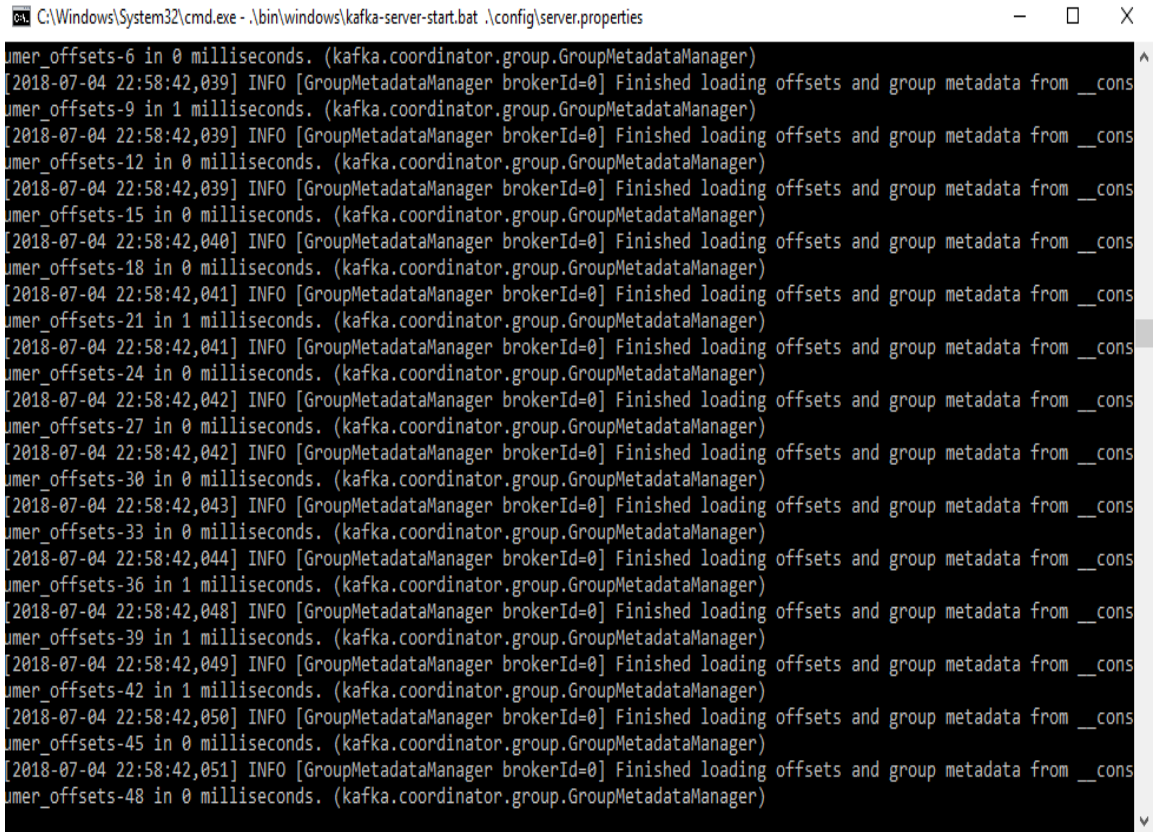

Figura 7.2 Rularea Kafka

În acest moment cluster-ul Kafka este activ și se pot crea topicuri pentru stocarea mesajelor. Așadar, următorul pas este crearea topicul tweets despre care s-a vorbit anterior. Pentru acest lucru este necesar :

1. Se deschide o altă linie de comandă în folderul *C:\kafka_2.12-1.1.0\bin\windows*și se introduce următoarea comandă :*kafka-topics.bat –create –zookeeper localhost:2181 –replication-factor 1 –partition 1 topic tweets*
2. Comanda introdusă semnifică faptul că se creează un topic numit tweets cu un factor de replicare 1 deoarece avem un singur server Kafka care rulează. În cazul unui flux mare de date se pot porni mai multe servere și atunci acest parametru va fi modificat.

Deoarece aceste tehnologii sunt pornite, se poate rula sistemul. Pentru rularea se deschide un IDE Eclipse pentru a importa proiectul. Pentru importare se selectează File -> Import -> Maven -> Existing Maven Project -> se selecteză locația și se importă fișierul. Fiind un proiect Maven, toate dependințele necesare spre librări externe sunt declarate în fișierul pom.xml. Pentru rularea proiectului este necesară aducerea tuturor librăriilor local, astfel se selectează proiectul apăsând click-dreapta, se selectează Run As aven -> Maven Install. Acest lucru este prezentat în figura 7.3 deoarece dacă librăriile nu sunt aduse corect local proiectul nu va putea funcționa

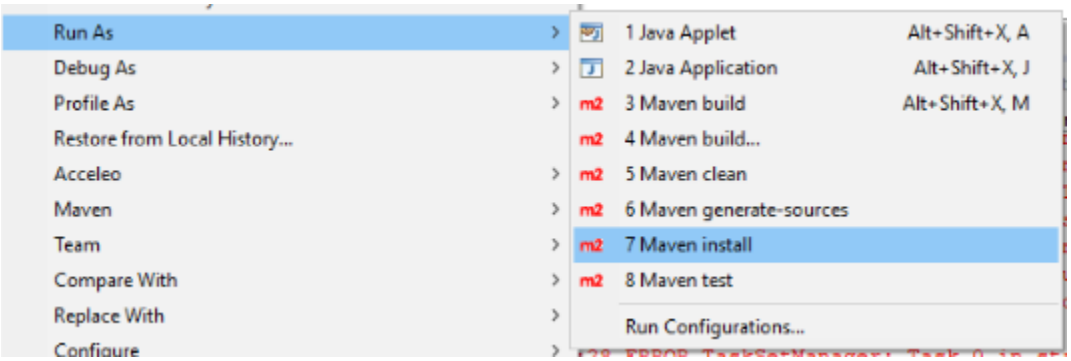

Figura 7.3 Aducerea locală a librăriilor necesare

După realizarea acestui pas proiectul poate fi rulat prin rularea ca Java Application a metodei main existente în clasa Classifier. După rularea cu succes în consolă va apărea meniul cu toate metodele posibile pe care utilizatorul le poate apela.Acesta va avea posibilitatea de a alege clasificarea pentru unul dintre brand-urile Apple, Samsung , Huawei sau LG sau poate alege un brand nou lucru pentru care va trebui să introducă o nouă listă de cuvinte cheie după care fluxul de date să fie filtrat.

Pentru instalarea Apache Zeppelin se accesează folderul *C:\zeppelin-0.8.0-bin-all\bin* și se rulează zeppelin.cmd. La lansarea în execuție a acestuia se deschide o linie de comandă iar în caz de succes apare mesajul prezentat în figura 7.4

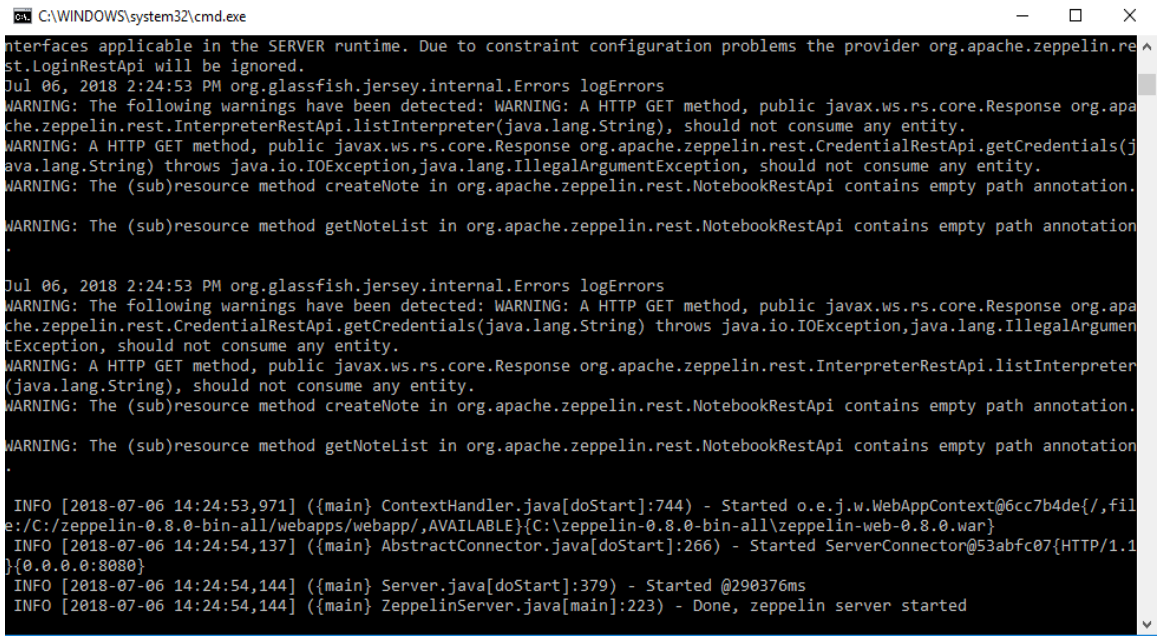

Figura 7.4 Instalarea Apache Zeppelin

Capitolul 8. Concluzii

8.1. Contribuții personale

Din categoria contribuțiilor personale face parte expunerea contextului problemei și poziționarea ei într-un domeniu actual cum ar fi Inteligența Business și procesarea în timp real a datelor.

Primul pas a fost stabilirea arhitecturii și a modulelor care vor constitui arhitectura, iar apoi au fost alese tehnologiile care să se mapeze pe fiecare modul. Astfel, Apache Spark Streaming API a fost aleasă pentru modulul de streaming, Apache Kafka a fost aleasă pentru modulul kafka , iar Apache Spark Mllib a fost aleasă pentru modulul de clasificare.

Integrarea componentelor și stabilirea comunicării dintre ele prin sistemul de mesagerie Apache Kaafka reprezintă decizii și constribuțiile personale, rezultate în urma unui amplu proces de analiză, fundamentare teoretică și învățare.

8.2. Obiective atinse

Obiectivele propuse în Capitolul 2 al acestei lucrări au fost în mare parte atinse. Astfel, streaming-ul de date necesar colectării acestora s-a realizat prin filtrarea tuturor tweet-urilor, în sistem întrând doar cele care conțin cuvinte cheie actuale despre dispozitive mobile și sunt în limba engleză. Colectarea și analiza lor este una în timp real, cu lateță scăzută.

În urma etapei de documentare și identificare a unei metode potrivite de clasificare a fost aleasă implementarea unui clasficiator NaiveBayes care împarte tweet-urile în două categori: positive și negative.După antrenarea sistemului cu acest clasficator a fost atins și obiectivul care își propunea ca sentimentele din tweet să fie extrase și să primească o polaritate.

Pentru promptitudine s-a atins și obiectivul prin care tehnologia Apache Kafka a fost integrată, pentru a asigura integritatea mesajelor, acestea fiind scutite de riscul de a fi pierdute în cazuri de eșec. Bineînteles, prin atingerea acestui obiectiv s-a atins și obiectivul care propunea integrararea tehnologiei Apache Zookeeper deoarece cele două tehnologii funcționează împreună.

8.3. Posibile dezvoltări ulterioare

Deși o mare parte din obiectivele propuse au fost îndeplinite, există un obiectiv care nu s-a dus spre ideplinire și anume rularea într-un serviciu Cloud a sistemului, cum ar fi EC2 pus la dispoziție de Amazon Web Servicies (AWS). Astfel, acest lucru poate fi considerat o dezvoltare ulterioară ce poate fi adusă sistemului.

O altă dezvoltare poate fi considerată crearea unei aplicații web, astfel încât rezultatele sistemului și introducerea de noi cuvinte cheie pentru un brand nou să nu se

realizeze în mediul de dezvoltare , ci să fie oferite utilizatorilor printr-o interfață mai placută și mai ușor de utilizat. Aceste două dezvoltări pot asigura o disponibilitate mare sistemului, deoarece aplicația web daca va fi rulată într-un serviciu cloud va fi disponibilă utilizatorilor non-stop.

Framework-urile utilizate oferă posibilități avansate de rulare într-un cluster, dar în contextul implementării acestui sistem Apache Spark a rulat pe un singur nod, iar Apache Kafka a avut un singur factor de replicare și un singur tweet. Pentru o performanță mai mare se poate crea un cluster Kafka cu mai multe servere și câte un topic pentru fiecare brand.

O altă îmbunătățire poate fi adusă clasificatorului în ceea ce privește clasele de polaritate, astfel se poate introduce și clasa netru pentru tweet-urile care fac parte din această categorie, momentan acestea fiind incluse în cele două categorii existente. Fiind vorba de un domeniu machine-learning, care cunoaște o expansiune amplă în zilele noastre, dezvoltările ulterioare pot include și îmbunătățirea algoritmilor existenți sau dezvoltarea de algoritmi noi, mult mai perfomați care să asigure o acuratețe mai mare.

Bibliografie

[1] Cunningham P., Cord M., Delany S.J , "Machine Learning Techniques for Multimedia", Springer, Berlin, Heidelberg, pag. 21-49, 2008.

[2] K. Nigam, J.Lafferty, A. McCallum, K. Nigam, J. Lafferty, and A. McCallum. „Using Maximum Entropy for Text Classification. In IJCAI-99 workshop on machine learning for information filtering", volum 1, pag. 61–67, 1999. [Online].Available:https://www.cc.gatech.edu/~isbell/reading/papers/maxenttext.pdf

[3] N. Cristianini, J. Shawe-Taylor, „An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods", Cambridge University Press, New York, 2000.

[4] A. Palsson, D. Szerszen, „Sentiment Classification in Social Media", Stockholm, 2016.[Online].Available:https://www.divaportal.org/smash/get/diva2:930520/FULLTEXT01.pdf

[5] S. Russell, P.Norvig, „Artificial Intelligence : A Modern Approach" , 3rd, Prentince Hall Press Upper Saddle River, NJ, USA, 2009

[6] C. Kaushik, A. Mishra, „A Scalable, Lexicon Based Technique for Sentiment Analysis" , YMCA University of Science & Technology, Faridabad, 2014.

[7] C. Sutton , A. McCallum, „An Introduction to Condition Random Fields in Foundation and Trends in Machine Learning", vol 4, 2012. [Online].Available: http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf

[8] S. Yin, Big Data for Modern Industry : „Challenges and Trends", IEEE, 2015.[Online].Available:https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7067026

[9] J. Dean ,S. Ghemawat, MapReduce: „Simplified Data Processing on Large Clusters în Communications of the ACM", vol. 51 , pag 107-113, 2008. [Online].Available:https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

[10] M. Stonebraker, U. Cetintemel, S. Zdonik, „The 8 Requirements of Real-Time StreamProcessing",2005.[Online].Available:https://sigmodrecord.org/publications/sigmodRecord/0512/p42-article-stonebraker.pdf

[11] W. Wingerath, F. Gessert, S. Friedrich, N. Ritter, „Real-time stream processing for Big Data în Information Technology", volum 58, pag 186-195, 2016.[Online].Available:https://vsiswww.informatik.unihamburg.de/getDoc.php/publications/561/Real-time%20stream%20processing%20for%20Big%20Data.pdf

[12] H.Karau,A.Konwinski,P.Wendell, M.Zaharia. „LearningSpark: Lightning-Fast Big Data Analytics", O'Reilly Media, Inc., Sebastapol, 2015

[13] A. Go, R. Bhayani, L. Huang , „Twitter Sentiment Classification using Distant Supervision,Standford",2009.[Online].Available:http://www.yuefly.com/Public/Files/2017-03-07/58beb0822faef.pdf

[14] K. Guido, „Cloud Computing & Confidentiality", University of Twente, 2010. [Online].Available:http://essay.utwente.nl/61039/1/MSc_G_Kok_DIES_CTIT.pdf

[15] Y. Ou, „The Concept Of Cloud Computing And The Main Security Issues In It", Turku University of AppliedSciences,2015.[Online].Available: https://www.theseus.fi/bitstream/handle/10024/96535/Ou_Yang.pdf?sequence=1 \

[16] Zhaoyu Li, "Naïve Bayes Algorithm for twitter sentiment analysis and its implementation in MapReduce", The Faculty of the Graduate School At the University of Missouri,2014.[Online].Available: https://mospace.umsystem.edu/xmlui/bitstream/handle/10355/45675/research.pdf?sequence=1

Anexa 1 Lista figurilor din lucrare

Anexa 2 – Glosar

Termern	Descriere
IDC	Internation Data Corportaion
Twitter	Rețea de socializare
Tweet	Mesaj publicat pe Twitter
SVM	Suport Vector Machines
CRF	Conditional Random Field
FCA	Format Concept Analysis
GFS	Google File System
HDFS	Hadoop Distributed File System
SQL	Structured Query Language
API	Application Programming Interface
RDD	Resilient Distributed DataSet
HQL	Hive Query Language
TF	Term Frequency
IDF	Inverse Document Frequency
DStream	Discretized Stream
JDK	Java Development Kit
IDE	Integrated Development Environment