MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**
**DEPARTAMENTUL CALCULATOARE**

# TweetSent – Sistem de analiză a sentimentelor din surse de date sociale bazat pe tehnologia Spark

LUCRARE DE LICENŢĂ

Absolvent: **Andreea GHIC**

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

**2017**

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

**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: **Andreea GHIC**

**TweetSent – Sistem de analiză a sentimentelor din surse de date sociale bazat pe tehnologia Spark**

1. **Enunţul temei:** *Proiectul își propune realizarea unui sistem de analiză big-data, pentru determinarea și clasificarea sentimentelor în timp real a unor mesaje tweet din rețeaua socială Twitter. Această analiză se va realiza pe baza unor date de intrare reale, iar rezultatele obținute reflectă sentimente din lumea actuală. De asemenea, proiectul va oferi și o clasificare la nivel de geolocație a tweet-urilor respective.*

2. **Conţinutul lucrării:** *Cuprins, Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalate și Utilizare, Concluzii, Bibliogrfie, Anexe*
3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare

4. **Consultanţi**: as.ing. Cosmina IVAN

5. **Data emiterii temei:** 1 noiembrie 2016

6. **Data predării:** 17 Iulie 2017

Absolvent: _____

Coordonator ştiinţific: _____

# Cuprins

# Capitolul 1. Introducere

În prezent, datele cresc și se acumulează mai repede decât oricând, aproximativ 90% din toate datele generate în lumea noastră au fost generate doar în ultimii doi ani. Crește exponențial atât volumul de date care necesită stocare și prelucrare, cât și sursele care furnizează aceste date.

Datele provin din nenumărate surse: senzorii stochează informații referitoare la mediul lor, sisteme de calcul complexe generează fișiere de logare, tranzacțiile online reprezintă 98% din plățiile făcute la nivel global, iar oamenii distribuie informații prin intermediul rețelelor sociale din toate colțurile lumii.

Aceasta fenomen se datorează evolutiei tehnologice, companiilor precum Amazon, Google, dar în special rețelele sociale precum Facebook și Twitter au contribuit la această creștere. Datele devin tot mai complexe, ele circulă la o viteză tot mai mare, iar volumul lor crește tot mai mult. Fenomenul este cunoscut sub numele de “Big Data”, care astăzi reprezintă un subiect foarte popular în domeniul Științei Calculatoarelor.

## 1.1. Contextul proiectului

Datorită mărimii si a complexității acestor seturi de date, ele au devenit foarte greu, chiar imposibil de gestionat cu instrumentele clasice de procesare a datelor. Printre tool-urile actuale din domeniul “Big Data Analytics” se numară și Hadoop, fiind unul din cele mai folosite la momentul actual. În tot mai multe cazuri este nevoie de procesarea datelor în timp real, care presupune analiza acestora în momentul în care sunt interceptate, pentru a obține rezultate de actualitate. Obținerea rezultatelor aproape instantaneu este necesară pentru a putea reacționa și a lua decizii imediat după ce un eveniment a avut loc.

Viteza cu care datele sunt create și trebuie procesate poate deveni extrem de mare. Ca urmare, a fost necesară dezvoltarea unor tehnologii noi de procesare și stocare a datelor pentru a face posibilă analiza unei asemenea informații complexe. Noi arhitecturi au fost dezvoltate și framework-uri noi au fost implementate pentru a depăși limitările atinse de către sistemele tradiționale. Arhitecturile de procesare big-data și cele mai populare tool-uri pentru procesarea datelor în timp real vor fi descrise în cadrul lucrării.

Cele mai multe companii folosesc rețelele sociale pentru a-și promova afacerea și pentru a fi mai aproape de actualii și posibilii viitori clienți. Folosind rețele sociale precum Facebook, Twitter sau Google+, întreprindere postează și distribuie infomații, însă metricile puse la dispoziție de acestea eșuează în a reflecta detalii de o relevanță semnificativă în campaniile de marketing, obținerea a noi clienți sau a generării de venituri.

Din fericite, inseparabilitatea rețelelor sociale și big-data facilitează aplicarea unor noi strategii de marketing. Relevanța informației și domeniul de aplicabilitate al datelor permit crearea mai multor abordări predictive de analiză.

Societatea umană este mereu influențată de opinia celor din jur. Fie că e vorba de achiziționarea unui noi produs, citirea unei noi cărți, vizionarea unui film, vizitarea unui loc nou, sau chiar votarea în cadrul unor alegeri importante, toate deciziile care urmează a fi luate sunt dependente de opiniile pozitive sau negative ale altora.

## 1.2. Conținutul lucrării

Lucrarea este alcătuită din 8 capitole, iar în cele ce urmează va fi prezentată structura acesteia, dar și conținutul fiecărui capitol:

**Capitolul 1 – Introducere –** realizează o plasare inițială în tema lucrării. În acest capitol se descrie contextul proiectului prin menționarea evoluției complexității datelor care necesită stocare și procesare, conținutul și motivația lucrării.

**Capitolul 2 – Obiectivele Proiectului –** enumără pornind de la contextul proiectului și motivația prezentată în capitolul de introducere temele tratate de această lucrare. Obiectivul lucrării este de a prezenta o serie de noțiuni teoretice din domeniul big-data, a metodelor de procesare și a diferitelor framework-uri disponibile pe piață, urmată de implementarea unei aplicații concrete de analiză și clasificare de sentimente ale datelor în timp real folosind framework-urile Apache Spark.

**Capitolul 3 – Studiu Bibliografic –** introduce conceptul de "Big Data" și îl devinește din mai multe puncte de vedere. De asemenea, acest capitol prezintă metodele de procesare batch și streaming folosite în analiza big-data și a arhitecturii lambda, care reprezintă o abordare hibridă propusă pentru depășirea limitărilor celor doua metode de analiză. Mai mult, realizează o prezentare a metodelor de clasificare a sentimentelor.

**Capitolul 4 – Analiză și Fundamentare Teoretică –** este centrat în jurul framework-ului principal de procesare big-data în timp real utilizat în lucrare. În acest capitol se realizează o descriere a celor mai folosite tool-uri de analiză și prezintă modelul de procesare a acestora, dar și studiul detaliat al framework-ului Apache Spark. Studiul acestor unelte/framework-uri a fost necesar în alegerea tehnologiei folosite la implementarea aplicației descrise în cadrul lucrării și pune în evidentă atât avantajele cât și limitările framework-ului Apache Spark. De asemenea, prezintă modelul conceptual de analiză de sentimente al aplicației.

**Capitolul 5 - Proiectare de Detaliu și Implementare –** descrie în detaliu arhitectura și modelul de procesare al unei aplicații realizate în Apache Spark de clasificare a sentimentelor mesajelor din rețeaua socială Twitter, intitulată TweetSent. Clasificare acestor mesaje, numite tweets, presupune analiza acestora cu diferiti algoritmi de analiză a sentimentelor. **Capitolul 6 – Testare și Validare –** descrie criteriile pe baza cărora a fost testată aplicația implementată.

**Capitolul 9 – Manual de Instalare și Utilizare –** prezintă etapele necesare instalării tututor tool-urilor necesare rulării aplicației și prezentarea modului de utlizare a sistemului.

**Capitolul 8 – Concluzii –** prezintă un rezumat al contribuțiilor aduse și propune și o serie de îmbunătățiri și extensii ale aplicației, enumerate în secțiunea de dezvoltări ulterioare.

# Capitolul 2. Obiectivele Proiectului

Acest capitol va prezenta obiectivele propuse. Lucrarea va realiza o descriere detaliată a conceptului de "Big Data", prin definirea acestuia din diferite perspective. De asemenea, lucrarea va prezenta diferitele modele de procesare și arhitecturi, urmate de o analiză succintă a framework-urilor folosite la ora actuală și de una detaliată a framework-ului utlizat. Pe baza acestor concepte și a relevanței datelor aflate în rețele sociale, atât pentru întreprinderi care se promovează prin intermediul acestora, cât și pentru pentru companii și personane individuale, lucrarea va descrie implementarea și funcționalitatea unui sistem de procesare și analizaă de sentimente, în timp real, intitulat TweetSent.

## 2.1. Obiectivul principal

Scopul lucrării este de a identifica și utiliza într-o aplicație reală un sistem de procesare Big Data pentru fluxuri de date în timp real. De asemenea, aceasta propune o soluție pentru clasificarea pe bază de sentimente a informațiilor din rețeaua socială Twitter și vizualizarea acestora la nivel de geolocație folosind framework-ul Apache Spark.

## 2.2. Obiective secundare

Pentru a obține rezultate relevante în urma analizei realizate, procesarea trebuie să fie una de timp real. Latența permisă într-un astfel de model de procesare este cel mult de ordinul secundelor.

De asemenea, sistemul implementat va folosi ca și date de intrare date reale din cadrul rețelei sociale Twitter. Accesul la acestea poate fi realizat pe bază de stream-uri, care facilitează execuția în timp real. Motivul pentru care datele de intrare sunt alese astfel este pentru a putea obține rezultate în timp real. Astfel, un obiectiv propus devine și obținerea unei clasificări de sentimente a celor mai recente subiecte care să reflecte realitatea, obținute pe baza unor criterii integrate în soluții de tip machine learning.

Opiniile exprimate online pot oferi perspective valoroase asupra aspectelor economice, sociale și politice ale națiunilor. Așa că, astăzi, nu ne bazăm doar pe opiniile oamenilor pe care îi cunoaștem, ci și pe străinii complet, în ceea ce privește atât interesele comerciale, cât și cele publice. Companiile conțin, de asemenea, date interne cu privire la e-mailuri, comunicări prin call center, feedback de la clienți, etc. O formă de evaluare a sentimentul din această multitudine de date păreri ce referă opinii legate de aspecte funcționale, tehnice, tehnologice, de trafic, etc. încep sa fie utilizate în tot mai multe domenii comerciale și de cercetare.

Un alt obiectiv este identificarea și utilizarea unei metode de clasificare prin care să se obțină rezultate cât mai precise, prin selectarea informației relevante din imensul volum de date produs de rețeaua socială Twitter.

Implementarea trebuie să facă față creșterii exponențiale a volumului de date care necesită procesare. Așadar, scalabilitatea sistemului devine un obiectiv major în ceea ce privește implementarea sistemului descris în lucrare.

Pe baza acestor caracteristici propuse pentru implementarea aplicației, modelul de procesare va presupune următorii pași, prezentați și în Figura 2.1:

- Citirea unor date reale
- Procesarea acestora in timp real
- Furnizarea datelor de ieșire în timp real

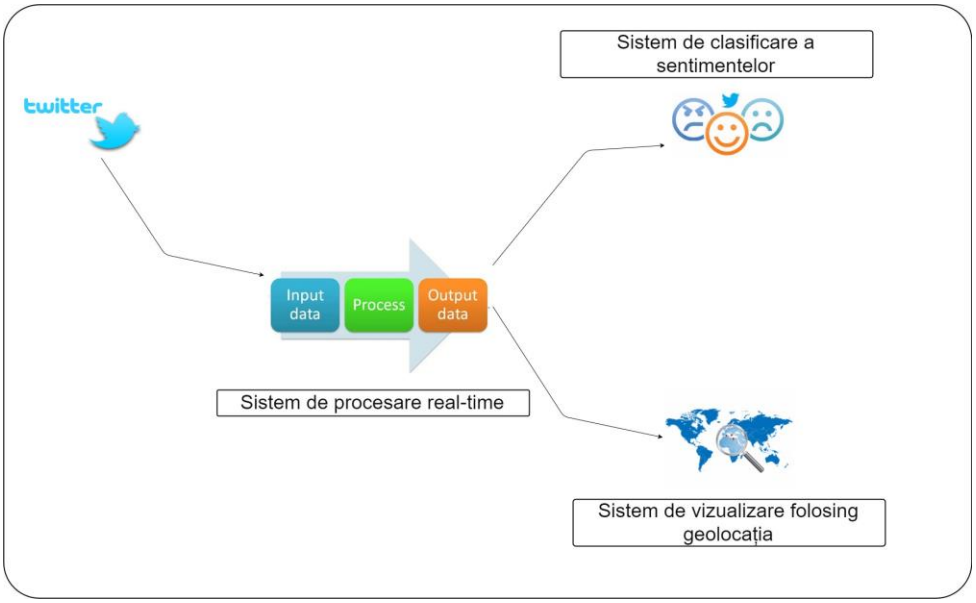



Figura 2.1 Obiectivul sistemului de analiză a tweet-urilor în timp real

Apache Spark reprezintă un sistem distribuit, open-source utilizat în analiză big-data în timp real, care e tolerant la eșec și garantează procesarea datelor, aceste propietăți ale framework-ului devenind obiective în cadrul implementării aplicației. De asemenea, reprezintă sisteme scalabile, fiind concepute pentru rularea în cluster. Un cluster reprezintă o soluție independentă de servere care colaborează ca un sistem unitar în scopul de a oferi servicii de mare disponibilitate. Scalarea orizontală este obținută prin alocarea mai multor noduri de execuție în cluster și implicit a mai multor resurse hardware.

# Capitolul 3. Studiu Bibliografic

Documentarea bibliografică are ca obiectiv prezentarea stadiului actual al domeniului/sub-domeniului în care se situează tema. Conceptul de "Big Data" este unul relativ nou, devenit popular în ultimul deceniu, iar definirea acestuia este relativ complexă datorită proprietăților care caracterizează volumele imense de date aflate într-o continuă creștere. De asemenea, metodele tradiționale de procesare nu au putut fi scalate la cerințele impuse de către analiza big-data. Capitolul 3 va defini acest concept și va prezenta diferitele arhitecturi și modele de procesare folosite pentru a depăși limitele atinse de sisteme tradiționale.

## 3.1. Big Data

Cantitatea de date generate în fiecare zi în lume explodează. Creșterea volumului de mass-media digitale și sociale ajută și mai mult. Abilitatea de a analiza această cantitate enormă de date aduce o nouă eră a creșterii productivității, a inovării și a excedentului de consum.

### *3.1.1. Definirea conceptului de "Big Data"*

Volumul de date existente în lume crește exponențial de la o zi la alta și la fel cum crește și interesul pentru domeniul "Big Data". Consultând Google Trends – un tool care analizează cât de des apare un cuvânt raportat la numarul de căutări, bazat pe motorul de căutare Google, putem observa cu ușurință o creștere a frecvenței de căutare de-a lungul anilor, reprezentată în Figura 3.1

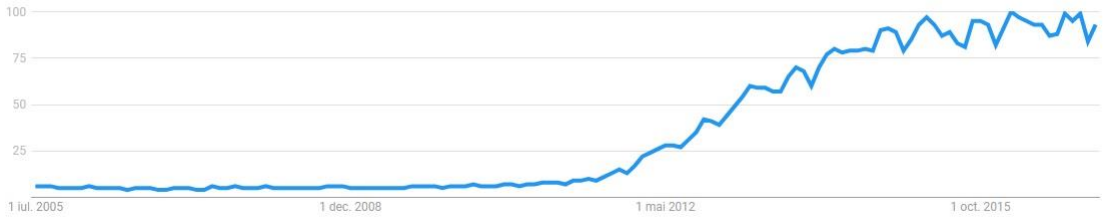



Figura 3.1 Interesul în căutare a termenului de "Big Data" pe Google[1]

Aproape 90% din datele existente în prezent la nivel mondial, au fost generate în ultimii doi ani, majoritatea fiind date nestructurate. *Big Data* este un concept care se referă la o inițiativă informatică care rezolvă problema procesării unei cantități imense de date, într-un interval de timp limitat [4].

Cu toate că, conceptul de *Big Data* este pe zi ce trece tot mai prezent, încă nu există o definiție unanim acceptată pentru acest termen. Conform MIT Technology

[1] https://trends.google.ro/trends/explore?date=2005-06-03%202017-01-5&q=Big%20Data

Review (2013)[2] noțiunea de “big” este greu de de identificat, deoarece un set de date vazut azi mare, în viitorul apropiat va părea mic.

La prima vedere, putem defini „Big Data” ca seturi mari de date foarte complexe, care sunt imposibil sau greu de gestionat cu instrumente clasice de procesare a datelor.

Termenul de „big” este raportat la tehologia utilizată în prezent pentru procesarea datelor, definit ca fiind mai mult decât poate fi procesat prin tehnici convenționale.

În lipsa unei definiții unice, marile companii care au contribuit la fenomentul de Big Data, folosesc o serie de definiții proprii:

**Microsoft**: Big Data este termenul folosit din ce în ce mai des pentru a descrie întregul proces de aplicare a puterii mari de calcul, întâlnită în domenii precum machine learning și inteligență artificială – pe seturi de date masive și complexe.

**Oracle**: Big Data este derivarea valorii din baza de date tradițională, augumentată cu noi surse de date nestructurate.

**Intel**: Oportunități Big Data apar în organizații care generează un volum de date mediu de 500 terrabytes pe săptămână.

Rezumând, Big Data reprezintă o nouă generație de tehnologii și arhitecturi destinate extragerii de valoare din cadrul volumelor foarte mari de date care au o mare varietate, permițând prelucrarea și analiza acestora în timp real.

În 2012, Meta (în prezent Gartner) a sugerat unul dintre cele mai populare moduri de a caracteriza “Big Data”, pe baza celor trei V-uri: Volum, Varietate, Viteză [5].

### *3.1.2. Cele 3 V-uri din Big Data*

O nouă definiție conturează dimensiunile Big Data, actualizată astfel: "Big data înseamnă informaţii de mare volum, mare viteză şi/sau mare varietate ce necesită noi forme de procesare pentru a facilita luarea deciziilor, descoperirea semnificaţiilor şi optimizarea proceselor."[3]

**Volmul** este cel mai provocator aspect al Big Data, deoarece impune o nevoie de stocare scalabilă și o abordare distribuită a interogării. Întreprinderile mari au deja o cantitate mare de date acumulate și arhivate de-a lungul anilor. Cantitatea acestor date ajunge cu ușurință la punctul în care sistemele convenționale de gestionare a bazelor de date ar putea să nu fie capabile să o gestioneze. Soluțiile bazate pe depozitul de date, nu au neapărat capacitatea de a procesa și analiza aceste date, datorită lipsei arhitecturii de procesare paralelă. Tehnologiile Big Data oferă o soluție pentru a crea valoare din aceste date masive și dificil de procesat. [2]

**Viteza** se referă atât la rapiditatea cu care datele sunt produse, cât și la rapiditatea cu care datele trebuie să fie prelucrate pentru a satisface cererea. Acest lucru implică

[2] https://www.technologyreview.com/s/519851/the-big-data-conundrum-how-to-define-it/

[3] https://www.todaysoftmag.ro/article/879/big-data-si-social-media-marea-schimbare

fluxuri de date, crearea de înregistrări structurate, precum și disponiblitatea pentru access [4]. Diferite tipuri de Big Data ar putea necesita o procesare la viteze diferite, conform următoarei clasificări [3]:

**Batch**: La un interval de timp, datele sunt stocate și procesate. Această metodă va fi prezentată în detaliu în capitolul 3.3.

**Near-time**: Se apropie de procesarea în timp real, intervalul de timp între două procesări fiind foarte mic. Datele sunt colectate instantaneu, dar procesarea are loc periodic, în anumit intervale de timp: de la o dată pe zi, la o dată la cateva ore sau chiar câteva minute.

**Real-time/Streaming**: datele sunt procesate concomitant cu primirea lor. Mai multe detalii vor fi prezentate în capitolul 3.3.

**Varietatea:** diferite surse de big data generează diferite forme de date. Pe măsură ce apar aplicații noi, apar și formate de date noi [3]. Varientatea include date tabelare (baze de date), date ierarhice, documente, XML, e-mailuri, blog-uri, mesaje instant, click stream-uri, fișiere log, date de contorizare, imagini statice, audio, video, date despre cursul acțiunilor, tranzacții financiare, etc [4].

Recent un al 4-lea V a fost ataşat ulterior definiţiei: veridicitatea. Veridicitatea se referă la cât sunt de încredere datele sau de îndoielnice. Calitatea datelor Big Data este mai puțin controlabilă deoarece provine din  diferite sursepentru care nu se poate garanta calitatea conținutului și forma lui de prezentare [4]. Veridicitatea aplică atât la datele de intrare, cât și în contextul datelor de ieșire obținute în urma procesării acestora [3].

## 3.2. Metode de procesare Big Data

### *3.2.1. Introducere*

Există diverși senzori prin care datele sunt culese prin diferite abordări, cum ar fi dispozitivele inteligente, camerele de luat vederi sau sateliții. Aceși senzori generează o cantitate imensă de date care trebuie stocate, procesare și analizate. Există organizații care colectează date din diferite motive, cum ar fi cercetarea, campaniile de piață, detectarea diferitelor tendințe pe piață și descrierea unui fenomen social în întreaga lume. Aceste organizații consideră aceste date drept noul *petrol* [4]. Fiind o cantitate prea mare de date pentru a fi extrase și procesate prin intermediul unui calculator obișnuit, e necesară o putere de procesare considerabilă și o capacitate de stocare foarte mare [1].

Acest lucru a dus la apariția mai multor tehnologii și tehnici de procesare care să ofere aplicabilitate într-o varietate de domenii, de la matematică, informatică, biologie, fizică, la probleme de rețelistică, statistică și aplicații sociale.

Metoda de procesare folosită pentru a analiza aceste informații, este principala caracteristică a intrumentelor utilizate pentru a studia aceste date. În continuare, se vor descrie cele mai populare doua metode, procesarea în manieră batch, detaliată în capitolul 3.3.2 și cea în timp real, numită și procesare de tip streaming, descrisă în capitolul 3.2.3.

[4] http :// ana.blogs.com /maestros/2006/11/data_is_the_new.html

În mod evident, există și sisteme care întegrează ambele modele de procesare, arhitectură despre care vom discuta în capitolul 3.2.4.

### *3.2.2. Procesarea batch folosind modelul Map Reduce*

Framework-urile de procesare batch prelucrează datele de tip Big Data într-un cluster de calculatoare și consideră datele colectate într-o perioadă de timp, pentru a fi procesate.

Pentru a face posibilă procesarea paralelă pe seturi mai de date, Google a introdus MapReduce, o paradigmă de dezvoltare folosită pentru a interoga volume imense de date în clustere de dimensiuni mari [3] și implementată în infrastructura Apache Hadoop [6], pe care se bazează majoritatea tehnologiilor de procesare batch.

Acest model este responsabil de distribuirea job-urilor Big Data într-un număr mare de clustere, se ocupă de eșecurile nodurilor, comunicațiile acestora, utilizarea memoriei și a rețelei.

Map Reduce se bazează pe împărțirea procesării în două etape: Map și Reduce, fiecare primind ca parametri de intrare o pereche cheie-valoare, al cărei tip este specificat de programator, rezultând tot o pereche cheie-valoare[5]. Acest model constă dintr-un sistem de fișiere, un nod principal și mai multe noduri secundare, noduri de date. Sistemul de fișiere împarte datele în bucăți, după care stochează fragmentele de date într-un mod distribuit pe nodurile de cluster. Framework-ul rulează pe un cluster cu setul de a mima seturi de date de dimensiuni foarte mici.

După cum se poate observa în Figura 3.2, întâi se aplică funcția map tuturor membrilor unui set de date, care returnează o lista cu rezultate, apoi funcția Reduce colectează si rezolvă rezulatele din una sau mai multe opreații de mapare executate în paralel. În urma executării repatate a acestor două funcții, setul de date de intrare este redus semnificativ, astfel încât la ieșire se va produce un subset al acestuia , reprezentând informația relevantă extrasă în urma analizei.

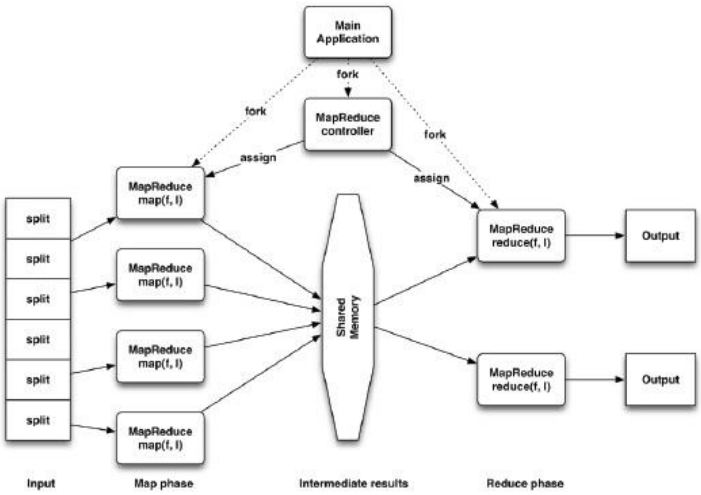



Figura 3.2 Modul în care MapReduce funcționează [7]

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

Prin urmare, procesarea batch, la baza căruia stă modelul MapReduce presupune trei etape: stocarea datelor într-un batch, procesarea lui și stocarea rezultatelor. Colectarea datelor presupune presupune citirea unui set de date de dimensiunea batch-ului. În momentul în care batch-ul este plin, datele sunt trimise către procesare. Astfel de obține un model de execuție "framed", fie din punct de vedere al timpului (la anumite interval impuse), fie din punct de vedere al volumului de date (impus de dimensiunea batch-ului).

Numărul de computere care rulează în cluster și dimensiunea setului de date sunt parametrii cheie care determină timpul de execuție al unei lucrări de procesare discontinue. Poate dura câteva minute, ore sau chiar zile. Prin urmare, cadrele de prelucrare în lot au o latență ridicată pentru a procesa datele mari. Ca urmare, acestea nu sunt adecvate pentru a satisface constrângerile în timp real [1].

### *3.2.3. Procesare Streaming*

Principalul dezavantaj al procesării batch – faptul că datele care sunt obținute la ieșire ca rezultat al analizei de tip big data apar cu o întârziere relativ mare, raportat la momentul când acestea au fost captate – duce la dezvoltarea unor modele de procesare noi.

Unul dintre acestea este procesarea de tip streaming, un model folosit pentru returnarea rezultatelor cu latență redusă. În plus, aceasta include majoritatea funcțiilor modelului batch, cum ar fi toleranța la erori sau utilizarea resurselor. Spre deosebire de procesarea batch, procesarea in timp real, numită și procesare streaming, are ca țintă procesarea datelor colectate într-o perioadă mică de timp. Prin urmare, procesarea streaming trebuie sincronizată cu fluxul datelor. Dacă un sistem în timp real necesită X + 1 minute pentru a procesa seturi de date care sunt colectate în X minute, atunci acest sistem poate să nu țină pasul cu volumele de date și rezultatele pot deveni depășite [8].

În plus, stocarea datelor în astfel de cadre se bazează de obicei pe ferestre de timp, care sunt nelimitate. Dacă viteza de procesare este mai mică decât viteza redusă a datelor, ferestrele de timp vor scădea unele date când sunt pline. Deci, sistemul devine inexact din cauza lipsei datelor. Prin urmare, toate ferestrele de timp ar trebui să aibă potențialul de a adapta variațiile vitezei datelor de intrare și de ieșire prin faptul că pot să alunece după cum și când este necesar [9].

În cazul congestionării rețelelor, sistemele în timp real ar trebui să poată face față datelor întârziate, care lipsesc sau care au suferit modificări (defecte). Acestea ar trebui să garanteze rezultate previzibile și repetabile. Similar cu sistemele de prelucrare batch, sistemele în timp real ar trebui să producă același rezultat având același set de date dacă operațiunea este reefectuată ulterior. Prin urmare, rezultatul ar trebui să fie determinist [9].

Sistemele de procesare a fluxului de date streaming ar trebui să fie disponibile tot timpul. Acestea ar trebui să gestioneze defectarea nodului, deoarece disponibilitatea ridicată este o preocupare importantă pentru procesarea streaming. Prin urmare, sistemul ar trebui să reproducă starea informațiilor pe mai multe calculatoare. Replicarea datelor poate crește rapid volumul de date. În plus, sistemele de procesare a datelor mari ar trebui să stocheze rezultatele anterioare și recente, deoarece este normal ca acestea să fie comparate. Prin urmare, ele ar trebui să poată fi scalate prin distribuirea puterii de

procesare și a capacității de stocare pe mai multe computere pentru a obține o scalabilitate incrementală. Aceste sisteme ar trebui să poată să se scaleze fără nicio interacțiune umană [9].

### *3.2.4. Arhitectura Lambda*

Modelul de prelucrare batch și-a dovedit acuratețea prelucrării datelor. Cu toate acestea, acesta nu are rezultate productive într-o manieră de latență scăzută. Pe de altă parte, modelul de procesare a datelor în timp real, și-a dovedit capacitatea de a produce rezultate la latență redusă. Totuși, acest model poate produce în unele cazuri rezultate inexacte deoarece nu ia în considerare informațiile procesate deja în timp ce procesează date noi.

În încercarea de a combina cele mai bune din ambele, un model arhitectural numit *Arhitectura Lambda* a devenit destul de popular, înlocuind procesarea înceată de tip batch cu, componente suplimentare de procesare în timp real și astfel vizează ambele provocări ale Big Data, Volumul și Viteza, în același timp [10].

Așa cum este ilustrat în Figura 3.3, Arhitectura Lambda descrie un sistem care cuprinde o serie de layere, fiecare fiind responsabil de anumite funcționalități construite pe layerele de mai jos. Acest sistem cuprinde trei nivele: datele sunt stocate într-un nivel de persistență, cum ar fi HDFS, care este periodic procesat de către layer-ul de batch (de exemplu o data pe zi), în timp ce stratul de viteză gestionează porțiunea de date care nu a fost încă prelucrată de batch, și nivelul de servire le consolidează pe cele două prin combinarea ieșirii batch-ului cu viteza layer-ului.

Beneficiul evident al unui sistem de procesare a datelor în timp real, prin compensarea latenței ridicate a prelucrării batch, este dat de complexitatea sporită în dezvoltare, implementare și întreținere. Dacă layer-ul de batch este implementat cu un sistem care suportă atât procesarea batch, cât și procesarea streaming (de exemplu, Spark), nivelul de viteză poate fi implementat cu minimă cheltuială folosind streaming API (exemplu, Spark Streaming), pentru a utiliza business logic-ul și implementarea existentă [10].

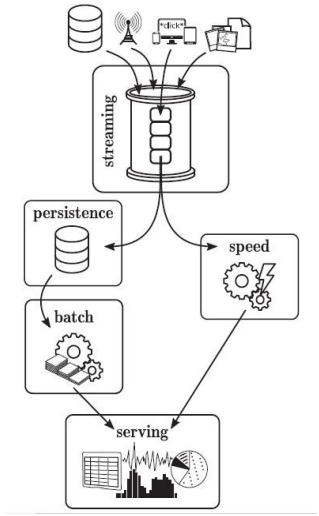



Figura 3.3 Arhitectura Lambda [10]

### *3.2.5. Analiza comparativă*

În urma descrierii celor două metode de procesare a datelor de tipul Big Data, cât și a modelului hibrid, putem spune că fiecare are atât avantaje, cât și dezavantaje. Modelul batch permite analiza unui volum mare de date simultan, însă cu o latență mare în ceea ce privește obținerea rezultatelor. Procesarea în timp real rezolvă această problemă, însă datele sunt disponibile doar pe o anumită fereastră de timp, cele vechi pierzându-se. O soluție pentru rezolvarea acestor dezavantaje, este oferită de arhitectura Lambda, care presupune implementarea simultană a doua metode de procesare diferite, lucru cu o complexitate semnificativă, care poate deveni o problemă prin necesitatea de resurse hardware mai mari.

În funcție de contextul în care se plasează problema, se poate alege modelul corespunzător. În cazul în care rezultatele analizei pot fi obținute zilnic, la sfârșitul zilei, latența nu reprezintă o problemă, deci se va alege procesarea batch. Dacă utilizatorii au nevoie de rezultate imedite – o situație foarte des întâlnită în aplicațiile sociale – procesarea de tip streaming este cea mai potrivită. Pentru sisteme cu o complexitate ridicată, în care combinarea celor două modele este necesară, arhitectura Lambda oferă soluția corespunzătoare.

## 3.3. Metode de clasificare a sentimentelor

### *3.3.1. Procesul de clasificare a textului*

Clasificarea textului face parte din procesul de extragere a datelor pentru a clasifica textul în funcție de conținutul acestuia. Se pot clasifica știri, articole sau cărți, în diferite categorii, în funcție de anumite caracteristici care au fost definite.

Atunci când este vorba de clasificarea textului, de obicei se face referire la clasificarea supravegheată care are două etape importante: etapa de instruire și etapa de testare. De obicei, prima etapă include crearea seturilor de date etichetate, pre-procesarea textului de antrenament, vectorizarea acestuia și instruirea clasificatorului. Etapa de testare constă în pre-procesarea textului pentru testare, vectorizarea lui și clasificarea [11]. Acest proces este prezentat în Figura 3.4.

**Crearea corpus-ului:** Colectarea textului în funcție de categorii. Fiecare text aparține unei categorii și este etichetat corect.

**Pre-procesarea**: Eliminarea tuturor elementelor nefolositoare din text, cum ar fi semnele de punctuație, *stop words* sau textul care nu poate fi citit.Acest pas este foarte important, deoarece fără el, formarea clasificatorului va fi afectată

**Vectorizarea textului**: Transformarea textului în vectori, pentru a putea fi recunoscut de către calculator. Textul va fi reprezentat ca vector de caracteristici bazat pe caracteristicile selectate.

**Instruirea clasificatorului**: Alegerea unui algoritm de clasificare și obținerea modelui de clasificare.

**Clasificarea**: După obținerea modelului, se poate aplica pe datele de testare, pentru a obține predicția de clasificare.

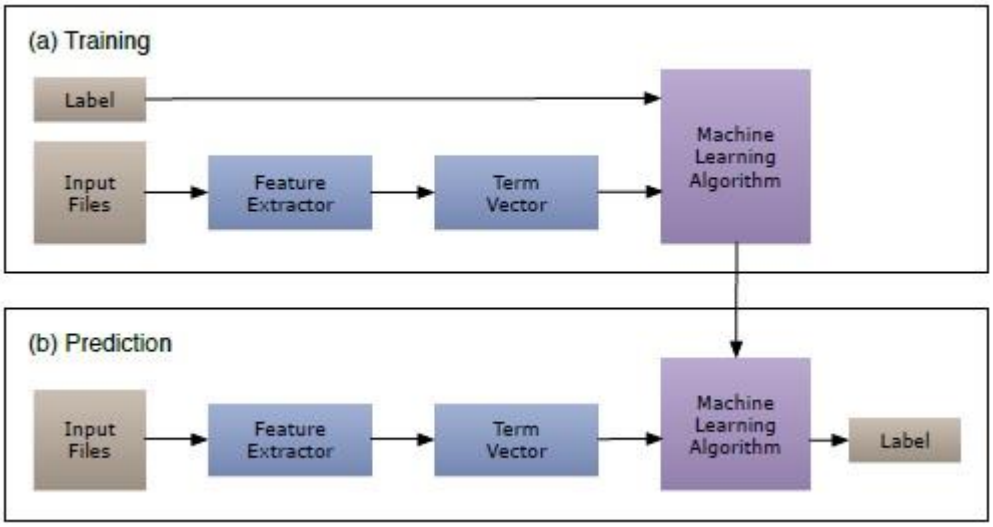



Figura 3.4 Clasificarea supravegheată a textului [11]

### *3.3.2. Clasificarea sentimentelor*

Multitudinea de date structurate și nestructurate stocate online reprezintă o metodă de cercetare a informațiilor pentru diverse domenii de activitate și cercetare.

În ultimii ani, efortul de cercetare din domeniul *data mining-ului* sau al analizei datelor a fost dedicat clasificării automate a textului pe baza unui subiect, a unui gen, a unei surse, a unei limbi etc., pentru a putea sorta și gestiona eficient datele. În unele aplicații este utilă analiza sentimentului exprimat în text.

Analiza sentimentelor este un proces de clasificare în care datele sunt clasificate ca având o polaritate pozitivă sau negative. În unele cazuri se poate indentifica și un sentiment neutru.

Există mai mulți algoritmi care sunt utilizați pentru clasificarea tradițională a textului și pot fi, de asemenea, aplicați pentru analiza sentimentului. Metodele pot fi împărțite în două mari grupe: tehnicile Machine Learning și tehnicile bazate pe Lexicon. Figura 3.5 ilustrează unele dintre cele mai utilizate tehnici de clasificare [12]. În continuare se vor detalia unele dintre ele.

### *3.3.3. Tehnici de clasificare a textelor*

#### *3.3.3.1. Machine learning (ML)*

Metoda ML utilizează algoritmi standard de machine learning pentru a rezolva analiza sentimentului ca o problemă de clasificare a textului, unde datele sunt etichetate cu una din cele trei clase: pozitive, negative sau neutre. Clasa "neutră" ar putea fi eliminată în anumite cazuri.

Tehnicile de machine learning pot fi în general divizate în două categorii: învățarea nesupravegheată și învățarea supravegheată.

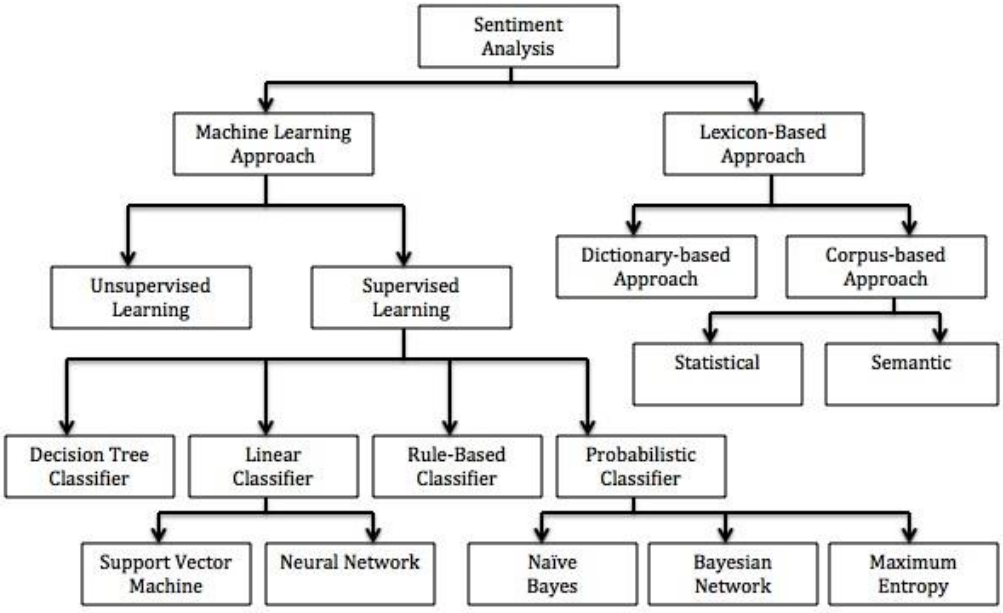



Figura 3.5 Tehnici de clasificare a textului [12]

Tehnicile ML nesupravegheate nu necesită seturi de date pentru antrenament. Prin urmare, metodele nesupravegheate sunt utile în cazurile în care datele de instruire nu sunt disponibile sau sunt greu de găsit. Aceste metode sunt utilizate pentru a grupa automat tipuri similare de obiecte de date într-o colecție de obiecte.

Învățarea supravegheată este metoda cea mai frecvent utilizată pentru analiza sentimentului. În următorul subcapitol (3.4.1.1), va fi descris una dintre cele mai folosite tehnici ML de supervised learning: Naïve Bayes.

În metodele de învățare supravegheată, se creează un model folosind un set de date de instruire, D = {X1,X2, ....,Xn}, unde X1,X2, ....,Xn sunt înregistrări separate, fiecare fiind etichetată manual cu o anumită clasă. Înregistrările sunt clasificate în funcție de caracteristicile sintactice sau lingvistice sau de o combinație a celor două. Un clasificator este apoi instruit folosind unul dintre algoritmi ML standard. După terminarea instruirii, clasificatorul este utilizat pentru a prezice eticheta unei instanțe de clasă necunoscută pe baza caracteristicilor selectate. Cheia pentru o mai mare precizie în analiza sentimentului sau orice problemă de clasificare a textului este selectarea unui set de caracteristici utile. Unele caracteristici posibile pentru clasificarea sentimentului sunt:

**Frecvența și prezența termenilor**: Numarul frecvenței apariției anumitor termenei specificați este cea mai comună caracteristică pentru problemele tradiționale de clasificare a textului. Acești termeni sunt cuvinte individuale, numite unigrame. Mulți cercetători au folosit n-grame (bigrame și trigrame) pentru a obține rezultate mai bune. Cu toate acestea, în cazul câtorva domenii, unigramele pot să funcționeze mai bine decât n-gramele, de exemplu, în cazul clasificării sentimentului filmelor [13]. Apariția

frecventă a termenilor s-ar putea să nu se dovedească întotdeauna a fi eficientă în orice cercetare da analiză a sentimentului, cum ar fi în cazul clasificării clasice a textului, identificarea subiectului unui document. În astfel de cazuri, prezența termenului ar putea fi o caracteristică mai bună pentru clasificarea sentimentului decât frecvența termenului [13]. Prezența termenului este o valoare binară atribuită unui termen și indică dacă este prezent sau nu în text.

**Etichetarea Părților de Vorbire**: EPV-ul cuvintelor individuale este o caracteristică frecvent utilizată în cercetarea analizei sentimentului. Unele părți de vorbire, cum ar fi adjectivele, sunt considerate a fi indicatori puternici de opinii sau de subiectivitate într-o propoziție. Una dintre cele mai vechi cercetări din analiza sentimentului a fost identificarea orientării semantice a diferitelor adjective [14] și găsirea ulterioară a unei corelații între prezența adjectivelor și subiectivitatea propoziției [15]. Cu toate acestea, adjectivele nu sunt singurii indicatori ai subiectivității propoziției. Cercetătorii au arătat că alte părți ale cuvântului, cum ar fi verbe (de exemplu „a iubi") și substantive (de exemplu „bijuterie") pot indica, de asemenea, subiectivitatea [13].

**Sentiment Lexicon**: Cuvintele care exprimă sentimente sau opinie au fost frecvent utilizate pentru a identifica eficient polaritatea propozițiilor. De exemplu, "uimitor", "bun", "fericit" indică sentimentul pozitiv, în timp ce "sărac", "trist", "terifiant" indică sentimentul negativ. Chiar dacă majoritatea cuvintelor care exprimă sentiment sunt adjective sau adverbe, ele pot fi, de asemenea, și substantive sau verbe. În afară de cuvintele simple, există și expresii și locuțiuni care exprima sentimente, cum ar fi "a fi în al nouălea cer". Cuvintele, frazele și expresiile formează împreună un lexicon de sentiment sau un lexicon de opinie care poate fi folosit pentru a identifica polaritatea sentimentului. De multe ori, un astfel de lexicon nu este suficient pentru a determina polaritatea unui text, deoarece pot exista propoziții fara cuvinte care exprimă sentimente sau cuvintele pozitive sau negative ar putea însemna contrariul în anumite domenii.

**Reguli pentru opinii:** Există multe expresii și enunțuri compuse care pot indica polaritatea propozițiilor în funcție de anumite reguli de compoziție sau de cunoștințe într-un anumit domeniu. În afară de cuvintele sau frazele care exprimă sentiment, aceste reguli pot fi de asemenea folosite pentru creșterea acurateței rezultatelor clasificării sentimentului.

**Modificatori de sentiment**: Modificatorii de sentimente schimbă polaritatea cuvintelor pozitive în negative și invers. Acești modificatori sunt caracteristici foarte importante care trebuie luate în considerare în timpul analizei de sentimente. Cuvintele de negare, de exemplu "nu", "non", "ne" sunt cele mai comune tipuri de astfel de modificatori. În exemplul urmator, " Alinei nu îi plac sporturile extreme ", dacă cuvântul de negare nu este atașat ca o caracteristică de antrenament, atunci ar putea fi identificat în mod greșit drept pozitiv din cauza prezenței cuvântului de sentiment pozitiv.

**Emoticon-uri**: Emoticoanele sunt simboluri folosite în comunicarea digitală, cum ar fi email-urile, chat-ul sau micro-blogging-ul pentru a transmite emoțiile umane. Aceste

emoticoane pot fi folosite pentru a identifica polaritatea sentimentului. De exemplu, o față zâmbitoare sau o inimă indică un sentiment pozitiv, în timp ce o față încruntată sau plângând indică un sentiment negativ. Emoticoanele pot fi o caracteristică eficientă pentru a identifica sentimentele datelor din mass-media sociale, recenzii sau email-uri.

3.3.3.1.1. Naive Bayes

Naive Bayes este o colecție de mai mulți algoritmi bazați pe Teorema Bayes[6], care este o teorie a probabilității care prezice probabilitatea întâmplării unui eveniment bazându-se pe noi dovezi legate de acel eveniment. Prin urmare, în conformitate cu Teorema Bayes, probabilitatea ca evenimentul A să aibă loc, știind că B este Adevarat, P(A|B) este:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

unde

1. P(B) = 0,
2. P(A) și P(B) sunt probabilitățiile A și B, care se întâmplă independent unul de celălalt
3. P(B|A) este probabilitatea ca evenimentul B să se întâmple dacă evenimentul A este adevărat.

În cazul Naive Bayes pentru clasificarea documentelor, „A” reprezintă „eticheta” sau „clasa”, în timp ce „B” reprezintă „caracteristica”. Principiul fundamental al clasificatorilor Naive Bayes este acela că toate caracteristicile folosite pentru a clasifica un document se presupune a fi independente una de cealaltă [17]. De exemplu, dacă caracteristicile „rosu”, „rotund” și „dulce” sunt folosite pentru a clasifica un fruct ca un măr, atunci Naive Bayes tratează fiecare caracteristică independent în calcularea probabilității ca un fruct să fie măr, indiferent de relațiile dintre caracteristici. Cu toate acestea, în multe scenarii din viața reală, caracteristicile selectate pot să nu fie independente unele de altele și acesta este un dezavantaj major al algoritmului Naive Bayes. În ciuda acestui dezavantaj, Naive Bayes depășește mulți alți algoritmi și este ușor de înțeles și ușor de construit pe un set de date de antrenament mic.

Naive Bayes este folosit pentru a prezice clasa unui document folosind probabilitatea. Pentru a explica Naive Bayes, se poate considera următorul exemplu: să prespunem că profesorul îi roagă pe studenți să redacteze o scrisoare, un eseu despre modelul lor în viată sau un eseu în care să argumenteze că educaţia primită în şcoală influenţează dezvoltarea ulterioară a tinerilor. Vrem să clasificăm răspunsurile studenților în clasele „Scrisoare”, „Eseu” și „Argument” în funcție de trei caracteristici sau cuvinte care apar în răspunsuri – „stimă”, „lider” și „educație”. Presupunem că avem 100 de studenți pentru instruire. Tabelul 3.1 arată setul de date de antrenament și numerele indică câte răspunsuri dintr-o clasă conțin caracteristica corespunzătoare.

[6] https://ro.wikipedia.org/wiki/Teorema_lui_Bayes

Cu ajutorul teoremei lui Bayes și a informațiilor furnizate în tabelul de mai jos, putem prezice clasa unui răspuns dat de un nou student, conținând cuvântul „stimă", „lider" și „educație".

| Răspuns student | **Stimă** | **Lider** | **Educație** | **Total** |
|---|---|---|---|---|
| **Scrisoare** | 15 | 0 | 10 | 30 |
| **Eseu** | 35 | 40 | 45 | 50 |
| **Argument** | 15 | 10 | 5 | 20 |
| **Total** | 65 | 50 | 60 | 100 |

Tabel 3.1 Setul de date pentru exemplul Naive Bayes

În conformitate cu ipoteza „naive" cum că toate caracteristicile sunt independente, o ecuație de probabilitate poate fi dedusă din teorema lui Bayes după cum urmează:

$$P(clasă|caracteristici) = \frac{P(c1|clasă)P(c2|clasă) \ldots . P(cN|clasă)\ P(clasă)}{P(caracteristici)},$$

unde c1, c2, cN sunt setul de caracteristici utilizate pentru clasificarea documentelor în clase.

Probabilitățile respective, dacă răspunsul elevului este „Scrisoare", „Eseu" sau „Argument", se pot calcula după cum urmează:

$$P(Scrisoare|Stimă, Lider, Educație)$$

$$= \frac{P(Stimă|Scrisoare)P(Lider|Scrisoare)\ P(Educație|Scrisoare)\ P(Scrisoare)}{P(Stimă), P(Lider), P(Educație)}$$

$$= \frac{0x0.5x0.333x0.3}{P(evidență)} = 0$$

$$P(Eseu|Stimă, Lider, Educație)$$

$$= \frac{P(Stimă|Eseu)P(Lider|Eseu)\ P(Educație|Eseu)\ P(Eseu)}{P(Stimă), P(Lider), P(Educație)}$$

$$= \frac{0.8x0.7x0.9x0.5}{P(evidență)} = \frac{0.252}{P(evidență)}$$

$$P(Argument|Stimă, Lider, Educație)$$

$$= \frac{P(Stimă|Argument)P(Lider|Argument)\ P(Educație|Argument)\ P(Argument)}{P(Stimă), P(Lider), P(Educație)}$$

$$= \frac{0.5x0.75x0.25x0.2}{P(evidență)} = \frac{0.01875}{P(evidență)}$$

Din calculele de mai sus vedem că probabilitatea, P(Eseu|Stimă, Lider, Educație) este mai mare decât probabilitățile, P(Scrisoarea|Stimă, Lider, Educație și P(Argument| Stimă, Lider, Educație). Prin urmare, fiind mult mai probabil ca noul răspuns să fie un eseu despre modelul în viață, în conformitate cu Teorema Bayes, clasificatorul Naive Bayes îl va eticheta cu „Eseu”.

Naive Bayes pot fi folosite pentru a rezolva diferite probleme de clasificare, cum ar fi detectarea spam-ului, categorizarea subiectului și analiza sentimentului.

### *3.3.3.2. Tehnici bazate pe lexicon*

Metodele bazate pe lexicon sunt utilizate pe scară largă în sarcinile de analiză a sentimentului. În această abordare, expresii de cuvinte, fraze și locuțiuni pozitive și negative sunt folosite pentru a clasifica sentimentele., formând împreună un lexicon de opinie. Este posibil ca acest lexicon să fie creat manual, deși această metodă ar dura mult timp, este ineficientă și predispusă la erori. De aceea, există tehnici de creare automată a unui lexicon de opinie, pentru analiza de sentimente.

Tehnicile pot fi împărțite în două categorii: abordarea bazată pe un dicționar și abordarea bazată pe un corpus.

**Abordarea bazată pe dicționar**: creează lexiconul de opinie într-un proces iterativ. În prima iterație se selectează manual un set mic de cuvinte de opinie cu polaritate pozitivă sau negativă cunoscută. Apoi, sinonimele și antonimele pentru cuvintele selectate sunt căutate și colectate. Această iterație continuă până când nu se găsesc cuvinte noi. Cuvintele selectate și căutate sunt adăugate la lista de seed. În cele din urmă, lista de seed este verificată manual pentru eventuale erori. Dezavantajul abordării bazate pe dicționar este că nu este adecvată pentru găsirea polarității unui cuvântul într-un context sau domeniu specific.

**Abordarea bazată pe corpus**: spre deosebire de abordarea bazată pe dicționar, abordarea bazată pe corpus nu se confruntă cu problema găsirii orientării cuvântului în funcție de context sau de domeniu. În această abordare, mai întâi este selectată o listă de cuvinte de opinie. Apoi modelele sintactice care apar în asociere cu lista seed de cuvinte sunt folosite pentru a determina orientarea altor cuvinte. De exemplu, adjectivele care apar după conjuncția “și” cu un cuvânt din lista seed sunt considerate a avea aceeași orientare sentimentală. În mod similar, cuvintele "dar" sau "totuși" indică o schimbare a sentimentului. Un corpus mare este folosit pentru a afla dacă două cuvinte combinate au aceleași sentimente. Legătura dintre cuvinte formează un grafic. Clustering-ul este aplicat pe grafic pentru a crea o listă de cuvinte pozitive și o altă listă de cuvinte negative. Un dezavantaj al abordării bazate pe corpuri este că necesită un corpus foarte mare pentru învățare, care este dificil de pregătit. Prin urmare, este posibil să nu fie la fel de eficace ca abordarea bazată pe dicționar dacă este aplicată singură.

În concluzie, pentru ca setul de date să fie procesat și analizat în timp real, se vor folosi tehnici moderne de procesare streaming, alături de metode eficiente de analiză a

sentimentelor. În capitolele 4 și 5 vor fi descrise aceste tehnici atât teoretic, cât și practic, în implementarea unui sistem de procesare și analiză a datelor de pe Twitter în timp real.

# Capitolul 4. Analiză şi Fundamentare Teoretică

Lucrarea prezentă are ca focus analiza de streaming, deocarece modelul de procesare batch nu este adevat pentru implementarea sistemului propus, un obiectiv major reprezentând procesarea datelor și obținerea rezultatelor în timp real. De asemenea, complexitatea arhitecturii lamba nu este necesară in această abordare, deoarece layer-ul de procesare batch nu aduce beneficii semnificative. Prin urmare, în acest capitol vor fi prezentate în detaliu o serie de tehnologii și framework-uri de procesare big-data sub formă de stream-uri, dezvoltate de diferite companii pentru a obține o analiză real-time, dar și framework-ul care stă la baza lor, și anume Apache Hadoop.

## 4.1. Apache Hadoop

Hadoop este un proiect open-source creat de Doug Cutting și este cea mai populară implementare a MapReduce-ului, propunându-și realizarea de procesări distribuite a unor seturi de date de dimensiuni mari, rulând pe mai multe clustere [4].

Clusterul Hadoop este compus din noduri master și noduri slave [18]. Pot exista una sau mai multe instanțe ale nodului principal, într-o implementare Hadoop, pentru a elimina riscul de a avea un singur punct de eșec. Clusterul Hadoop poate avea sute sau mii de noduri.

Un nod master are trei roluri importante – JobTracker, TaskTracker și NameNode. JobTracker-ul interacționază cu aplicațiile clienților și distribuie job-uri MapReduce celorlalte noduri. TaskTracker primeste sarcini-uri de la JobTracker, sarcini ca map și reduce. NameNode-ul stochează și ține evidența arborilor directoarelor de fișiere și fișierele metadata din cluster. Hadoop are propriul său sistem de fișiere cunoscut sub numele de Sistemul de fișiere distribuite Hadoop (HDFS) [7] inspirat de sistemul de fișiere Google (GFS)[8].

Nodurile workers procesează și analizează datele executând funcțiile map și reduce. Fiecare nod worker este alcătuit din două roluri DataNode și TaskTracker. Sarcina DataNode-ului este de a stoca datele în HDFS și de a le replica în clustere. În HDFS fișierele sunt stocate la blocuri, fară ca datele să fie stocate în cache. Toate datele sunt replicate în trei DataNode-uri pentru a asigura fiabilitate si acces ușor la ele. Arhitectura și implementarea Hadoop sunt ilustrate în Figura 4.1.

Hadoop lucrează foarte bine în procesarea unor seturi mari de date. Chiar dacă framework-ul este foarte bun pentru ce a fost dezvoltat, are câteva limitări. În primul rând, deoarece nu este tocmai ușor de transpus o problemă într-un set de perechi cheie-valoare, ceea ce înseamnă că modelul de date este restrictiv. Hadoop poate procesa doar seturi de date gata recepționate, deci MapReduce poate procesa date doar în maniera batch. Datele rezultate pot fi produse cu întârzieri destul de mari, acest lucru depinzând de dimensiunea blocurilor de date care trebuie procesate și de puterea de calcul a

[7] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[8] https://en.wikipedia.org/wiki/Google_File_System

sistemului. Toate aceste limitări au dus la dezvoltarea unor noi metode de procesare, precum cea de tip streaming, adică în timp real.

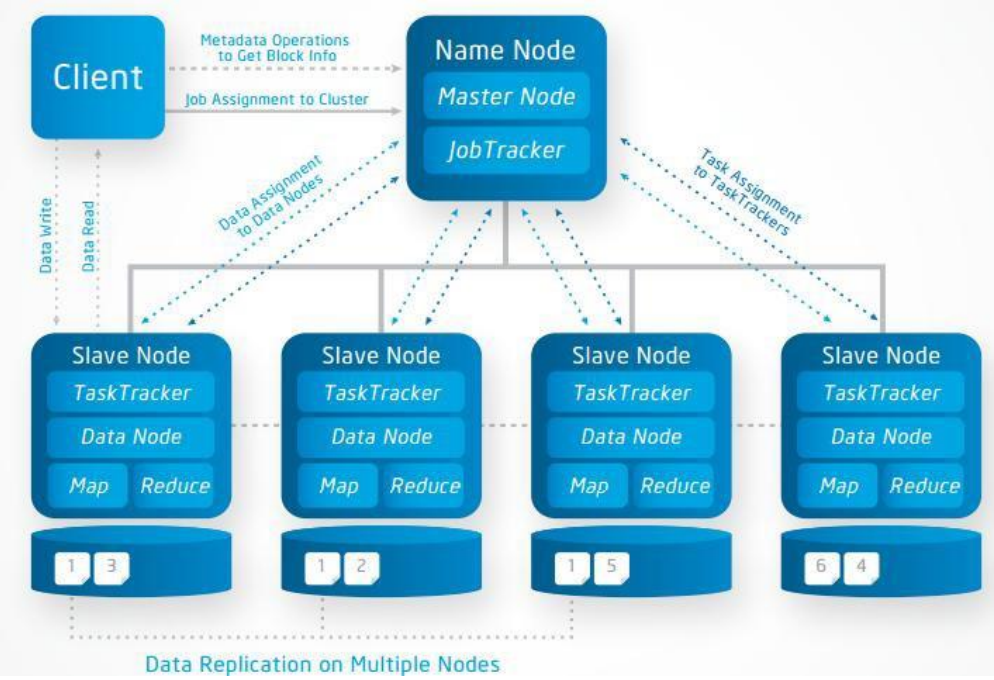



Figura 4.1 Arhitectura și implementarea Hadoop [9]

**NoSQL**

Bazele de date SQL relaționale pot fi asociate cu tehnologii Big Data, cum ar fi Hadoop, pentru a oferi capabilități puternice de analiză. Cu toate acestea, apariția și creșterea bazelor de date NoSQL[10] din ultimul deceniu au afectat în mare măsură progresul Big Data Analytics. NoSQL înseamnă pur și simplu că nu utilizează SQL ca limbaj de interogare spre deosebire de bazele de date relaționale tradiționale și este, în esență, un sistem de baze de date non-relaționale. Sistemul de management al bazelor de date relaționale (RDBMS)[11] stochează și interoghează date structurate grupate în tabele. Tabelele reprezintă entități unice, de exemplu, "studenți" și "școli" pentru un sistem de management universitar și au o schemă fixă predefinită. Tabelele sunt formate din coloane unice care reprezintă proprietățile entității și rândurile care reprezintă înregistrări individuale. Una dintre provocările primare ale RDBMS este incapacitatea sa de a mări orizontal, având în vedere formatarea sa structurată și aderarea la proprietățile ACID:

[9] http://www.rosebt.com/blog/hadooparchitecture-and-deployment
[10] https://en.wikipedia.org/wiki/NoSQL
[11] https://en.wikipedia.org/wiki/Relational_database_management_system

**Atomicitate**: Toate sau nimic dintr-o tranzacție va reuși.
**Consistența**: O tranzacție va duce baza de date dintr-o stare consecventă la o alta.
**Izolarea**: Toate tranzacțiile au fost independente una de cealaltă.
**Durabilitate**: O tranzacție de succes va persista, chiar dacă este aplicația. este închisă

NoSQL abordează provocările din RDBMS și face ca scalarea să fie fezabilă. Nu se bazează pe proprietățile ACID care sunt incompatibile cu cerințele de disponibilitate și de performanță ale aplicațiilor la scară largă. NoSQL este construit pe teorema CAP:
**Consistență**: garantează că nodurile de stocare conțin elemente de date identice pentru aceeași bucată de date la un moment dat.
**Disponibilitate**: garantează că fiecare cerere primește un răspuns cu privire la faptul că a reușit sau nu a reușit.

Dispozitivele de rețea sunt componente fizice, care sunt responsabile pentru interacțiunea și comunicarea dintre calculatoarele dintr-o rețea[12]. Aceste componente pot eșua, după care rețeaua este împărțită în sub-rețele care determină partiționarea în rețea (**toleranță la partiționare**) [1].

Teorema CAP forțează proiectanților de sisteme să aleagă între disponibilitate și consistență. Disponibilitatea se concentrează pe construirea unui sistem disponibil și apoi pe încercarea de a o face cât mai consistent posibil. Acest model poate duce la stări inconsistente între nodurile sistemului care pot determina rezultate diferite pe diferite computere. Cu toate acestea, face sistemul disponibil tot timpul. Pe de altă parte, consistența se concentrează mai întâi pe construirea unei stări consistente și apoi pe încercarea de a o face cât mai disponibilă. Acest model poate duce la un sistem indisponibil atunci când se partiționează rețeaua. Se garantează că un sistem are date identice când sistemul este disponibil. Figura 4.2 ilustrează relațiile dintre teorema CAP, proprietățiile ACID și baze de date NoSQL.

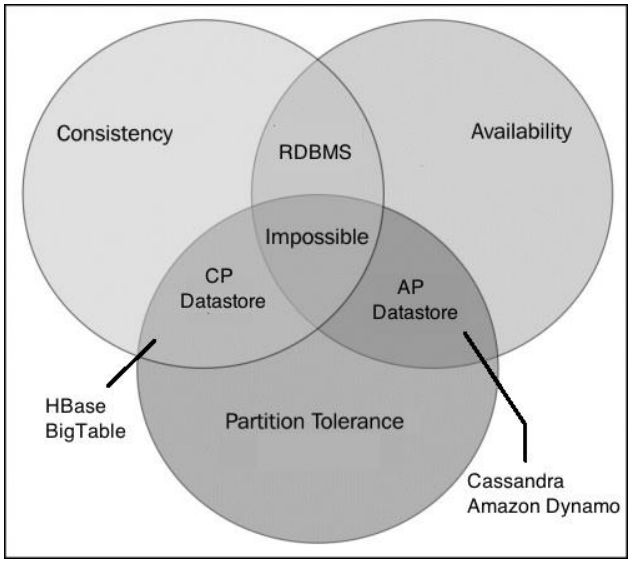



Figura 4.2 Relațiile dintre CAP, ACID și NoSQL

[12] https://en.wikipedia.org/wiki/Networking_hardware

## 4.2. Framework-uri de procesare streaming

Motivul care a dus la dezvoltarea unui număr relativ mare de framework-uri pentru prelucrare de tip streaming, a fost nevoia de a trece peste limitările impuse de către analiza batch, și anume latența mare pe care aceasta o introduce. Procesarea în timp real presupune procesarea unui flow continuu de date, infinit, astfel încât rezultatele obținute să fie disponibile cu o latență minimă, de ordinul a câtorva secunde, și să fie direct accesibile de către utilizatorul final. Prin urmare, volumul de date acumulat pentru analiză la un moment de timp nu este foarte mare, însă viteza pentru Big Data este considerabilă.

În continuare vor fi descrise succinct 3 astfel de framework-uri, Apache Spark, Apache Flink și Apache Beam.

**Apache Spark**[13] este un framework dezvoltat pentru calculul distribuit cu disponibilitate de scalare a memoriei, care permite dezvoltarea de aplicații în mai multe limbaje de programare, cum ar fi Java, Pyhton, Scala și R. Scopul este de a rula aceste aplicații în mod paralel între mai multe clustere. Este compatibil cu Apache Mahout, care folosește motorul de procesare în loc de MapReduce. Prin urmare, acesta oferă performanțe computaționale și eficiență în timp, în timpul procesării datelor mari.

Apache Spark suportă patru sub-aplicații care sunt modulare, SQL, MlLib, GraphX și Streaming. Ele sunt interoperabile și fiecare dintre acestea poate fi extins prin proiecte open-source din ecosistemul Apache Hadoop. Deci, există o legătură strânsă între Spark și Hadoop, diferind puțin arhitectura.

Comparațiile între Spark și Hadoop arată că utilizând Spark, programele rulează până la de 100 de ori mai rapid decât Apache Hadoop MapReduce cu anumite setări ale memoriei, sau până la de 10 ori mai rapid în local storage [20].

**Apache Flink**[14] este o platformă open source pentru procesarea streaming și prelucrarea în batch-uri a seturilor de date. Baza este realizată de motorul de streaming, scris în Java și Scala [21] și oferă distribuție de date, comunicare, toleranță la erori în stream-urile de date distribuite. Flink suportă biblioteci suplimentare pentru a viza scenarii specifice în cadrul procesării datelor și data mining. FlinkML este o bibliotecă de machine elarning pentru Flink, care oferă API pentru construirea algoritmilor de învățare automată scalabili [22].

În ciuda faptului că platforma nu include niciun sistem de stocare a datelor, Flink poate fi integrat cu alte proiecte open source și poate stoca date de intrare pe sisteme distribuite, cum ar fi HDFS sau Hbase. Conținutul fluxurilor de date poate curge în Flink printr-o platformă de mare viteză și latență redusă, Apache Kafka.

Flink și Spark pot fi comparate unul cu celălalt, deoarece ambele pot rula peste Hadoop, ambele suportă biblioteci suplimentare pentru machine learning, vizualizări grafice și pot efectua calcule în memorie.

Apache Spark și Flink sunt scalabile și tolerante la erori. Ele procesează date mari cu viteză mare de procesare și latență redusă bazată pe ferestre de timp. Cu toate acestea,

[13] https://spark.apache.org/
[14] https://flink.apache.org/

motoarele de streaming Apache Spark și Flink nu oferă o fereastră bazată pe diferiți factori, cum ar fi sesiunile, categoriile și utilizatorii. Ele emit o fereastră de date după un anumit număr de înregistrări sau o cantitate de timp.

**Apache Beam**[15] este un proiect open source bazat pe modelul Google Cloud Dataflow [22]. Este un framework unificat care are permite o exprimare relativ simplă a calculului paralel într-un mod care este independent de motorul de execuție fundamental. Oferă posibilitatea de a controla cantitatea de latență și corectitudine pentru orice domeniu specific de probleme. Apache Beam permite calcularea unor surse de date nelimitate și neordonate bazate pe timpul evenimentului. În plus, separă noțiunea logică de prelucrare a datelor de la implementarea fizică de bază. Prin urmare, Apache Beam permite procesarea în batch, micro-batch sau streaming folosind un model de programare unificat. Motorul de streaming grupează fluxuri de date nelimitate în ferestrele de date. Apache Beam suportă trei tipuri de ferestre pentru a face față datelor nelimitate: fix, sliding și sesiuni. Acest cadru procesează fiecare înregistrare separat, pentru a o procesa în fereastra de date corectă.

Fiind un model unificat pentru a efectua procesare batch și streaming pe motoare de procesare distribuite, cum ar fi Apache Spark și Apache Flink.

După cum se poate observa și în Figura 4.3, acest model permite efectuarea acelorași conducte pe diferiți alergători cu modificări minime ale codului. Acesta susține patru runneri diferiți: DirectPipelineRunner, DataflowPipelineRunner, FlinkPipeline Runner și SparkPipelineRunner. DirectPipelineRunner permite utilizatorilor să execute pipeline-ul pe o mașină locală. DataflowPipelineRunner execută pipeline-ul pe Google Cloud Dataflow. FlinkPipelineRunner permite utilizatorilor să execute pipeline-uri pe un cluster Flink. SparkPipelineRunner permite utilizatorilor să execute pipeline-uri pe un cluster Spark. Apache Beam este încă în faza de implementare. Prin urmare, funcționalitățile modelului Cloud Dataflow nu sunt pe deplin suportate de către runner-ii Spark și Flink.

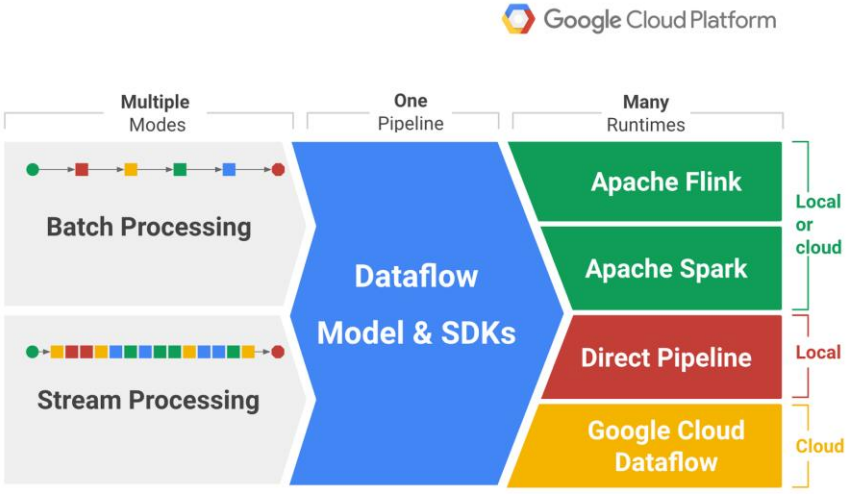



Figura 4.3 Arhitectura Apache Beam [16]

[15] https://beam.apache.org/
[16] http://www.googblogs.com/tag/dataflow/

### *4.2.1. Descriere generală*

Apache Spark este un framework de cluster computing dezvolat în Scala. Este un motor rapid și general pentru procesarea paralelă a datelor la scară largă.

Spark extinde modelul popular Map Reduce. Putem spune că una dintre cele mai importante caracteristici a Spark-ului este faptul că lucrează în memorie. Acest fapt implică faptul că este mai eficient să facă calcule, precum algoritmi iterativi, interogări interactive și procesarea fluxului de date datorită evitării blocajului de citire/scriere pe disc.

Proiectul Spark conține mai multe componente integrate: Spark SQL, Spark Streaming, MLib și GraphX. Aceste componente au fost construite pentru a lucra împreună ori de cate ori se dorește acest lucru. Astfel, ele pot fi combinate ca biblioteci într-un proiect software. Spark este nucleul lor, responsabil pentru programare, distribuire și monitorizarea aplicațiilor peste cluster.

Aceste componente diferite ale stivei Spark, așa cum se arată Figura 4.4, sunt descrise în subcapitolele ce urmează.

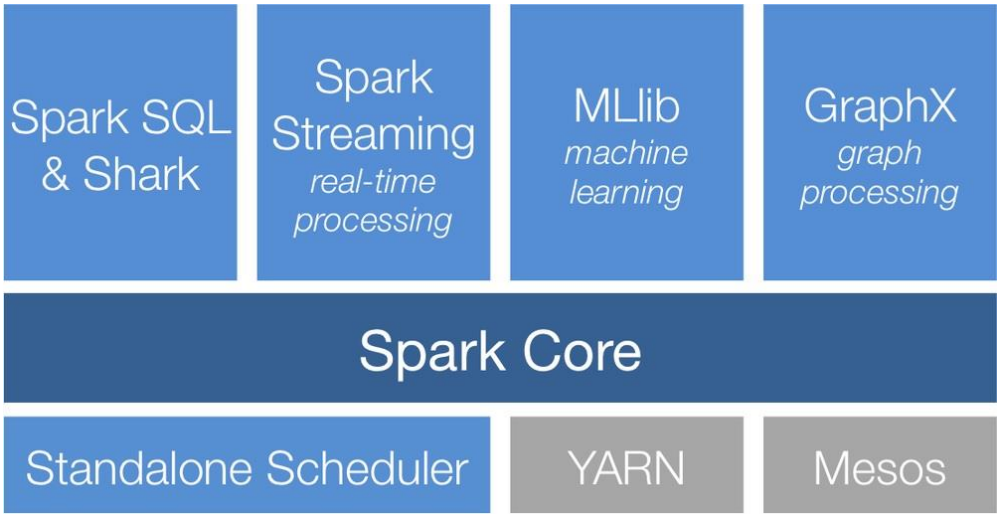



Figura 4.4 Arhitectura Spark[17]

### *4.2.2. Concepte de bază*

Pentru a putea înțelege cum funcționează Spark, trebuie cunoscuți următorii termeni:

- Job: este un calcul paralel care citește câteva intrări din HDFS, HBase, Cassandra, local, și efectuează unele calcule asupra datelor (de ex. save, collect, etc)

[17]https://matthewdixon2015.wordpress.com/2015/01/06/using-apache-spark-for-cva-pricing-of-derivatives/

- Tasks: Fiecare etapă are câteva task-uri, un task pe partiție. Un task este executat pe o partiție de date pe un executor (mașină).
- Executor: Proces responsabil pentru executarea unui task

Numărul de executori folosiți în programe este direct legat de cantitatea de timp în care task-ul unui job este executat. Figura 4.5 arată relevanța folosirii unui număr corect de executori și cum se reduce exponential timpul de execuție al job-urilor. Există însă un prag în care creșterea numărului de executori nu mai reduce timpul de execuție.

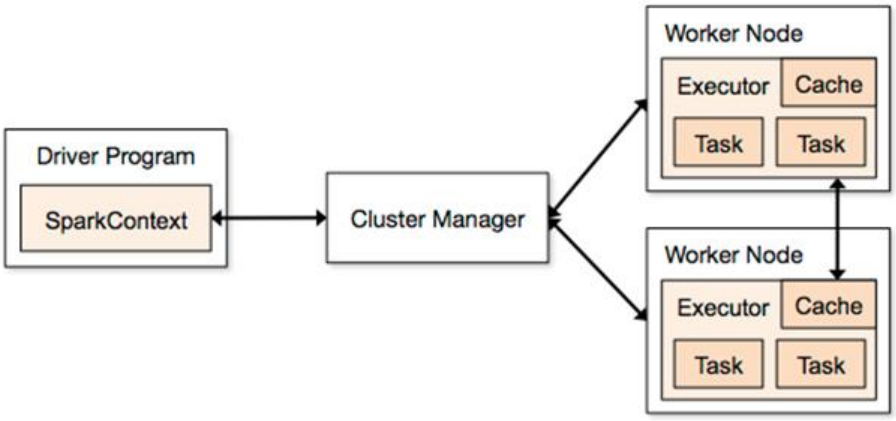



Figura 4.5 Modul general de prezentare a unui cluster Spark[18]

- Driver: Programul/procesul responsabil pentru rularea job-ului peste motorul Spark
- Master: Mașina pe care ruleaza programul Driver
- Slave: Mașina pe care ruleaza programul Executor
- Stages: Job-urile sunt împărțite în etape (stages). Etapele sunt clasificate ca etape Map sau Reduce. Ele sunt împărțite pe baza limitelor computaționale, toate calculele/operatorii nu pot fi actualizate într-o singură etapă. Se întâmplă în mai multe etape

### *4.2.2.1. Resilient Distributed Datasets*

Modelul MapReduce are lipsuri în ceea ce privește aplicațiile care aplică iterativ o anumită funcție pe aceeași sursă de date la fiecare pas de procesare, cum ar fi aplicațiile de machine learning sau de procesare a grafurilor (graph processing). Aceste tipuri de

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

aplicații necesită un model de procesare iterativ. Modelul Map Reduce se descurcă greu cu aplicații iterative deoarece el stochează datele pe hard disk-uri și reîncarcă aceleași date direct de pe hard disk la fiecare pas.

Sarcinile (jobs) MapReduce se bazează pe replicarea fișierelor în sistemul distribuit de fișiere pentru recuperarea erorilor (fault recovery). Acest mecanism adaugă supraîncărcare semnificativă la transmisia de rețea (acest mecanism încarcă semnificativ transmisia de rețea) atunci când e vorba de replicarea unor fișiere de dimensiune mare.

RDD este memorie abstractă tolerantă la erori care evită replicarea datelor și minimizează căutarea pe disk. Permite aplicațiilor să stocheze datele în memorie în diferite etape de procesare, ceea ce duce la o accelerare substanțială a reutilizării viitoare a datelor. În plus, RDD-urile țin minte operațiile folosite pentru a le putea construi. Când are loc o eroare, RDD-urile pot reconstrui datele cu supraîncărcare minimă a rețelei.

RDD-urile oferă unele restricții privind folosirea memoriei partajate pentru a permite toleranța scăzută a erorilor de supraîncărcare. Acestea sunt concepute a fi read-only și partiționate. Fiecare partiție conține înregistrări care pot fi create prin operații deterministe numite transformări.

Transformările includ operații de tipul map, reduce, groupBy și join. RDD-urile pot fi create prin alte RDD-uri existente sau print-un set de date într-un spațiu de stocare stabil. Aceste restricții facilitează reconstrucția partițiilor pierdute deoarece fiecare RDD are suficiente informații din alte RDD-uri despre cum să le reconstruiască. Când există o cerere de păstrare (cache) a RDD-ului, motorul de procesare Spark stochează partițiile implicit în memorie. Cu toate acestea, partițiile pot fi stocate pe hard disk atunci când nu este disponibil spațiu suficient de memorie sau utilizatorii solicită memorarea (cache) pe hard disk. În plus, Spark stochează RDD-urile de dimensiuni mari pe hard disk, deoarece acestea nu pot fi păstrate în memoria cache. RDD-urile pot fi împărțite pe baza unei chei asociate fiecărei înregistrări, după care acestea sunt distribuite în mod corespunzător.

Fiecare RDD constă într-un set de partiții, un set de dependințe cu RDD-ul părinte, o funcție de calcul și metadate despre schema de partiționare. Funcția definește modul în care un RDD este construit din RDD-ul părinte. De exemplu, dacă un RDD este reprezentat printr-un fișier HDFS, atunci fiecare partiție reprezintă câte un bloc al fișierului HDFS. Metadatele vor conține informații despre fișierul HDFS, cum ar fi locația și numărul de blocuri. Acest RDD este creat dintr-un spațiu de stocare stabil, deci nu are dependințe cu RDD-urile părinte. O transformare poate fi efectuată pe unele RDD-uri, care reprezintă diferite fișiere HDFS. Output-ul transfornării este un nou RDD. Acest RDD va conține un set de partiții în funcție de natura transformări. Fiecare partiție va depinde de un set de alte partiții din RDD-ul părinte. Aceste partiții părinte sunt folosite pentru a recompune partiția copil când este necesar.

Spark împarte dependințele cu RDD-ul părinte în două tipuri: înguste (narrow) și largi (wide). Dependințele înguste indică faptul că fiecare partiție a copilului RDD depinde de un număr fix de partiții din părintele RDD, cum ar fi transformările *map*. Dependințele largi indică faptul că fiecare partiție din copilul RDD poate depinde de toate partițiile părintelui RDD, cum ar fi transformările *groupByKey*. Această clasificare ajută Spark să îmbunătățească mecanismul de execuție și recuperare. RDD-urile cu dependințe înguste pot fi compuse într-un singur cluster, cum ar fi operația *map* urmată de operația *filter*. În mod contrariu, depedințele largi necesită ca datele din partițiile

părinte să fie amestecate între diferite noduri. Mai mult, o eroare a nodului poate fi recuperată mai eficient cu o dependință îngustă, deoarece Spark va recompune un număr fix de partiții pierdute. Aceste partiții pot fi recompuse în paralel pe noduri diferite. În schimb, un singur eșec al nodului ar putea necesita o execuție completă în cazul dependințele largi. În plus, această clasificare ajuta Spark să programeze planul de execuție și să ia checkpoint-uri.

Planificatorul de sarcini al Spark-ului folosește structura fiecărui RDD pentru a optimiza planul de execuție. Obiectivul său este de a construi un graf aciclic direct (DAG - Directed Acyclic Graph) al etapelor de calcul pentru fiecare RDD. Fiecare etapă conține cât mai multe transformări posibile cu dependințe înguste. O nouă etapă începe atunci când există o transformare cu dependințe largi sau o partiție păstrată în, care este stocată în alte noduri. Planificatorul plasează sarcini bazate pe localizarea datelor pentru a minimiza comunicarea în rețea. Dacă o sarcină are nevoie să proceseze o partiție din cache, planificatorul trimite această sarcină unui nod care are partiția cache-uită.

#### *4.2.2.2. Discretized Streams*

Spark își propune să obțină avantajele modelului de procesare batch pentru a construi un motor de streaming. Acesta implementează modelul de fluxuri discretizate (D-streams) pentru a face față calculului fluxului. Modelul D-Streams descompune un flux de date într-o serie de loturi mici la intervale mici de timp numite Micro-batch-uri. Fiecare micro-batch își stochează datele în RDD-uri. Apoi, micro-batch-ul este procesat și rezultatele sale sunt stocate în RDD-urile intermediare.

D-Streams oferă două tipuri de operatori pentru a construi programe de streaming: ieșire și transformare. Operatorii de ieșire permit programelor să stocheze date pe sisteme externe, cum ar fi HDFS și Apache Kafka. Există doi operatori de ieșire: save și foreach. Operatorul de salvare scrie fiecare RDD într-un D-stream către un sistem de stocare. Operatorul foreach execută o funcție definită de utilizator (UDF) pe fiecare RDD într-un D-stream. Operatorii de transformare permit programelor să producă un nou D-stream de la unul sau mai multe fluxuri părinte. Spre deosebire de modelul MapReduce, D-Streams suportă atât operații stateless, cât și stateful. Operațiunile stateless acționează independent pe fiecare micro-lot, cum ar fi map și reduce. Operațiile stateful funcționează pe mai multe micro-batch-uri -cum ar fi agregarea pe o fereastră sliding.

D-streams oferă operatori stateful care sunt capabili să lucreze pe mai multe micro-batch-uri, cum ar fi Windowing și Incremental Aggregation. Operatorul windowing grupează toate înregistrările dintr-o anumită perioadă de timp într-un singur RDD. De exemplu, dacă o fereastră este la fiecare 10 secunde și windowing-ul este o dată la o secundă, atunci un RDD va conține înregistrări din intervale [1,10], [2,11], [3,12] etc. Aceleași înregistrări apar la mai multe intervale și prelucrarea datelor se repetă. Pe de altă parte, operatorul de agregare incrementală vizează procesarea individuală a fiecărui micro-batch. Apoi, rezultatele mai multor micro-batch-uri sunt agregate. În consecință, operatorul de agregare incrementală este mai eficient decât operatorul de ferestre, deoarece prelucrarea datelor nu se repetă.

Spark suportă ferestre fixe și de tip sliding, în timp ce colectează date în micro-bach-uri și procesează datele în funcție de ora sosirii. Spark utilizează ferestrele de timp

pentru a colecta cât mai multe date posibil pentru a fi procesate într-un micro-batch. Spark Streaming permite limitarea numărului de înregistrări colectate pe secundă. În consecință, latența lob-urilor va fi imbunătățită. În plus, Spark Streaming poate controla rata de consum bazată pe întârzierile batch-urilor și timpii de procesare, astfel încât job-ul să poată fi procesat cât mai repede.

#### *4.2.2.3. Parallel Recovery*

D-stream-urile utlizează o abordare de recuperare a datelor numită recuperare paralelă. Această recuperare verifică periodic starea anumitor RDD-uri. Apoi replică în mod asincron punctele de control către noduri diferite. Când un nod eșuează, mecanismul de recuperare detectează partițiile pierdute și lansează sarcini pentru a le recupera de la ultimul punct de control. Proiectarea RDD-urilor simplifică mecanismul de control, deoarece RDD-urile sunt disponibile doar pentru citire, lucru care permite Spark-ului să capteze snapshot-uri de RDD-uri, în mod asincron.

Spark utilizează structura RDD pentru a optimiza captarea de checkpoint-uri. Stochează checkpoint-urile într-un spațiu de stocare stabil. Punctele de control sunt benefice pentru RDD cu dependințe largi, deoarece recuperarea de la un eșec poate necesita o re-execuție completă a job-ului. În schimb, RDD cu dependințe înguste nu pot necesita checkpoint-uri. Atunci când un nod eșuează, partițiile pierdute pot fi re-calculate în paralel pe cealalte noduri.

### *4.2.3. Spark Streaming*

Apache Spark Streaming este o extensie Apache Spark API și este capabil să proceseze stream-uri de date într-un timp real. Core-ul Apache Spark este un cluster bazat pe un framework computațional. Apache Spark este capabil de a lua cantități uriașe de date de intrare, încărcându-le în mai multe noduri cluster și procesându-le în paralel. Datele sunt separate în seturi mici și imutabile și asignate la diverse noduri din sistem. Fiecare nod cluster procesează doar datele ce au fost asignate către acesta, astfel asigurându-se o procesare rapidă folosindu-se conceptul de localizare a datelor. Aplicațiile Spark sunt executate în paralel (mai multe instanțe ale aceluiași proces funcționează pe diferite seturi de date pe fiecare nod din cluster). Pentru a aloca resurse și pentru procesarea nodurilor se folosesc manageri la nivelul cluster-ului. Odata ce nodurile sunt alocate, motorul Spark accesează executorii din noduri, care sunt procese ce primesc, stochează și procesează datele dintr-un anumit nod.

Avantajul folosirii unui astfel de framework este acela că fiecare nod are propriul proces de execuție care rulează cât timp aplicația Spark nu e oprită, rulând task-uri multiple (folosind thread-uri multiple). Totodata, Spark nu impune utilizarea managerului său autonom de cluster și sprijină utilizarea managerilor de clustere, cum ar fi Mesos sau YARN, capabili să se conecteze la mai multe aplicații.

Programele Apache Spark pot fi scrise în diverse limbaje de programare, datorită faptului că se ofera API pentru:

- Python
- Java

- Scala

  În plus, ca extensii la aplicația de bază Apache Spark, sunt disponibile, de asemenea:

- execuția interogărilor SQL;
- procesarea operațiilor Hadoop Map și Reduce;
- machine learning;
- prelucrarea datelor grafice și procesarea fluxului de date.


Figura 4.6 Arhitectura Spark Streaming[19]

Apache Spark Streaming permite integrarea diferitelor sisteme de mesagerie și introducere de date, cum ar fi Amazon Kinesis, Apache Flume, Apache Kafka, sisteme de social media cum ar fi Twitter, sisteme de baze de date distribuite precum Hadoop Distributed File System pentru a accepta datele de intrare pentru procesare. Datele o dată introduse în sistem sunt prelucrate utilizând funcții de nivel înalt, cum ar fi: map, reduce, window si join. În cele din urmă, rezultatul generat poate fi trimis la mai multe data sinks, conform cerințelor utilizatorului. De exemplu, datele de ieșire generate pot fi stocate în sisteme de fișiere distribuite, cum ar fi HDFS sau pot fi stocate în sistemul de baze de date normale.

Stream-urile de date sunt acceptate în sistemul Apache Spark Streaming de la o sursă externă de date. Aceste stream-uri continue de date sunt în continuare defalcate sau împărțite în mai multe secțiuni sau batch-uri, care sunt apoi trimise la motorul Apache Spark. Motorul de bază Apache Spark tratează aceste batch-uri de date ca seturi de date imuabile care trebuie procesate conform logicii aplicației. Datele de ieșire generate după prelucrare sunt de asemenea livrate în batch-uri.

Sream-urile de date continue de intrare sunt reprezentate ca fluxuri discreționate (Discretised Streams) sau DStreams în Apache Spark Streaming. Aceasta este o abstracție la nivel înalt furnizată pentru a reprezenta stream-urile continue primite ca intrări sau stream-urile de date generate de procesarea stream-urilor de intrare. Sistemul DStream este defalcat intr-o serie continuă de seturi de date imutabile, distribuite și partiționate, cunoscute ca Seturi de date distribuite reziliente (Resilient Distributed Data sets) sau

[19] http://www.gkrishnan.com/bigdata/2015/06/09/apache-spark-part4-spark-modules

RDD. Fiecare RDD apartine unui anumit Dstream si conține date dintr-un anumit interval de timp. Acest lucru este reprezentat de Figura 4.7 următoare:


Figura 4.7 Reprezentarea modului in care DStream-urile sunt partitionate in Resilient Distributed Datasets

Caracteristici Apache Spark Streaming:

- ***Fault tolerance-*** datorita Resilient Distributed Datasets care sunt seturi de elemente de date imutabile care pot fi re-calculate si determinate in cazul unui esec.
- ***Throughput*** - pentru a asigura o performanță ridicată, Spark oferă facilitatea de a primi mai multe stream-uri în paralel. Acest lucru se poate realiza dacă sunt create mai multe receptoare. Aceste receptoare acceptă simultan date de la DStreams de intrare pentru a menține o rată ridicată de ingestie a datelor.

***Scalabilitate*** - fiind un framework bazat pe cluster, Apache Spark Streaming, sau mai degrabă Apache Spark acceptă executarea job-urilor în mai multe noduri conectate unul la celălalt. Aceste noduri de cluster funcționează în paralel pentru a echilibra sarcina de procesare. Apache Spark este capabil să crească sau să scadă dinamic numărul de noduri care lucrează în paralel utilizând metoda Dynamic Allocation. În funcție de statusul batch job -urilor, Apache Spark include sau exclude nodurile procesoarelor, făcându-l astfel foarte scalabil.

### *4.2.4. Spark Machine Learning Library*

Machine Learning Library (MlLib) este biblioteca Spark a funcțiilor de machine learning. Conceput pentru a funcționa în paralel pe clustere, MLlib conține o varietate de algoritmi de machine learning și este accesibil din toate limbajele de programare ale Spark. Designul și arhitectura MlLib sunt simple: permite invocații ale diferiților algoritmi pe seturi de date distribuite, reprezentând aceste date ca RDD-uri. Acest modul oferă funcționalități care includ: statistici, clasificare, regresie, filtrare colaborativă, clusterizare, extracție de caracteristici, optimizare, etc.

### *4.2.5. Spark SQL*

Spark SQL este o componentă deaspura Spark Core-ului care oferă suport pentru date structurate și nestructurate. Acesta oferă o interfață care interacționează cu Spark prin intermediul limbajelor de interogare precum SQL și HiveQL. Limbile de interogare sunt traduse în operațiile Spark și tabelele bazei de date sunt reprezentate ca RDD-uri.

Din Spark versiunea 1.3 data frame-urile au fost introduse în Apache Spark, astfel încât datele Spark să poată fi procesate într-o formă tabulară și funcțiile tabulare (cum ar fi selectați, filtrați, groupBy) pot fi folosite pentru procesarea datelor. Modulul Spark SQL se integrează cu formatele Parquet și JSON pentru a permite stocarea datelor în formate care reprezintă mai bine datele. Acest lucru oferă, de asemenea, mai multe opțiuni de integrare cu sistemele externe.

## 4.3. Twitter APIs

Twitter[20] este o rețea socială creată și fondată de Jack Dorsey, Biz Stone, Evan Williams și Noah Glass în 2006. Utilizatorii înregistrați pot posta și vizualiza mesaje cu un maxim de 140 de caractere pe lungime numite "tweets". Utilizatorii activi pe Twitter au crescut exponențial (vezi Figura 4.8) și sunt în prezent peste 310 milioane.

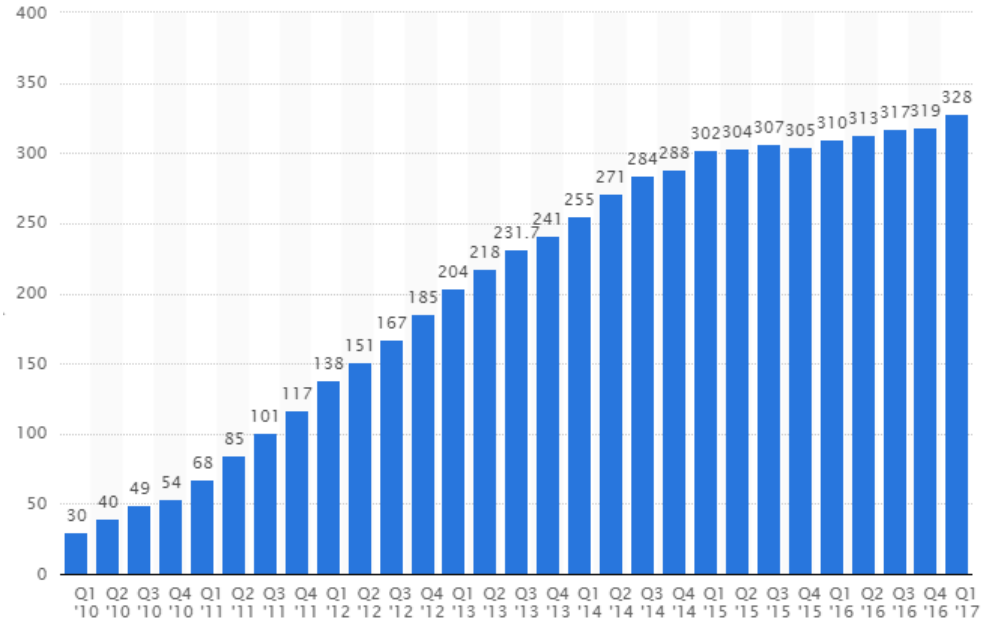



Figura 4.8 Numărul de useri activi, într-o lună, exprimat în milioane din 2010 - 2016 [21]

Utilizatorii de pe Twitter împărtășesc opinii, sentimente și opinii despre tot felul de mărci, subiecte, locuri și persoane. Ei dau follow unii altora și diferitelor persoane publice pentru a fi la curent cu tendințele și produsele actuale. Mesajele tweets sunt vizibile publicului în mod prestabilit, cu toate acestea, utilizatorii își pot restricționa mesajele tweets doar la cei care îi urmăresc. Multe aplicații comerciale și instituții de

[20] https://twitter.com/
[21] https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

cercetare au utilizat informațiile disponibile publicului în tweet-urile de 140 de caractere pentru dezvoltarea de produse și cercetarea academică.

Pentru a permite accesul programatic la tweets pentru analiză, Twitter furnizează două tipuri diferite de API - API-urile REST [22] și API-urile Streaming[23]. API-urile REST permit postarea de tweet-uri, citirea profilului utilizatorilor și a datelor acestora și efectuarea de căutări singulare ale tweet-urilor istorice în ultimele 7 zile, bazate pe locație, cuvinte cheie etc. Cu toate acestea, pentru a accesa tweets în timp real, trebuie să fie utilizat API-ul Twitter Streaming API.

**Twitter Streaming API**

API-ul Streaming oferă acces cu latență scăzută la date din Twitter în timp real. S pre deosebire de API-urile REST, API-ul Streaming necesită menținerea unei conexiuni HTTP persistente. Figura 4.8 și Figura 4.9 arată modul în care API-ul REST și respectiv API-ul Streaming vor gestiona în mod diferit solicitarea HTTP a unui utilizator pentru aceeași aplicație.

Astfel, în cazul API-urilor REST, requestul de tip HTTP a utilizatorului va stabili o conexiune la API-ul Twitter. Cu toate acestea, în cazul fluxului API, procesul HTTP și procesul de streaming vor fi difuzate separat. Procesul de streaming colectează tweet-uri în timp real și apoi filtrează, analizează și le stochează într-un data store. Procesul HTTP va interoga datele atunci când un utilizator generează o solicitare HTTP. Ambele API-uri utilizează OAuth[24] pentru a permite accesul autorizat la utilizatori și aplicații.

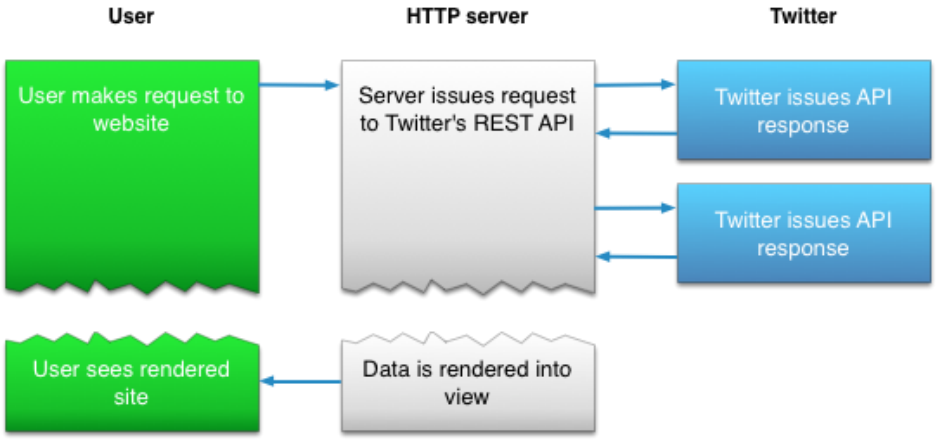



Figura 4.9 Exemplu de Twitter REST API [25]

[22] https://dev.twitter.com/rest/public.
[23] https://dev.twitter.com/streaming/overview.
[24] https://dev.twitter.com/oauth
[25] https://dev.twitter.com/streaming/overview

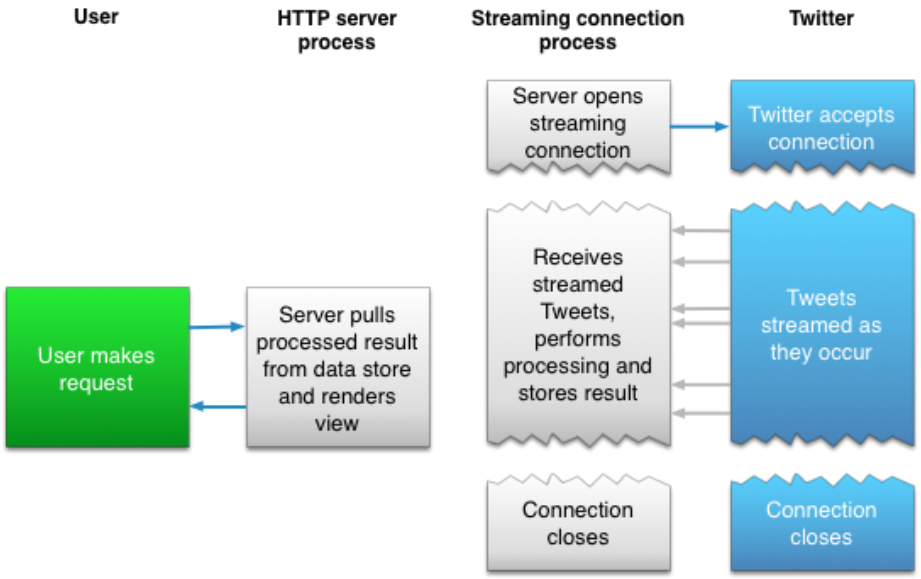



Figura 4.10 Exemplu de Twitter Streaming API [26]

După ce se stabilește o conexiune cu API, serverele Twitter vor menține o conexiune deschisă, atât timp cât nu există erori la nivelul serverului, erori de rețea, mai multe intrări cu aceleași acreditări sau scădere bruscă a ratei de streaming.

API-ul Streaming furnizează în prezent 11 parametri pentru request [27], care pot fi utilizați pentru a specifica ce date vor returna endpoint-urile API-ului. Unii dintre acești parametri utilizați în mod obișnuit la toate endpoint-urile de streaming sunt descrise mai jos:

**delimited**: acest parametru este setat la șirul "lungime" pentru a indica faptul că tweeturile vor fi delimitate în flux, astfel încât clientul să știe câți octeți să citească înainte de sfârșitul tweet-ului

**stall_warnings:** dacă acest parametru este setat la "true", atunci clientul va primi periodic mesaje de avertizare dacă riscă să fie deconectat din cauza ratei reduse de streaming.

**filter_level**: acest parametru este setat la "none" în mod implicit, ceea ce înseamnă că toate mesajele tweet disponibile vor fi afișate. Nivelul de filtrare poate fi, de asemenea, setat la "low" sau "medium". Acesta din urmă va livra tweet-urile care apar ca rezultate de top pentru căutări pe site-ul Twitter.

**language**: acest parametru va indica limba în care trebuie să fie inserate mesajele tweets.

[26] https://dev.twitter.com/streaming/overview
[27] https://dev.twitter.com/streaming/overview/request-parameters

**follow**: acest parametru va conține o listă de ID-uri separate prin virgule care indică ce tweet-uri de utilizator trebuie trimise în flux. Fluxul livrat poate consta din tweet-uri create de utilizator, re-tweet-uri de către utilizator, răspunsuri la tweet-uri create de utilizator și re-tweet-uri de orice tweet create de utilizator.

**track**: acest parametru va conține o listă de fraze separate prin virgule care vor fi folosite ca, cuvinte cheie pentru filtrarea tweet-urilor care urmează să fie livrate pe flux. Textul de tweet, menționări ale numelor de utilizatori, textul în hashtags și url-urile afișate vor fi verificate pentru potriviri la lista de cuvinte cheie.

**location**: Acest parametru conține o listă de perechi de linii longitudinale și latitudine, separate prin virgulă, cu un set de casete delimitate pentru filtrarea tweeturilor în funcție de locația geografică. Dacă tweets se încadrează în caseta de legare, acestea vor fi livrate în flux.

# Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol va fi descrisă soluția aleasă pentru implementarea unui sistem de analiză Big Data, pe baza studiuliu bibliografic de la capitolul 3 și al analizei și fundamentării teoretice realizate și prezentate la capitolul 4. Sistemul permite procesarea datelor în timp real și realizează o analiză continuă pentru identificarea sentimentelor din conținutul mesajelor din rețeaua socială Twitter.

Soluția reprezintă o aplicație implementată folosind framework-ul Apache Spark, dezvoltată în limbajul de programare Java 8. Aplicația va realiza o analiză în timp real a tuturor tweet-urilor trimise pe rețeaua social numită Twitter, având ca scop analiza acestor tweet-uri pentru a determina sentimentele conținutului lor.

Capitolul 5 reprezintă metoda de clasificare și analiză a acestor mesaje care circula în mediul online și vor fi prezentate tehnologiile folosite, formatul datelor de intrare, modele de clasificare, arhitectura sistemului și implementarea folosită.

În continuare se va face referire la sistemul implementat în Apache Spark, **TweetSent**.

## 5.1. Framework-uri si tool-uri

Pentru implementarea și rularea întregului sistem au fost folosite o serie de tool-uri și framework-uri. Ca mediu de dezvoltare a fost utlizat Eclipse IDE, versiunea Neon, iar pentru managementul integrării și al compilării Apache Maven 3.5. Partea de pocesare a datelor este scrisă în întregime în Java, comunicând cu o serie de librării externe pentru realizarea analizei tweet-urilor în timp real. Partea de vizualizare, este dezvoltată în Python 3.6.1, JavaScript, HTML, CSS. Pentru rularea aplicației, a fost folosit framework-ul Apache Spark.

- **JDK 1.8.0_121**[28]: include instrumente utile pentru dezvoltarea și testarea programelor scrise în limbajul de programare Java și care rulează pe platforma Java.

- **Eclipse Neon**[29]: mediu de dezvoltare (IDE) pentru dezvoltarea aplicațiilor în Java. Oferă suport pentru integrarea proiectelor folosind Maven.

- **Apache Maven 3.5.0**[30]: tool pentru managementul proiectelor software, bazat pe conceptul de POM (project object model). Maven permite o integrare simplă a diferitelor componente externe într-un proiect și realizează compilarea și integrarea aplicației în mod automat.

[28] http://www.oracle.com/technetwork/java/javase
[29] http://www.eclipse.org/neon/
[30] https://maven.apache.org/

- **Twitter4J**[31]: o librărie open-source pentru accesarea API-ului Twitter din Java. Twitter4J permite integrarea serviciilor Twitter în aplicații Java și reprezintă o metodă ușoară pentru accesarea funcționalității acestuia. Twitter4J implementează o serie de funcționalități precum autentificarea utilizatorului, accesul la stream-uri de tweet-uri și diferite metode de acces la câmpurile din care acesta este compus

- **Twitter Streaming API**[32]: permite programatorului acces cu latență mică la stream-ul global de tweet-uri din Twitter. O implementare corespunzătoare a acestui API va trimite un mesaj indicând un nou tweet sau un alt tip de eveniment din rețeaua socială Twitter, evitând overhead-ul asociat accesării unui serviciu de tip REST.

- **Apache Spark 1.6.0**[33]: este un framework puternic, de procesare, open source, construit în jurul vitezei, ușurinței de utilizare și analizelor sophisticate, cu scopul procesării paralele a datelor la scară largă.

- **Stanford CoreNLP 3.7.0**[34]: oferă un set de instrumente de tehnologie a limbajului uman. Tool-ul poate da forme de bază ale cuvintelor, părțile lor de vorbire, indiferent dacă sunt nume de companii, oameni etc., normalizează datele, orele și cantitățile numerice, marchează structura propozițiilor și dependențe sintactice, indică care expresii sustantivale se referă la aceleași entități, indică sentimentul, extrage relații specifice sau deschise între mențiunile entității, obține citatele pe care oamenii le-au spus etc.

- **Redis 3.2.9**[35]: este un model NoSQL, reprezentând o paradigmă diferită de stocare a datelor. Dezvoltatorii îl numesc server de structuri de date, motiv întemeiat, deoarece filozofia Redis pornește de la ideea că datele ar trebui stocate în memoria RAM. În felul acesta Redis poate efectua în principiu operațiile cu o viteză ridicată, eliminând procesul laborios întâșnit de exemplu la SQL, unde datele sunt scrise pe disc.

- **Jedis 2.9.0**:[36] este o bibliotecă client în Java pentru Redis

- **Python 3.6.1**[37]: este un limbaj de programare dynamic multi-paradigmă, care pune accentul pe curățenia și simplitatea codului.

[31] http://twitter4j.org/en/
[32] https://dev.twitter.com/streaming/overview
[33] https://spark.apache.org/docs/1.6.0/
[34] https://stanfordnlp.github.io/CoreNLP/download.html
[35] https://redis.io/
[36] https://redis.io/clients#java
[37] https://www.python.org

- **Flask 0.12.2**[38]: este un micro-framework web scris în Python. Se numește micro-framework deoarece nu necesită tool-uri sau librării speciale. Nu are un strat de abstractizare a bazei de date, validarea formularului sau orice alte componente în care bibliotecile terțe existente oferă funcții comune.

- **HTML5**[39]: limbaj pentru structurarea și prezentarea conținutului World Wide Web[40]

- **Datamaps**[41]: permite vizualizarea unei hărți personalizate pentru aplicații web, într-un singur fișier JavaScript.

### *5.1.1. Formatul datelor de intrare*

Datele de intrare folosite pentru a determina sentimentele din surse de date sociale reprezintă tweet-uri reale, citite accesând API-ul Twitter Stream API.

Twitter Stream API este oferit de Twitter pentru a permite accesul dezvoltatorilor, la fluxul global de date Tweet. Se pot obține astfel toate tweet-urile publice globale, în format JSON, incluzând toate informațiile despre acest tweet. Odată stabilită conexiunea la API-ul de streaming, datele vor curge constant în sistem. Acest API, acceptă diverse filtre, de exemplu cuvinte cheie sau limbi. Este posibil ca datele returnate să cu conțină informații geografice, iar în sistemul prezentat se vor utiliza doar tweet-uri cu informații geografice. Cu filtrul de limbă, sunt analizate doar tweet-urile în engleză.

Pentru a stabili întreaga conexiune, primul pas care trebuie făcut este obținerea datelor de autentificare, specifice unui dezvoltator de aplicații Twitter, prin crearea unui cont care pune la dispoziție următoarele date: cheia cosumatorului (*Consumer Key*), cheia secretă a consumatorului (*Consumer Key*), token-ul de acces (*Access Token*) și parola acestuia, token-ul de acces secret (*Secret Access Token*).

După ce se stabilește o conexiune cu API, serverele Twitter vor menține o conexiune deschisă, atât timp cât nu există erori la nivelul serverului, erori de rețea, mai multe intrări cu aceleași acreditări sau scădere bruscă a ratei de streaming.

După autentificare, se pot obține tweet-uri reale, în timp real, numite statusuri. Stream-urile publice sunt împărțite în trei categorii:

- GET statuses / sample - returnează un număr de statusuri publice alese aleator
- POST statuses / filter - toate tweet-urile care corespund cel puțin unui filtru specificat ca și parametu. Este posibilă specificarea unui număr multiplu de filtre, însă cel puțin unul trebuie să fie prezent.
- GET statuses / firehose - permite accesul la totalitatea tweet-urile publice din rețeaua socială Twitter, însă necesită drepturi de acces suplimentare.

[38] http://flask.pocoo.org/
[39] https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
[40] https://ro.wikipedia.org/wiki/World_Wide_Web
[41] http://datamaps.github.io/

Pentru accesul aplicației TweetSent la stream-ul de statusuri este folosită librăria Spark Streaming Twitter, care pune la dispoziție pachetul Twitter.utils, care conține toate funcțiile încorporate pentru a transmite date din Twitter, folosind metoda createStream, din clasa TwitterUtils. Conform definiției metodei, aceasta creează un flux de date care returnează tweet-urile primite de pe Twitter, utilizând autentificarea implicită de Twitter4J.

Librăria Twitter4J permite atât autentificarea la API-ul de Twitter, cât și un acces simplificat la această resursă, prin metoda sample(). Această metodă returnează tweet-uri sub forma unui obiect de tip Status. Acest obiect este construit pe baza json-ului returnat de API-ul Twitter Stream API și încapsulează câmpurile și valorile din interiorul structurii de date de tip json. Forma unui status este prezentată în Figura 5.1


```
{"accessLevel":0,
"contributors":[],
"createdAt":1498293473000,
"currentUserRetweetId":-1,
"extendedMediaEntities":[],
"favoriteCount":0,
"favorited":false,
"hashtagEntities":[],
"id":878532620203765760,
"inReplyToStatusId":-1,
"inReplyToUserId":-1,
"lang":"en",
"mediaEntities":[],
"place":
    {   "accessLevel":0,
        "boundingBoxCoordinates":
        [[  {"latitude":51.477225,"longitude":-0.228589},
            {"latitude":51.530348,"longitude":-0.228589},
            {"latitude":51.530348,"longitude":-0.149791},
            {"latitude":51.477225,"longitude":-0.149791} ]],
        "boundingBoxType":"Polygon",
        "country":"United Kingdom",
        "countryCode":"GB",
        "fullName":"Kensington, London",
        "id":"01c7ed8caf11e8b2",
        "name":"Kensington",
        "placeType":"city",
        "url":"https://api.twitter.com/1.1/geo/id/01c7ed8caf11e8b2.json"
    },
"possiblySensitive":false,
"quotedStatusId":-1,
"retweet":false,
"retweetCount":0,
"retweeted":false,
"retweetedByMe":false,
"source":"<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for iPhone</a>",
"symbolEntities":[],
"text":"Morning puppies, time for breakfast soon!! Remember if you don't have a
        hood we have several hoods for you to borro… https://t.co/f3ozUrl53L",
"truncated":true,
"urlentities":
    [   {   "displayURL":"twitter.com/i/web/status/8…",
            "end":140,
            "expandedURL":"https://twitter.com/i/web/status/878532620203765760",
            "start":117,
            "text":"https://t.co/f3ozUrl53L",
```


Figura 5.1 Structura parțială a unui status emis de Twitter4J

Câmpurile relevante pentru TweetSent sunt următoarele:

- id: conține id-ul unui tweet postat pe rețeaua de socializare
- screenName: numele utilizatorului care folosește rețeaua

- text: conține textului unui status (tweet) postat de către un utilizator al rețelei sociale.

- geoLocation: reprezintă coordonatele de longitudine și latitudine ale utilizatorului care a emis tweet-ul. Coordonatele sunt reprezentate sub forma unui obiect de tip GeoLocation, care conține cele două câmpuri.

- originalProfileImageURL: imaginea de profil a utilizatorului

- createdAt: momentul în care a fost postat tweet-ul

## 5.2. Metode folosite pentru determinarea sentimentelor

În prezent, aproximativ 90% din toate datele generate în lumea noastră au fost generate doar în ultimii doi ani.. Unele din principalele surse de date din zilele noastre sunt rețelele sociale. Analiza acestor date are un mare potențial în găsirea de informații relevante în cadrul rețelelor sociale Twitter este un serviciu dezvoltat pentru a permite comunicarea între oameni prin trimiterea de mesaje scurte.

Conform cercetării [19], Twitter, ca a doua cea mai mare platformă media socială, chiar în spatele Facebookului, generează în jur de 350 000 de tweets în fiecare minut, sau 21 de milioane pe oră. Astfel de volume de date prezintă provocări pentru ingineri pentru a dezvolta o soluție inovatoare pentru arhitecturi eficiente și necesită capabilități de prelucrare a datelor pentru a aplica extracția de date relevante.

În contextul TweetSent, cele doua metode de clasificare folosite pentru a determina sentimentul sunt machine learning și deep learning, folosite prin intermediul librariilor MlLib Spark și Stanford Core NLP. Tabelul 5.1 este un exemplu de clasificare a sentimentului unui tweet, dobândit după execuția mai multor pași, descriși în subcapitolele 5.2.1 și 5.2.2, în care sunt prezentate etapele de pre-procesare, respectiv de clasificare a unui tweet. Tweet-urile pot fi clasificate în trei categorii: pozitive, negative și neutre.

| Exemple de clasificare a unui tweet, obținut de TweetSent | |
|---|---|
| Pozitiv | For now, just love yourself and give yourself the love you think you deserve |
| Neutru | I'm at Chomp Chomp Food Centre in Singapore |
| Negativ | Accident on #NY430 Both directions from Town of Mina; Town of Sherman Line to Town of Chautauqua |

Tabel 5.1 Categoriile de clasificare a unui tweet

### *5.2.1. Pre-procesarea tweet-urilor*

Mesajele Twitter, spre deosebire de majoritatea celorlalte surse de date, sunt extrem de scurte, cu o limită maximă de 140 de caractere și conțin limbi diferite, limbaj informal și greșeli. Astfel, unele tehnici de analiză a sentimentului și selecții ale funcțiilor în cazul altor surse ar putea să fie dificil de aplicat la surse de date Twitter.

Scopul procesării tweet-urilor înainte de analiza efectivă, este de a le curăța de atributele redundante și de a obține seturi de date normalizate pentru instruire și testare.

În urma stream-ului de date primit, fiecare text al tweet-ului din obiectele de tip status, este pre-procesat prin aplicarea de expresii regulate – regex, eliminarea stop-words-urilor și tokenizare.

O expresie regulată este un set de caractere care specifică un model. În TweetSent aceste expresii sunt folosite pentru a specifica șiruri de caractere care se doresc a fi extrase din text, deoarece nu aduc valoare în analiza de sentimente. Mesajele de pe Twitter conțin multe elemente nedorite în clasificarea lor, cum ar fi URL-uri, semnul de RT(re-tweet), hashtags sau diferite semne de punctuatie. Prin implementarea expresiilor regulate, aceste informații pot fi eliminate. Mențiunile Twitter sunt înlocuite cu spații albe și fiecare tweet este transformat în format text cu litere mici. Funcția de curățare a datelor, este exemplificată mai jos, în Figura 5.2.

```
String[] newTweetText = tweetText.toLowerCase()
         .replaceAll("\n", "")
          .replaceAll("rt\\s+", "")
          .replaceAll("\\s+@\\w+", "")
          .replaceAll("@\\w+", "")
          .replaceAll("\\s+#\\w+", "")
          .replaceAll("#\\w+", "")
        .replaceAll("(?:https?|http?)://[\\w/%.-]+", "")
        .replaceAll("(?:https?|http?)://[\\w/%.-]+\\s+", "")
          .replaceAll("(?:https?|http?)//[\\w/%.-]+\\s+", "")
          .replaceAll("(?:https?|http?)//[\\w/%.-]+", "")
        .split("\\W+");
```

Figura 5.2 Funcție pentru aplicarea expresilor regulate unui tweet

Clasificarea sentimentelor pe Twitter este de obicei afectată de natura zgomotoasă (abrevieri, forme neregulate) de date tweets. O procedură populară de reducere a zgomotului datelor textuale este eliminarea stop words-urilor prin utilizarea listelor de expresii pre-compilate sau a unor metode mai sofisticate pentru identificarea dinamică a expresiilor. Această metodă se bazează pe ideea că eliminarea stop words-urilor reduce spațiul caracteristic al clasificatorilor și îi ajută să producă rezultate mai precise.

Stop words-urile sunt cuvinte care nu au nicio valoare pentru analiza sentimentului, deoarece ele nu modifică contextul în niciun fel. Prin urmare, acestea trebuie eliminate, pentru a nu schimba polaritatea sentimentului. Natural Language Toolkit furnizează biblioteci și fișiere cu stop words. Un astfel de fișier este utilizat și în sistemul actual. Acestea sunt clasificate în funcție de limbă și, astfel, soluția TweetSent

va atașa numai stop words în limba engleză. Exemple de cuvinte stop words: “I”, “me”, “you”, “us”, “yourselves”, “which”, “with”, “from”, “to”, etc.

După eliminarea stop words-urilor, urmează procesul de tokenizare, un proces de divizare a șirurilor de text în token-uri, care sunt reprezentate de cuvintele din text. Twet-urile sunt vectorizate și gata de analiză.

### *5.2.2. Analiza sentimentelor*

Clasificatorul de sentimente din sistemul TweetSent are trei clase, “pozitiv”, “negativ” și “neutru”. Chiar dacă acest tip de clasificare a sentimentelor pe data tweet are un număr de clase mai mic comparativ cu alte probleme de clasificare a textului, aceasta are provocări mult mai mari. Unele cuvinte pot avea o anumită orientare într-un context, și cu totul alta într-un context diferit. De exemplu, în timp ce cuvântul “trist” este un cuvânt negativ, acesta ar trebui să indice sentimentul pozitiv dacă se folosește în propoziția "Titanicul mă face trist de fiecare dată când îl urmăresc", deoarece filmul a avut succes în transmiterea emoției intenționate. Astfel, identificarea contextului textului ar fi adesea crucială pentru a fi etichetată cu precizie pozitivă sau negativă. Acest lucru face ca analiza sentimentului să fie destul de dificilă, deoarece nu există o modalitate definitivă de înțelegere a contextului unui text.

Unele fraze, cum ar fi cele interogative sau afirmative pot conține cuvinte sentimentale, dar de fapt sa nu exprime niciun sentiment. De exemplu, propozițiile „Este acest film bun?” Și „Dacă filmul obține un rating bun, îl voi viziona” nu transmit nicio informație subiectivă despre niciun film, în ciuda faptului că conțin cuvântul pozitiv „bun”. Cu toate acestea, nu este necesar ca toate declarațiile interrogative sau condiționale să fie lipsite de informații despre sentimente. De exemplu, afirmația condiționată „Dacă doriți să urmăriți un thriller bun, vizionați The Prestige” exprimă sentimente pozitive pentru filmul “The Prestige”. Astfel, este dificil de identificat o propoziție ca fiind neutră atunci când conține fie cuvinte pozitive, fie negative.

Una dintre cele mai dificile provocări de cercetare în NLP este identificarea declarațiilor sarcastice. O altă provocare în analiza sentimentului este prezența cuvintelor de negare, cum ar fi, „nu”. S-ar putea considera ca negațiile să inverseze întotdeauna orientarea sentimentului. De exemplu, "Nu-mi place acest film" inversează orientarea cuvântului, de la pozitiv la negative.

Analiza sentimentului utilizând date sociale, cum ar fi Twitter tweets, de asemenea, prezintă o provocare care nu este de obicei găsită în timpul analizei sentimentului de text formal. Social media conține adesea jargon de internet și cuvinte scrise rapid și prescurtat, care se schimbă rapid în timp și argumente culturale nu numai în limbi diferite, ci și în limba engleză. Textul social media este, de asemenea, informal, prin urmare, există multe cuvinte care pot implica o orientare diferită în funcție de grupa de vârstă a autorului. Propoziții precum "Filmul a fost rău" sau "Russel Peters are un simț al umorului ridicol" transmite emoții pozitive pentru film și Russel Peters, chiar dacă cuvintele răi și ridicole sunt cuvinte negative.

În continuare, vor fi descrise etapele realizate pentru analiza sentimentelor de date Twitter, din aplicația TweetSent. Au fost folosite doua metode de analiză, Naive Bayes –

algoritm de machine learning, pus la dispoziție de Spark MlLib, și Stanford Core NLP, o extensie care oferă o analiză fundamentală a limbajului natural care folosește un model computațional asupra arborilor, folosind deep learning.

### *5.2.2.1. Modelul Naive Bayes*

Pentru a instrui și a testa clasificatorul și a prezice etichetele din noile recenzii pe baza atributului lor de text, setul de date trebuie împărțit în seturi de testare și instruire.

Machine learning supravegheată este una din tehnicile utilizate în cadrul acestei secțiuni, deoarece algoritmul încearcă să prezică date neetichetate, după ce rulează pe tweet-uri pozitive, negative și neutre. Pentru a obține o precizie mai mare a clasificatorului, antrenamentul și testarea sunt direcționate spre setul de date aleatoriu amestecat.

Naïve Bayes este algoritmul de machine learning care poate funcționa eficient pentru clasificarea datelor. Este un clasificator probabilistic care atribuie etichete valorilor caracteristice, în timp ce această etichetă este preluată de la un set de date cunoscut. Clasificatorii Bayes au ipoteze despre caracteristici, în care o caracteristică este independentă de cealaltă, dacă aparțin unei anumite clase, adică pozitive, negative sau neutre. Mai mult decât atât, Naïve Bayes alege eticheta pentru datele de intrare pe baza probabilității etichetelor așa cum apar în setul de antrenament. Probabilitatea estimării este determinată pentru fiecare etichetă cu referință de estimare a datelor de intrare.

În plus, caracteristicile pot genera diferite tipuri de modele pe care le specifică. Prin urmare, diferite modele pot fi analizate și executate ca clasificatoare pentru analizele de sentiment. Clasificatorul Bernoulli ia în considerare caracteristicile boolean-e, astfel încât nu face diferența în precizie dacă aceleași cuvinte au apărut de mai multe ori în cadrul atributului de text tweet. Clasificatorul Multinomial clasifică precizia textului și a ieșirilor, în timp ce ia în considerare numărul de cuvinte ca, caracteristică.

**Instruirea clasificatorului:**

Primul pas în aplicarea acestei tehnici de analiză, este instruirea unui clasificator, care apoi să poată prezice tipul de sentiment al fiecărui tweet, în timp real.

Găsirea unui corpus potrivit pentru instruirea modelului este foarte importantă, deoarece, indiferent de cât de bun este algoritmul, implementarea sau reglarea parametrilor, rezultatul final nu va fi unul bun dacă nu este utilizat un corpus adecvat tipului de date analizat.

În această secțiune se vor prezenta două seturi de date diferite, utilizate în analiza de sentimente din surse sociale. Selecția celei mai bune a fost concentrată pe setul de date care este clasificat manual, oferind un set de adnotări de încredere asupra tweet-urilor. Cele două seturi de date sunt: Stanford Twitter Sentiment corpus[42] și Sentiment Strength Twitter Dataset[43].

Setul de date Stanford Twitter Sentiment:

[42] http://help.sentiment140.com/

[43] http://sentistrength.wlv.ac.uk/documentation/

Stanford Twitter Sentiment corpus constă în două seturi diferite, unul pentru instruire și unul pentru testare. Setul de antrenament conține 1,6 milioane de tweet-uri etichetate automat drept pozitive sau negative pe baza emoticoanelor. De exemplu, un tweet este etichetat ca pozitiv dacă conține :), :-), :), :D sau =)) și este etichetat ca negativ dacă conține :(, :-(, sau :((. Deși adnotarea automată a sentimentului folosind emoticoane este rapidă, acuratețea este susceptibilă deoarece emoticoanele ar putea să nu reflecte sentimentul real al tweet-urilor, sau să lipsească.

Setul de test pe de altă parte, este adnotat manual și conține 177 mesaje negative, 182 pozitive și 139 de mesaje neutre. Aceste tweet-uri au fost colectate prin căutarea Twitter API cu interogări specifice, inclusiv nume de produse, companii și persoane. Deși setul de date este relativ mic, acesta a fost folosit pe scară largă în domeniul de specialitate în diferite sarcini de evaluare.
Setul de date Sentiment Strength:

Acest set de date cuprinde 10714 de tweet-uri marcate manual, cu puterea lor pozitivă și negativă a sentimentelor. Adică o rezistență negativă este un număr între 1 (Nu negativ) și -5 (extrem de negative). Setul de date a fost construit pentru a evalua SentiStrenth, o metodă bazată pe lexicon pentru detectarea rezistenței sentimentului.

În această lucrare s-a propus re-adnotarea tweet-urilor, cu etichete de sentiment (pozitiv = 4, negativ = 0 și neutru = 2), mai degrabă decât puncte forte ale sentimentului, ceea ce va permite utilizarea acestui set de date pentru clasificarea subiectivității în plus față de detectarea rezistenței sentimentului.

În acest scop, se atribuie o singură etichetă de sentiment fiecărui tweet pe baza următoarei reguli inspirate de modul în care SentiStrength funcționează: un tweet este considerat neutru dacă valoarea absolută a diferenței dintre eticheta pozitivă și cea negativă este 0, pozitiv dacă este mai mare decât 0 și negativ altfel.

Setul de date original este public, împreună cu încă 5 seturi de date de pe diferite platforme sociale, inclusiv MySpace, Digg, forumul BBC, forumul Runners World și YouTube. Pentru a testa acuratețea, instruirea a fost aplicată și pe un fișier cu toate sursele.

**Testarea modelului:**

Prima sursă de date, Stanford Twitter Sentiment conține două fișiere, unul de instruire, unul de test. În aplicația TweetSent, modelul Naïve Bayes a fost antrenat pe o sursă de date de 1,6 milioane de tweet-uri clasificate automat și testat pe 498 de tweet-uri clasificate manual.

A doua sursă de date, Sentiment Strength, conține doar câte un fișier pentru fiecare rețea socială, Twitter, MySpace, Digg, forumul BBC, forumul Runners World și YouTube. Acest set de date a fost prelucrat astfel încât să fie posibilă instruirea modelului pe ~64% din informații și testarea pe ~36% din ele. Având în vedere că datele includ mai multe rețele sociale, s-au facut teste pe un set de date format doar din tweet-uri și pe un set de date format din toate status-urile din toate rețelele.

**Acuratețea modelului:**

Modelul de clasificare cu cea mai mare acuratețe va fi considerat în continuare pentru analiza de sentimente din rețeaua socială Twitter.

Acuratețea a fost calculată folosind următoarea ecuație:

$$acuratețea = \frac{potrivite}{estimate}\ x\ 100\%$$

, unde potrivite = numărul total de etichete de sentiment care au corespuns valorii estimate

și estimate = numărul total de sentimente prezise utilizate pentru testarea modelului

| Set de date | Instruire | Testare | Acuratețe |
|---|---|---|---|
| Stanford Twitter Sentiment | 1,6 milioane tweet-uri | 498 tweet-uri | 57,83 % |
| Sentiment Strength – doar pentru Twitter | 4241 tweet-uri | 1565 tweet-uri | 85,23 % |
| Sentiment Strength – pentru toate rețelele sociale | 10713 tweet-uri | 3942 tweet-uri | 74,15 % |

Tabel 5.2 Acuratețea modelului Naive Bayes pe mai multe seturi de date

După fazele de instruire și testare, s-au obținut procentele de acuratețe conform Tabelului 5.2. Cu toate acestea, deoarece aceste clasificări sunt alese aleator de algoritm, noua iterație a instruirii și testelor poate genera procente diferite față de cele de mai sus.

În concluzie, setul de date Sentiment Strength este cel ales pentru analiza de sentimente a tweet-urile procesare în timp real de către TweetSent.

### *5.2.2.2. Modelul Stanford CoreNLP*

Cea de-a doua metodă de analiză de sentimente este implementata cu Stanford CoreNLP.

Stanford CoreNLP este o bibliotecă de analiză a limbajului natural Java. Stanford CoreNLP integrează toate instrumentele NLP, inclusiv tagger-ul de părți de vorbire (POS), recunoașterea entității (NER), parserul și instrumentele de analiză a sentimentului și oferă fișiere model pentru analiza limbii engleze.

Fiecare tweet este pus intr-un obiect Annotation și apoi o secvență de adnotatori adaugă informații într-un pipeline de analiză. Adnotările folosite în TweetSent se pot vizualiza în Figura 5.3.

În etapa de tokenizare, se împarte tweet-ul în token-uri. Cea de împărțire divizează o succesiune de token-uri în propoziții. Apoi, se aleg părțiile de propoziție și se generează formele de bază a lor. La sfărșit, se analizează sentimentele cu un model computațional asupra arborilor propozițiilor, folosind deep learning.

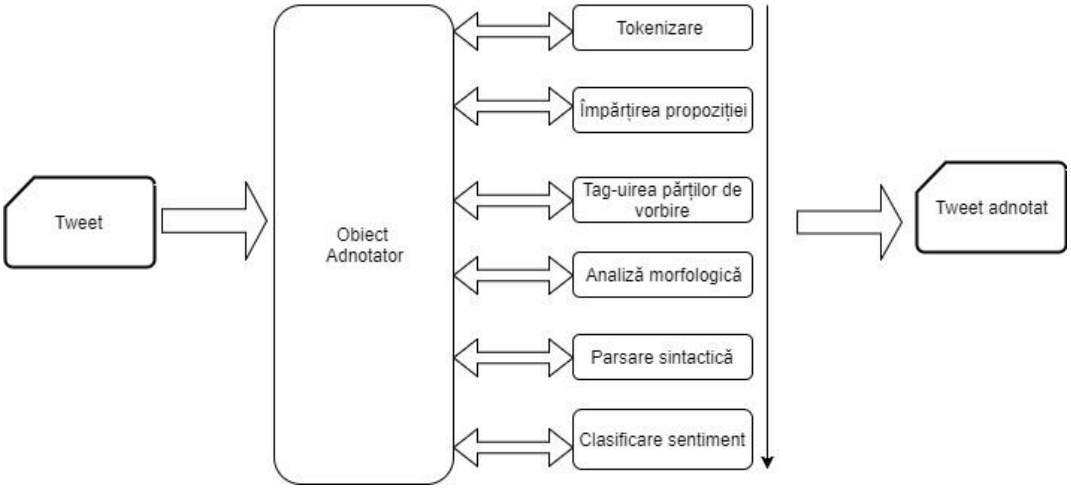



Figura 5.3 Arhitectura generală a sistemului de analiză a sentimentelor cu Stanford CoreNLP

De exemplu, pentru tweet-ul “For now, just love yourself and give yourself the love you think you deserve”, vom avea următorii pași[44]:

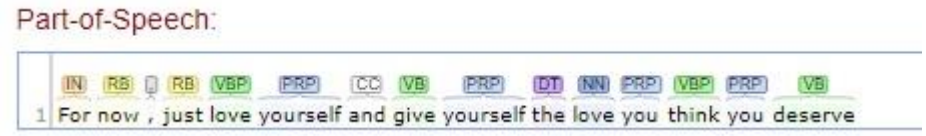
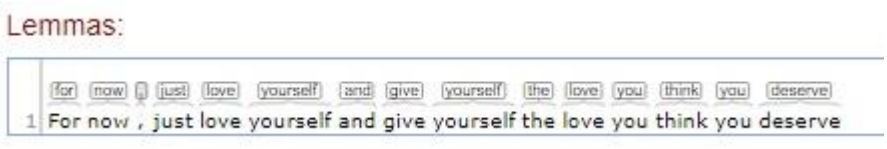



Figura 5.4 Identificarea părților de vorbire – imagine realizată cu ajutorul *corenlp.run*




[44] http://corenlp.run/

Figura 5.5 Generarea cuvintelor de bază vorbire – imagine realizată cu ajutorul *corenlp.run*

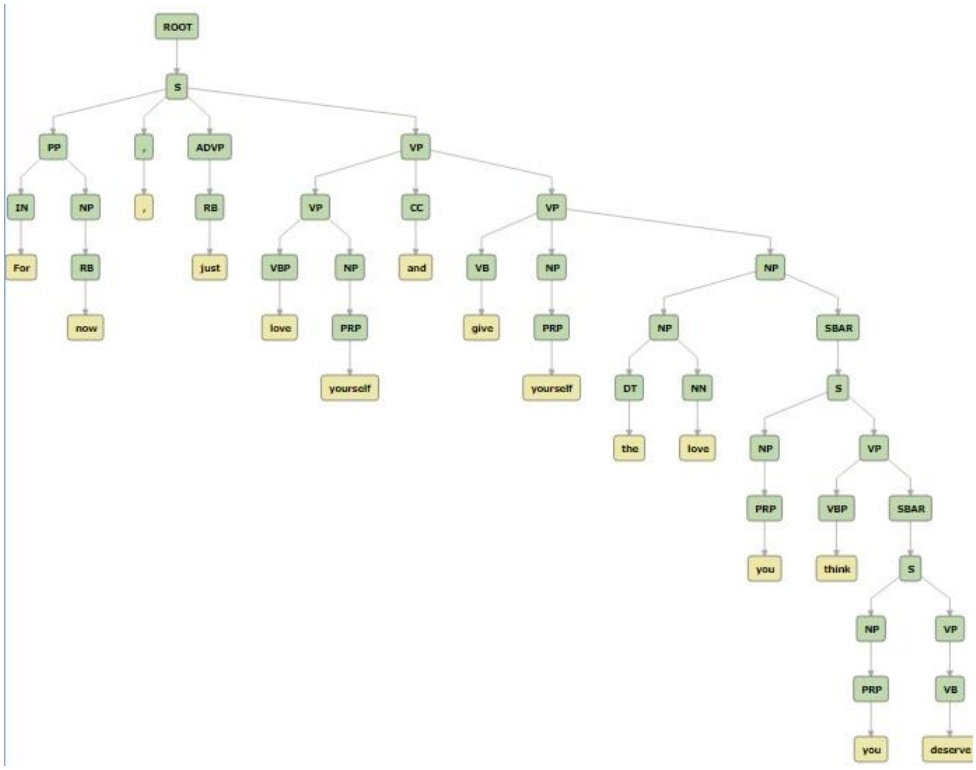



Figura 5.6 Genararea arborelui– imagine realizată cu ajutorul *corenlp.run*

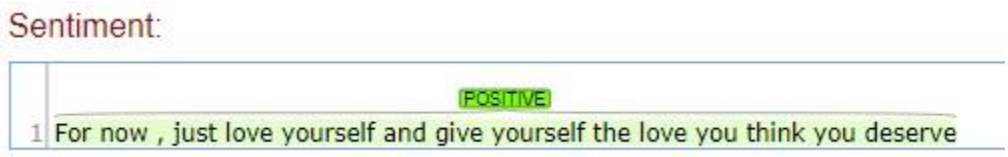



Figura 5.7 Clasificarea întregului tweet– imagine realizată cu ajutorul *corenlp.run*

## 5.3. Arhitectura TweetSent

Implementarea unui sistem de procesare în timp real în Apache Spark presupune proiectarea la nivel de detaliu a arhitecturii, care va fi apoi transpusă în cod. În continuare vor fi descrise componentele acesteia.

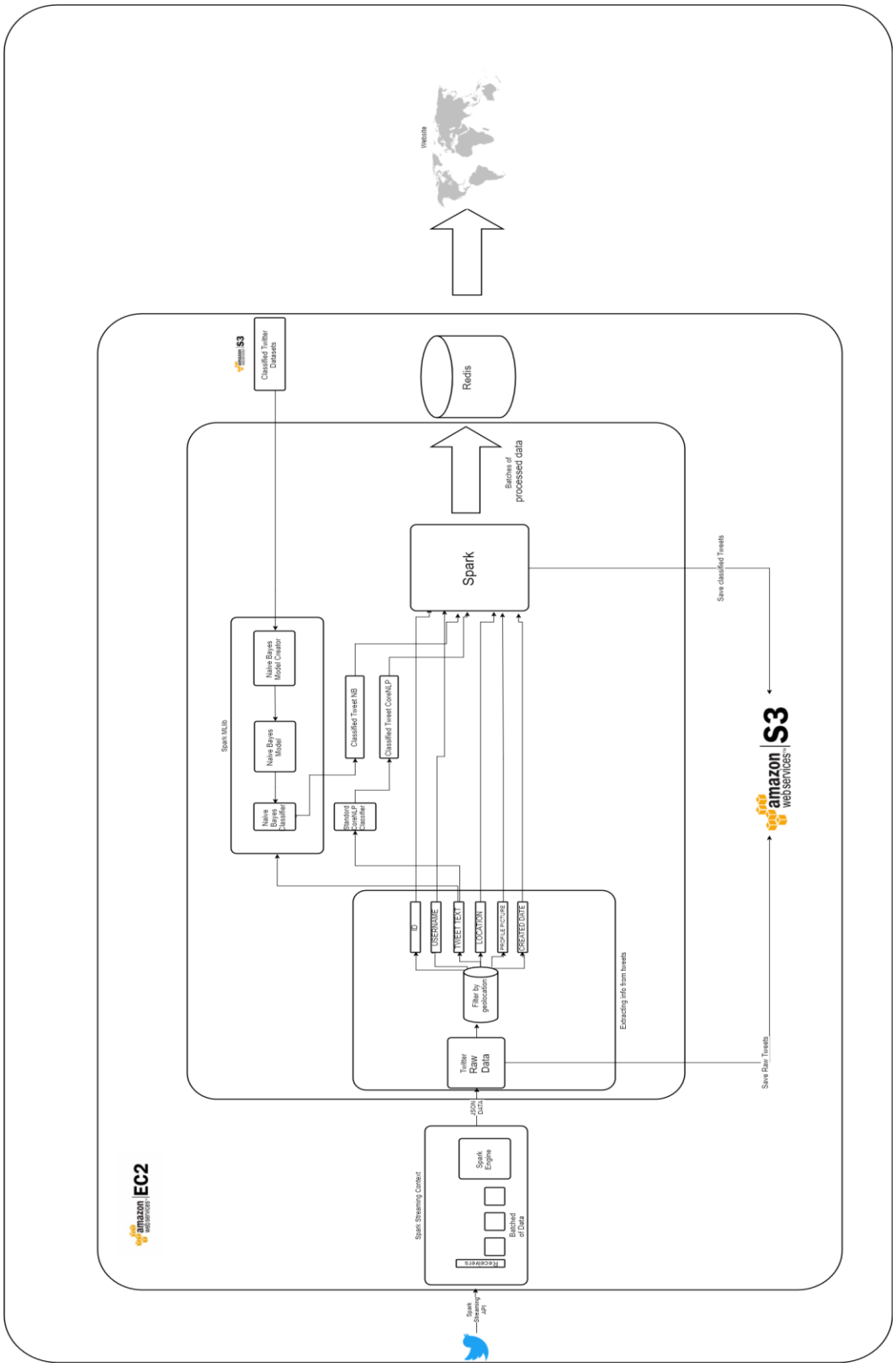



Figura 5.8 Arhitectura TweetSent

TweetSent este o aplicație de procesare si analiză a tweet-urilor în timp real, folosind Apache Spark. Pentru a implementa o astfel de aplicație, se impune folosirea componentelor Spark Streaming, SparkSQL și Spark Mllib.

După autentificarea cu Twitter pentru a putea extrage informații, Spark Streaming împarte fluxul live de date în batch-uri (numite micro-batch-uri), de un interval de timp predefinit, de N secunde și tratează fiecare batch ca Resilient Distributed Datasets (RDD). În continuare acest lucru permite procesarea acestor RDD-uri folosind operatori precum map, reduce, reduceByKey, join și window. Toate aceste rezultate, de după etapa de procesare, sunt returnate tot sub formă de batch-uri.

E foarte important intervalul de timp a batch-urilor într-o aplicație de streaming, deoarece, în cazul în care această valoare N este prea scazută, micro-batch-urile nu vor avea suficiente date pentru a returna rezultate semnificative în timpul analizei. La începutul fiecărui interval se creează un nou batch și toate datele care sosesc în intervalul respectiv se adaugă acelui batch. La sfărșitul intervalului de timp, batch-ul nu mai crește. Fiecare batch creează un RDD și este procesat folosind funcțiile Spark pentru a crea alte RDD-uri. Rezultatele procesare sunt apoi trimise către alte sisteme. Această arhitectură la nivel înalt, este prezentată în Figura 5.9.

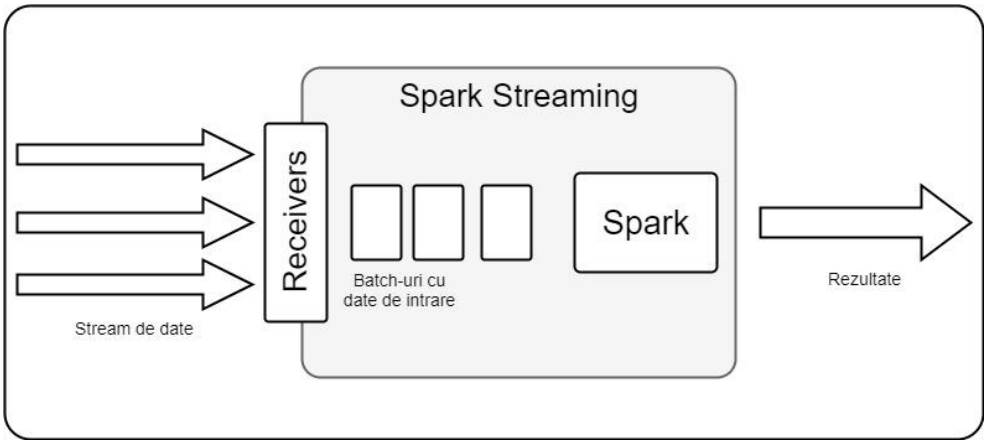



Figura 5.9 Arhitectura Spark Streaming

După cum se poate observa în Figura 5.9, pentru orice sursă de intrare, Spark Streaming lansează receiveri, care sunt task-uri care rulează în executorii aplicației și colectează date de intare și le salvează ca D-stream-uri. Acestea primesc date de intrare și le replică (în mod implicit) unui alt executor pentru toleranța la erori. Aceste date sunt stocate în memoria executorilor în același mod ca și D-stream-urile stocate în memoria cache.

D-stream-urile, sunt o secvență de RDD-uri (conform Figurii 5.10), unde fiecare RDD are cate o singură parte a datelor din stream.

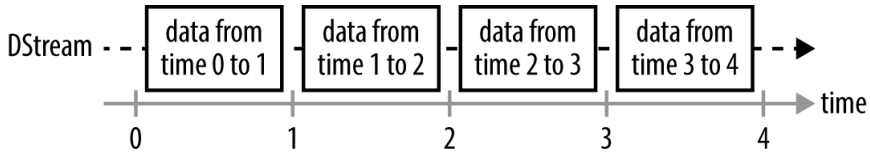



Figura 5.10 Arhitectura unui D-stream

TweetSent, creează D-stream-uri din datele primite de la Twitter, și după aplică o transformare de tip filtru, folosind metoda *filter().* Acest lucru creează intern RDD-uri ca în Figura 5.11.

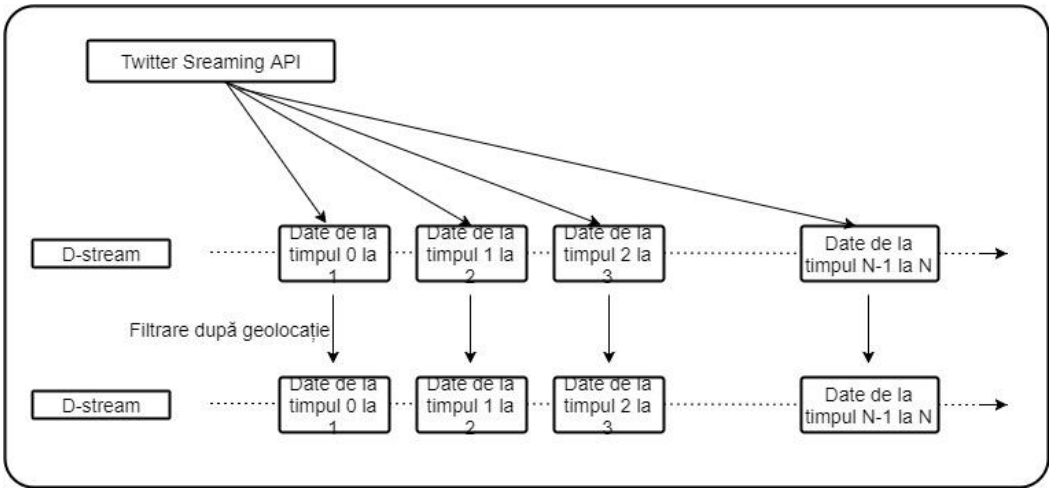



Figura 5.11 Crearea internă a secvențelor de RDD-uri dintr-un D-stream, după aplicarea unei transformări

Comparativ cu Spark Streaming, alte framework-uri de procesare în timp real, procesează fluxurile de date pentru fiecare eveniment, nu ca micro-batch-uri. Abordarea micro-batch permite utlizarea altor biblioteci Spark, în aceeași aplicație.

Spark Streaming Context este punctul principal de intrare pentru toate funcțiile de streaming și dispune de metode integrate pentru primirea datelor streaming în programul TweetSent.

Transformările aplicate D-stream-urilor pot fi împărțite în două categorii: transformări stateless și transformări stateful. În transformările stateless, prelucrarea fiecărui batch nu depinde de date din alte batch-uri anterioare. Printre aceste transformări se numără funcții *map(), filter()* sau *reduceByKey().* Transformările stateful dimpotrivă, folosesc date sau rezultate intermediare din batch-uri anterioare pentru a calcula rezultatul batch-ului curent.

Sistemul curent foloseste transformări stateless. După etapa de filtrare aplicată fiecărui RDD din D-stream-uri, se folosește funcția *map(),* metodă care permite aplicarea unei funcții fiecărui element din D-stream, returnând tot un D-stream ca rezultat. În această funcție de *map()*, se vor aplica cele două metode de analizare a sentimentului, pe fiecare tweet, și anume, Naive Bayes și Stanford CoreNLP. În prealabil se va crea și testa modelul Naive Bayes pe un set de antrenament, conform descrierii subcapitolului 5.2.2.1.

După aplicarea tuturor transformărilor și analizelor pe text, tweet-urile clasificate vor fi publicate pe un canal Redis, care vor fi apoi redirecționate către interfața web.

După definirea logicii de calcul a fluxului, sistemul este pregătit de primirea și procesarea datelor, lucru care se începe folosind metoda de pornire din obiectul Streaming Context, *start().* În cele din urmă așteptăm ca întregul proces să fie oprit folosind metoda *awaitTermination()* a contextului de streaming.

Întreg sistemul folosește Amazon S3 pentru stocarea datelor și Amazon EC2 pentru deploy-ul în cloud, o instanță free t2.micro, de 1GB RAM și 30 GB storage.

## 5.4. Implementarea arhitecturii TweetSent

Implementarea arhitecturii sistemului presupune transpunerea acesteia în cod Java. Fiecare componentă corespunde unei clase, iar nivelul de paralelism, stream-urile prin care comunică componentele și tipul de grupare a acestora sunt specificate la nivelul clasei care implementează arhitectura.

### *5.4.1. Componentele sistemului TweetSent*

**TwitterSentimentAnalysis** este componenta principală, care primește stream-urile de date, le procesează și le trimite către Redis pentru a putea fi disponibile pentru vizualizare.

În primul rând, se va seta contextul Spark de streaming, având doi parametri - configurația aplicației și timpul de streaming. Dat fiind faptul că Spark transmite date în micro-batch-uri, trebuie stabilită o anumită perioadă de timp, astfel încât pentru fiecare set de timp (time_set), fie secunde sau milisecunde, se vor transmite date. În TweetSent se vor seta 15 secunde, deci pentru fiecare 15 secunde, vor fi transmise datele de pe Twitter.

După cum a fost menționat mai sus Twitter-ul are nevoie de 4 chei pentru a autentifica un utilizator, adică cheia cosumatorului (Consumer Key), cheia secretă a consumatorului (Consumer Key), token-ul de acces (Access Token) și parola acestuia, token-ul de acces secret (Secret Access Token), toate acestea vor fi transmise ca argumente pentru program. Apoi, se va folosi clasa ConfigurationBuilder pentru a lua cheile pentru autentificarea Twitter. După toate acestea va începe streaming-ul Spark utilizând clasa TwitterUtils.createStream(), care va lua contextul de streaming și detaliile de autorizare ca parametri.

Acum, aplicația de streaming poate transmite datele sau să le stocheze. Tweet-urile sunt redate în obiecte de tipul Java D-streams, care sunt filtrare dupa geolocatie (sistemul are nevoie doar de acele tweet-uri care au specificate coordonatele geografice, pentru a putea fi redate pe hartă). Structura unui JSON de tweet are foarte multe informații irelevante pentru sistemul TweetSent, asa că vor fi utlizare doar câteva dintre ele, respectiv id-ul tweet-ului, username-ul utilizatorului, textul postat, coordonatele geografice, url-ul imaginii de profil și data creării. Textul tweet-urilor în limba engleză este apoi analizat cu ajutorul claselor MllibSentAnalyzer și CoreNlpSentAnalyzer, descrise în cele ce urmează și salvat într-un obiect de tip Tuple9. Restul tweet-urilor în alte limbi, sunt clasificate ca fiind neutre.

După clasificarea acestuia, întreg tuplul este publicat pe canalul “TweetChannel” al Redis-ului, care rulează pe localhost, port 6379.

În tweet-urile redate, vom primi diferite limbi de tweets, din care analizam spre sentimente doar cele în engleza

**NaiveBayesModelCreator** folosește Spark MlLib cu algoritmul Naïve Bayes pentru a crea un model de clasificare a tweet-urilor. Pentru acest lucru, se creează un context Spark, se alege cel mai potrivit dataset pentru a instrui și a testa clasificatorul și a prezice etichetele din noile recenzii pe baza atributului lor de text. Setul de date trebuie împărțit în seturi de testare și instruire.

Naïve Bayes este algoritmul de machine learning care poate funcționa eficient pentru clasificarea datelor. Este un clasificator probabilistic care atribuie etichete valorilor caracteristice, în timp ce această etichetă este preluată de la un set de date cunoscut. Clasificatorii Bayes au ipoteze despre caracteristici, în care o caracteristică este independentă de cealaltă, dacă aparțin unei anumite clase, adică pozitive, negative sau neutre. Mai mult decât atât, Naïve Bayes alege eticheta pentru datele de intrare pe baza probabilității etichetelor așa cum apar în setul de antrenament. Probabilitatea estimării este determinată pentru fiecare etichetă cu referință de estimare a datelor de intrare.

Dataset-ul de tip .csv pe care urmează a fi facut antrenamentul se încarcă într-un obiect de tipul DataFrame, obiect care stochează datele ca o colecție de date organizată în coloare. Deoarece acest document csv are mai multe coloane, dintre care doar două ne interesează în procesul de creare a modelului, se va da discard celorlalte din obiectul DataFrame. Au rămas doar coloanele de polaritate și text ale tweet-ului, salvate într-un obiect JavaRDD<LabeledPoint>, după ce se parsează fiecare tweet, aplicându-se expresii regulate și eliminându-se stop word-urile, conform celor descrise în subcapitolul 5.2.1.

Modelul Naive Bayes utilizat în sistemul TweetSent este creat după cum arata Figura 5.12, folosindu-se un clasificator multinomial și un parametru lambda = 1.0. Clasificatorul Multinomial clasifică precizia textului și a ieșirilor, în timp ce ia în considerare numărul de cuvinte ca, caracteristică, iar parametrul lambda este folosit pentru a seta un “additive smoothing”.

```
NaiveBayesModel naiveBayesModel = NaiveBayes.train(labeledRDD.rdd(), 1.0, "multinomial");

naiveBayesModel.save(jsc.sc(), PropertiesLoader.naiveBayesModelPath);
```

Figura 5.12 Cod reprezentând comenzile de creare și salvare a unui model Naive Bayes

Pentru a verifica acuratețea acestui model, înainte de a-l aplica pe tweet-uri în timp real, s-au făcut teste pe mai multe seturi de date, enumerate și prezentate în subcapitolul 5.2.2. Testarea acurateței constă în aplicarea modelului pe un anumit set de date deja clasificat, pentru a putea compara rezultatele.

**MLlibSentAnalyzer** este clasa care folosește modelul Naive Bayes creat pentru a prezice sentimente pe textele tweet-urilor. Înainte de a aplica modelul, tweet-ul este parsat și transformat într-un vector de cuvinte. Polaritatea tweet-ului este normalizată în felul următor: dacă după aplicarea modelului, aceasta este egală cu „4”, înseamna că tweet-ul este unui pozitiv și i se atribuie valoarea „1”, dacă este „0” înseamnă că este unul negativ și va primi valoarea „-1”, iar dacă este „2”, va fi normalizat cu valoarea „0”. Această polaritate este trimisă înapoi la TwitterSentimentAnalysis.

În cele din urmă, **CoreNlpSentAnalyzer** este a doua clasă care analizează conținutul fiecărui tweet și ii atribuie polaritatea corespunzătoare. CoreNLP aplică o serie de adnotări textului, cum ar fi “tokenize, ssplit, pos, lemma, parse, sentiment”, explicate în capitolul 5.2.2.2, după care creează un arbore din cuvintele tweet-ului și prezice plaritatea sentimentului.

Diagrama corespunzătoare acestor clase este transpusă în Figura 5.13

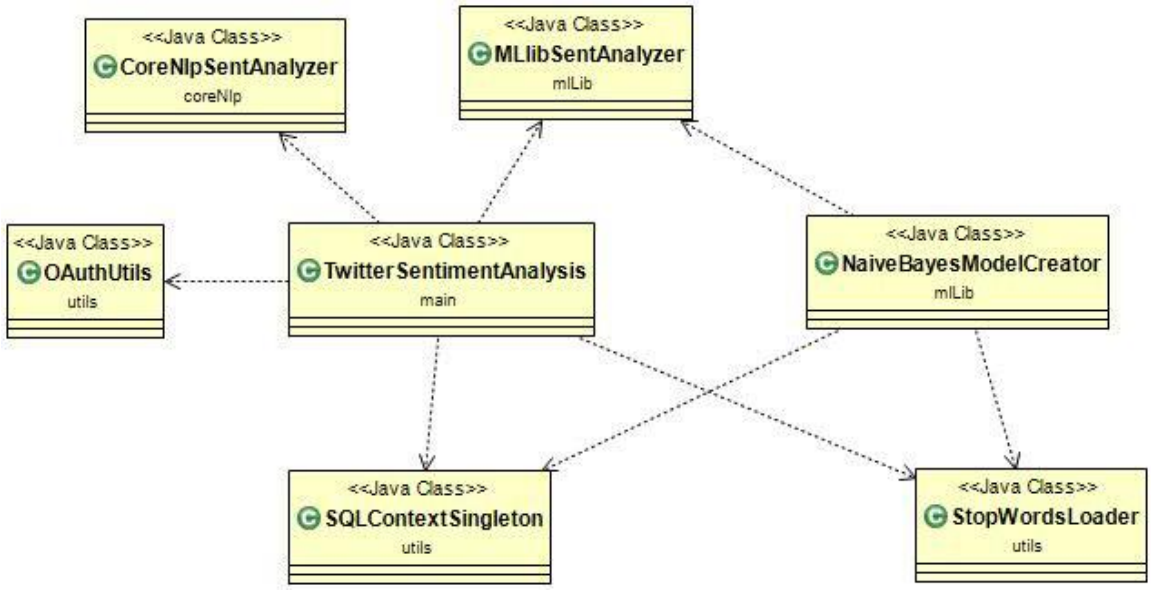



Figura 5.13 Diagrama simplificată de clase a TweetSent

# Capitolul 6. Testare şi Validare

Apache Spark poate implementa soluția pentru exploatarea de date Twitter de clasificare a textului și analiză a sentimentului, la nivel scalabil. Arhitectura necesită noduri în clustere, care vor funcționa în paralel pentru procesarea Twitter și alte tehnologii de ultima generație.

Apache Spark poate rula în diferite moduri:

- Modul standalone local - procesele Spark funcționează ca parte a procesului Java Virtual Machine
- Modul standalone cluster – Spark rulează într-un grup de noduri atunci când framework-ul de planificare a job-ului este aplicabil
- YARN – o resursă de management și design arhitectural central pentru sistemele de calcul Hadoop cluster
- MESOS – o open source de abstractizare a kernelului pentru calculele distribuite în clustere.

Mai mult, clusterele Spark pot fi implementate de platforme terțe, cum ar fi Elastic Cloud Compute de Amazon, IBM Bluemix sau Microsoft Azure HDInsight.

TweetSent a fost testat în trei moduri diferite: modul standalone local, modul standalone local pe o instanță Amazon EC2 și modul standolone cluster pe Amazon EC2.

Putem observa cu ajutorul interfeței Spark, că în modul standalone local și în Amazon EC2, nu există mari diferențe legate de performanțele de rulare, timpul de procesare a batch-urilor fiind aproximativ același. Figura 6.1 ilustrează timpii de execuție și de procesare a input-ului sub formă de batch-uri pe aceste două moduri.

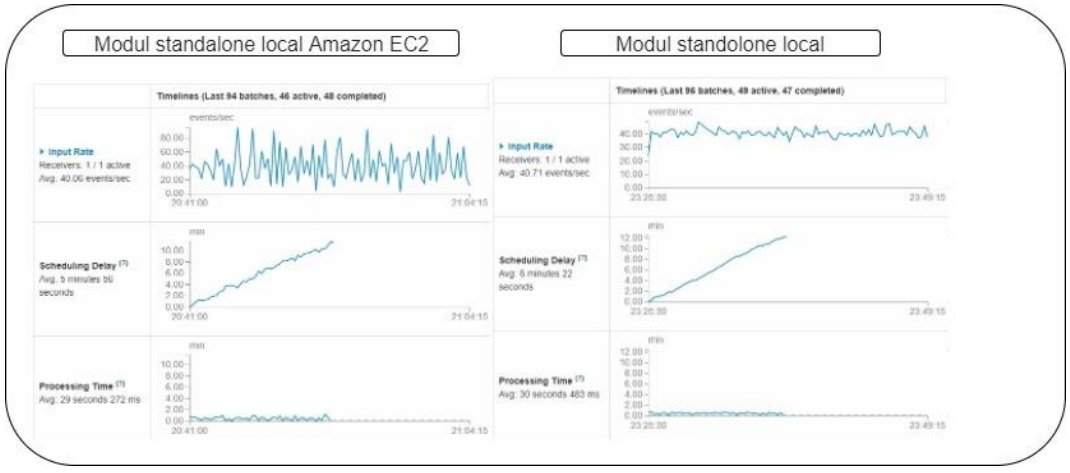



Figura 6.1 Modul standalone local vs. Modul standalone local în host-uit în Amazon EC2

Al treilea mod, cel de clusterizare a fost realizat tot în Amazon EC2, alcătuindu-se un cluster de trei noduri, un nod master și doi slaves. Amazon Free Tier pune la dispoziție un singur tip instanță, o instanță t2.micro, cu următoarele performanțe hardware: 1 CPU, 1GB RAM și maximul de 30 GB storage.

Pentru a concepe clusterul, s-au urmărit o serie de pași. Primul lucru a fost crearea separată a fiecărui nod, în Amazon EC2. Pentru un management mai bun al acestor noduri, a fost instalat FileZilla[45], care a fost de ajutor pentru a încărca ușor resurse în fiecare nod. Al doilea pas a fost instalarea resurselor necesare rulării sistemului, Java, Spark și Redis. După cum se poate observa și în arhitectura din Figura 6.2, nodul master, trimite task-uri către ceilalți doi workeri. Pentru a se putea întâmpla asta, trebuie setat nodul de master, în fișierul de configurări Spark al fiecărui nod, și nodurile slave menționate în fișierul *slaves* al nodului master, pentru ca acesta să știe unde să direcționeze task-urile. Ultimul pas, este acela de a configura legăturile de ssh între toate aceste noduri, pentru a putea comunica unul cu altul, respectiv masterul cu ambii slaves si fiecare slave cu masterul. Clusterul va putea fi pornit prin comanda *./sbin/start-all.sh* , executată pe nodul master. Pentru rularea întregului sistem, se va executa imediat după, comanda

*./bin/spark-submit --class main.TwitterSentimentAnalysis --master spark://ip-172-31-26-211:7077 /home/ec2-user/jar/twitterStreamingSparkAWS-SNAPSHOT-1.0.jar,* care va porni aplicația. Din păcate, resursele puse la dispoziție de Amazon Free Tier nu sunt suficiente pentru rularea acestei aplicații, timpul în care memoria ajunge la saturație este foarte scurt.

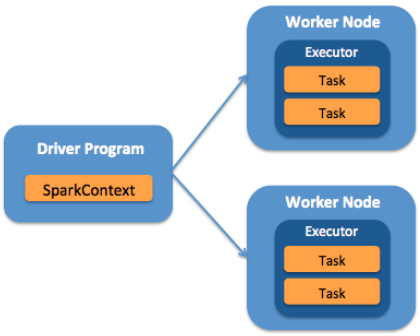



Figura 6.2 Arhitectura unui cluster Spark

În ceea ce privește analiza de sentimente, s-au folosit mai multe seturile de date pentru instruirea modelelor pentru o acuratețe cât mai ridicată.

[45] https://ro.wikipedia.org/wiki/FileZilla

# Capitolul 7. Manual de Instalare si Utilizare

Pentru o rulare cu succes a TweetSent, environment-ul pe care se dorește rularea, trebuie să fie pregătit în prealabil cu următoarele:

- **JDK 1.8.0_121:** http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- **Eclipse Neon:** http://www.eclipse.org/downloads/eclipse-packages/
- **Apache Maven 3.5.0:** https://maven.apache.org/docs/3.2.5/release-notes.html
- **Apache Spark 1.6.0:** https://spark.apache.org/downloads.html
- **Redis 3.2.9:** https://redis.io/download
- **Python 3.6.1** https://www.python.org/downloads/release/python-361/
- **Flask 0.12.2:** https://pypi.python.org/pypi/Flask

După instalarea tuturor tool-urilor, proiectul poate fi rulat în două moduri, din IDE-ul Eclipse, sau din command prompt. Pentru rularea din cmd, se va pregăti înainte un jar, prin comanda *mvn package* din folderul aplicației, iar apoi se va rula cu java *–jar TweetSent.jar*. Pentru vizualizarea interfeței, e nevoie de rularea fișierului *app.py* din folderul *webapp,* folosind comanda *python.exe app.py,* se deschide un broswer web și se accesează localhost:9999.

Pentru a folosi Apache Spark în EC2, va trebui configurat un cont Amazon AWS. Acest cont se poate crea pe http://aws.amazon.com/. Următorul lucru care trebuie făcut este accesarea consolei AWS IAM și selectarea opțiunii *Utilizator*. Aici se vor crea credențialele de acces la instanțe. Aceste credențiale sunt AWS_ACCESS_KEY_ID și AWS_SECRET_ACCESS_KEY. După acest pas, se creează cheia de acces, un fișier cu expensia *.pem,* care va trebui downloadată într-un loc sigur. Odată executați aceși pași, se pot crea instanțele, din dashboard-ul Amazon EC2, definind câtă memorie și câte core-uri să aiba disponibile. Recomandarea minimă este de o memorie de minim 8GB RAM, cu unu sau două core-uri, pentru fiecare nod din cluster. După crearea instanțelor, se vor instala cerințele minime pentru rularea aplicației în cluster pe fiecare nod, și anume, Apache Spark 1.6.0, Java 8, Redis 3.2.9. Pe fiecare nod, se va copia și jar-ul aplicației. Se va alege master-ul și slaves, iar pentru crearea legaturii dintre ei, se vor face următoarele configurări:

- Pe fiecare nod, se va crea câte un fișier *spark-env.sh* în care va fi precizat ip-ul masterului, prin *SPARK_MASTER_IP = <ip_master>*. Acest fișier se va copia in folderul *conf,* al Spark-ului spark-1.6.1-bin-hadoop2.6\conf
- Pe master se va crea un fișier *slaves* în care se pun unul sub altul ip-urile slave-ilor
- Pe fiecare nod, se vor crea configurările de ssh, cu următoarele comenzi, în folderul /home/ec2-user/.ssh: *ssh-keygen -t rsa -P "" / cat id_rsaSlave1.pub >> authorized_keys* (în nodul master se vor copia cheile slave-ilor și în fiecare slave cheia masterului)

Acum, clusterul este pregătit de execuția sistemului TweetSent, în modul standalone cluster. Pentru acest lucru, trebuie pornite spre execuție toate nodurile, din

folderul ului spark-1.6.1-bin-hadoop2.6, cu comanda *./sbin/start-all.sh.* După ce masterul și workerii au pornit, aplicația se rulează cu ./bin/spark-submit --class main.TwitterSentimentAnalysis --master spark://<master_ip>:7077 /home/ec2-user/jar/twitterStreamingSparkAWS-SNAPSHOT-1.0-url.jar.

Pentru a putea vizualiza UI-ul pus la dispoziție de Spark, se va accesa <ip_master>:8080 și <ip_master>:4040.

# Capitolul 8. Concluzii

În acest capitol for fi prezentate realizările și obiectivele care au fost atinse prin această lucrare și prin implementarea prezentată. De asemenea, vor fi prezentate și o serie de îmbunătățiri care pot fi aduse sistemului TweetSent și enumerate posibile dezvoltări ulterioare.

## 8.1. Obiective atinse

Precum a fost descris în capitolul 2, scopul prezentei lucrări a fost definirea conceptelor de bază din domeniu Big Data și a diferitelor metode de procesare, documentarea aprofundată a domeniului de procesare masivă de tip streaming și motivarea interesului pentru tehnologii noi de procesare a cantității enorme de date, care definesc potențialul ridicat în descoperirea informațiile cheie în diferite domenii de actualitate și de viitor. Având subiectul centrat în jurul procesării în timp real, lucrarea a prezentat în detaliu framework-ul folosit, dar și câteva din framework-urile recent apărute. Framework-ul Spark a dovedit o performanță bună atât în decursul ultimilor ani în industrie, cât și în cadrul aplicației TweetSent.

În ceea ce privește rezultatele obținute prin testarea și validarea aplicației TweetSent, acestea au fost conform obiectivelor propuse. Sistemul a dovedit a fi scalabil, tolerant la eșec, iar cu un timp de răspuns de ordinul milisecundelor putem afirma că procesarea datelor este una în timp real. Mai mult, conform obiectivelor propuse pentru aceasta, rezultatele au arătat veridicitate și au reflectat realitatea lumii în care trăim, prin determinarea sentimentelor pe textele postate în rețeaua social Twitter.

Conform așteptărilor, framework-ul Apache Spark a dovedit aptitudini de procesare de stream-uri de date in timp real. Lucrarea a reușit să pună în evidentă principalele caracteristici ale acestora, printe care se numără viteza de procesare, latența redusă, toleranța la eșec, scalabilitatea orizontală, ușurința de dezvoltare a aplicațiilor, reutilizabilitatea codului și nu numai.

## 8.2. Posibile dezvoltări ulterioare ale sistemului TweetSent

Cu toate că obiectivele de a realiza o aplicație real time de analiză de sentimente folosind framework-ul Apache Spark au fost îndeplinite, există o serie de îmbunătățiri care pot fi aduse sistemului actual.

În ceea ce primeste clusterizarea, una din îmbunătățiri este crearea unui cluster cu performanțe hardware avansate, pentru a permite procesarea cu succes a datelor pe mai multe noduri. O altă îmbunătățire ar fi adaugarea sistemului Apache Kafka, pentru a organiza datele în grupuri de consumatori și topic-uri și pentru o mai bună performanță în ceea ce privește viteza de streaming a tweet-urilor.

În legătură cu analizarea și clasificarea tweet-urilor, sistemul dispune de algorimi de analizare a textelor doar în limba engleză, deci o altă dezvoltare ar fi permiterea analizei de sentimente pe mai multe limbi, ceea ce presupune crearea de seturi de date în limbile respective, pentru a instrui un model pentru fiecare limbă în parte. Crearea setului

de date se poate face manual, pentru o mai bună acuratețe. Antrenarea modelului conține aceiași pași ca la limba engleză.

De asemenea, soluții pentru îmbunătățirea acurateții algoritmilor actuali poate deveni o provocare pentru a dezvolta sistemul, dar și implementarea unor noi algoritmi.

O dezvoltare semnificativă ar fi și migrarea întregului sistem pe framework-ul Apache Beam, a cărui versiune stabilă a fost publicată în primăvara acestui an, 2017. Deoarece permite rularea pe mai mulți runneri, sistemul TweetSent va putea fi rulat și cu Apache Spark și cu Apache Flink.

# Bibliografie

[1] Farouk Salem, *Comparative Analysis of Big Data Stream Processing Systems***,** Teză master**,** Aalto University School of Science, iunie 2016. [Online]: https://aaltodoc.aalto.fi/bitstream/handle/123456789/21577/master_Salem_Farouk_2016.pdf?sequence=1

[2] Ketaki Subhash Raste, *Big Data Analytics-Hadoop Performance Analysis*, Teză master, Faculty of San Diego State University, aprilie 2014. [Online] https://sdsu-dspace.calstate.edu/bitstream/handle/10211.3/120375/Raste_sdsu_0220N_10274.pdf?sequence=1

[3] Aftabl A. Chandio, Nikos Tziritas, Cheng-Zhong Xu, *Big-Data Processing Techniques and Their Challenges in Transport Domain*, Research Gate, februarie 2015.

[4] Institutul național de cercetare-dezvoltare în informatică, *Utilizarea tehnologiilor Big Data în sistemele informaționale guvernamentale*, noiembrie 2015. [Online]: http://comunicatii.gov.ro/wp-content/uploads/2016/02/BigData_nov2015.pdf

[5] Doug Laney, *3D Data Management: Controlling Data Volume, Veocity, and Variety*, META Group, februarie 2001. [Online]:https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

[6] C.L. Philip Chen, Chun-Yang Zhang. *Data-intensive applications, challenges, techniques and technologies: A survey on Big Data*, Information Sciences Volume 275, august 2014. [Online]: http://dx.doi.org/10.1016/j.ins.2014.01.015

[7] Eugene Ciurana and Masoud Kalali, *Getting Started with Apache Hadoop*, DZone Refcardz.

[8] B. Ellis, *Real-time analytics: Techniques to analyze and visualize streaming data*. John Wiley & Sons, 2014.

[9] M. Stonebraker, U. Çetintemel, and S. B. Zdonik, *The 8 requirements of real-time stream processing*, *SIGMOD Record*, vol. 34, no. 4, pag 45-79, 2005.

[10] Wolfram Wingerath, Felix Gessert, Steffen Friedrich, and Norbert Ritter, *Real-time stream processing for Big Data*, ianuarie 2016. [Online]: https://vsis-www.informatik.uni-hamburg.de/getDoc.php/publications/561/Real-time%20stream%20processing%20for%20Big%20Data.pdf

[11] 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, decembrie 2014.

[12] Najeefa Nikhat Choudhury, *Sentiment Analysis of Twitter Data for a Tourism Recommender System in Bangladesh,* Aalto University School of Science noiembrie 2016.

[13] B. Pang, L. Lee, and S. Vaithyanathan, *Sentiment Classification using Machine Learning Techniques,* CoRR, cs.CL/0205070, 2002.

[14] V. Hatzivassiloglou and K. McKeown, *Predicting the Semantic Orientation of Adjectives,* Universidad Nacional de Educación a Distancia iulie 1997.

[15] V. Hatzivassiloglou and J. Wiebe, *Effects of adjective orientation and gradability on sentence subjectivity*, iulie 2000, Universität des Saarlandes, Saarbrücken, Germany.

[16] W. Medhat, A. Hassan, and H. Korashy, *Sentiment Analysis algorithms and applications: A survey,* Ain Shams Engineering Journal, 2014.

[17] I. Rish, *An empirical study of the Naive Bayes Classifier*. In IJCAI 2001 workshop on empirical methods in artificial intelligence, volume 3, IBM New York, 2001.

[18] R. D. Schneider, *Hadoop for Dummies*, John Wiley & Sons Canada, Ltd., special edition, 2012.

[19] *How Much Data Is Generated Every Minute On Social Medi*a?, WeRSM | We Are Social Media. [Online] http://wersm.com/how-much-data-is-generatedevery-minute-on-social-media/.

[20] Apache Spark, *Lightning-fast cluster computing*, [Online] http://spark.apache.org.

[21] S. Haridi, *Research Challenges in Data-Intensive Computing*, The stratosphere Project, Apache Flink, Ericsson [Online] https://www.sics.se/sites/default/files/pub/sh-stratosphere-format_16- 0.v1_0.pdf.

[22] F. Beligianni, *Streaming Predictive Analytics on Apache Flink*, [Online]. Available: http://www.divaportal.se/smash/get/diva2:843219/FULLTEXT01.pdf;jsessionid=amPOFJ_4yWQESez47vK7vW9GOHMmr1bs8HgtZzWS.diva2-search7-vm.

## Anexa 1 -Lista figurilor și a tabelelor din lucrare