MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

FILTRAREA MESAJELOR ÎN SERVICII DE MICRO-BLOGGING

LUCRARE DE LICENŢĂ

Absolvent: **Adrian Cornel PETRI**

Coordonator ştiinţific: **Șef lucrări ing Cosmina IVAN**

2014

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Adrian Cornel PETRI**

FILTRAREA MESAJELOR ÎN SERVICII DE MICRO-BLOGGING

1. **Enunţul temei:** *Implementarea unei aplicaţii care să filtreze mesajele aparţinând unui serviciu de micro-blogging.*

2. **Conţinutul lucrării:** *Introducere,Obiectivele proiectului, Studiu bibliografic, Analiză şi fundamentare teoretică, Proiectare de detaliu şi implementare, Testare şi validare, Manual de instalare şi utilizare, Concluzii, Bibliografie, Anexă*

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

4. **Consultanţi**:

5. **Data emiterii temei:** 1 februarie 2014

6. **Data predării:** 11 septembrie 2014

Absolvent: _____

Coordonator ştiinţific: _____

Cuprins

Capitolul 1. Introducere

1.1. Contextul proiectului

Internetul este o retea vastã de calculatoare care leagã milioane de reţele mai mici din întreaga lume (inter = între, net = retea). Internetul este o modalitate de comunicare a calculatoarelor. O reţea este un grup de calculatoare si echipamente de calcul conectate pentru a partaja informaţii şi resurse. Calculatoarele dintr-un birou, sunt des interconectate pentru a putea utiliza aceleasi fişiere si imprimante. Toate calculatoarele legate la Internet pot schimba informaţii între ele. Este la fel de usor de comunicat cu un calculator din cealalta parte a lumii ca si cu unul din aceeasi camera. Înainte de a putea schimba informatii, douã calculatoare de pe Internet trebuie sa se "gaseascã" reciproc şi sã comunice într-un limbaj accesibil amândoura.

Odatã cu dezvoltarea internetului a apãrut nevoia omului de a discuta asupra anumitor teme cu alţi utilizatori sau de a afla pãreriile lor. De asemenea omul a simţit nevoie de a-şi putea scrie gândurile online, atât sub anonimat cât şi sub numele lor adevãrat cu scopul de a se descarca. În acest context a apãrut termenul de blogging. Definiţia pe care Wikipedia o dã blogului este: "Blogul este un website în care intrãrile sunt scrise in ordine cronologicã şi afişate in ordine cronologicã inversã. Cuvantul blog poate avea şi functia gramaticalã de verb, insemnând menţinerea sau adãugarea de conţinut într-un blog." Primul blog a apãrut în anul 1997 şi a prins un contur tot mai mare ajungând ca în zilele noastre cuvântul „blog" în limba englezã sã fie atât substantiv cât şi verb(to blog something)[1].

Deşii blog-ul era foarte popular, postãrile aici erau de obicei lungi iar utilizatorii pierdeau mult timp cãutând informaţii utile. În acest context apare micro-bloggingul. Micro-bloggingul este un mediu care existã sub formã de blogging. Un microblog diferă faţã de un blog tradiţional prin conţinutul său care de obicei este mai mic atât în mărimea mesajului dar şi a fişierului. Microblogul permite utilizatorilor să facă schimb de elemente mici de conţinut, cum ar fi fraze scurte, imagini individuale, sau link-uri video. Aceste mesaje mici sunt numite micro-postări.

Ca şi în cazul blogging-ului tradiţional, micro-bloggeri pot posta mesaje despre teme care variazã de la cele simple, cum ar fi "ceea ce fac eu acum," la cele tematice, cum ar fi "maşini sport". De asemenea existã si micro-bloguri comerciale , pentru a promova site-uri web, servicii şi produse,etc.

Utilizatorii aveau acum la dispoziţie microbloging-ul şi puteau cãuta sau posta mesaje în interiorul serviciilor.Totuşi cãutarea presupunea ca utlizatorul sã verifice toate mesajele şi sã aleagã din interiorul lor doar pe acelea care îl interesau. Deşi în servicile de micro-blogging postãrile erau împãrţite pe conţinut, cãutarea în interiorul lor presupunea un timp îndelungat. În acest context apare filtrarea de mesaje dupã conţinut. În acest tip de sortare fiecare utilizator se presupune a lucra independent unul faţã de celãlalt. Ca şi rezultat sortarea dupã conţinutul mesajului poate exploata doar postãrile. Dupã aplicarea acestui filtru de mesaje se vor afişa doar informaţiile relevante faţã de mesajul nostru.

1.2. Provocãri

Pentru a putea filtra mesajele din cadrul micro-bloggingului va trebui să identificăm cele mai potrivite soluţii pentru realizarea unui set de provocări. În cele ce urmează vor fi enumarate câteva provocări identificate în cadrul proiectului de faţă:

Mesaje scurte. O provocare poate fi consideratã limitarea lungimii textului. În micro-bloggingul folosit pentru filtrare limitarea este de 500 de caractere.În termenii clasificării textele scurte conţin date distribuite de aceea poate fi consideratã o provocare clasificarea şi filtrarea lor.

Identificarea cuvintelor cheie. Este foarte important să detectăm cuvintele cheie ale mesajului pentru ca ulterior să putem filtra mesajele ca relevante şi irelevante faţă de topicul nostru. Mesajul nostru poate fi unul foarte general(„Tehnologia Informaţiei”), unul general(„Baze de date”), sau chiar unul foarte specific(„MySql Database”).

Limbaje informale. O altă problemă ar putea fi consideratã structura informală a serviciilor de micro-blogging. Aceasta este mai puţin structuratã, poate conţine abrevieri datoritã limitării textului iar utilizatorii modifică anumite cuvinte sau le combină cu emoticoane.Aşadar vocabularul este foarte mare iar surse externe precum Wikipedia nu vor fi suficiente pentru a aduce destule informaţii despre text.De asemenea este necesar să putem identifica care cuvinte sunt comune şi care cuvinte cheie sunt utile pentru clasificarea textelor. Datoritã limbajelor informale, numărul cuvintelor care trebuie blocat să intre în lista de mesaje trebuie sa fie mai mare.

Schimbări ale vocabularului. Vocabularul din cadrul serviciului de micro-blogging se schimbă în mod constant datoritã cuvintelor şi a frazelor noi. De exemplu, filme şi produse noi pot apărea ca şi denumiri şi pot deveni foarte populare.Astfel clasificarea de texte nu va putea fi una staticã, va trebui să răspundă la schimbările din cadrul vocabularului.

Limbaje diferite. Micro-bloggingul este folosit de utilizatori de pe întreaga suprafaţă a planetei şi de aceea mai multe limbi vor fi utlizate în postări.

Aşadar sistemul de filtrare va trebui să fie unul dinamic ,ar trebui să urmărească îndeaproape sistemul de micro-blogging şi să raspundă la mesaje. Pentru clasificarea textelor ar trebui depaşitã provocarea cu referire la limita textului, iar cuvintele să poatã fi extrase într-o modalitate informală şi multilingualã. În final sistemul ar trebui să fie capabil să identifice cuvintele cheie şi să filtreze mesajele irelevante.

Capitolul 2. Obiectivele Proiectului

2.1. Specificaţia proiectului

Proiectul de faţă propune implementarea unei aplicaţii care să se ocupe cu preluarea, prelucrarea si filtrarea mesajelor provenite dintr-un serviciu de micro-blogging. Aplicaţia trebuie să fie capabilă să dividă mesajele din micro-blogging in două mari categorii: relevante faţa de mesajul dat sau irelevante faţă de mesajul dat.De asemenea aplicaţia trebuie să fie capabilă sa filtreze mesajele chiar dacă acestea sunt în altă limbă, au in componenţa sa un url sau conţin cuvinte doar asemănătoare cu cele folosite in mesajul original.

2.2. Obiective generale

Obiectivul principal este de a construi un filtru de mesaje care să fie capabil să sorteze mesajele provenite dintr-un serviciu de micro-blogging. Pentru aceasta ne vom concentra in principal asupra conţinutului mesajelor aduse. Prin gruparea lor in mesaje relevante sau mesaje irelevante vom putea obţine doar mesajele care vor avea acelaşi subiect de discuţie sau idee principală cu mesajul dat.Studiile arată faptul că, două mesaje au idei aproximativ asemănătoare dacă cuvintele folosite in mesaje au aproximativ acelaşi câmp semantic.[2]

În primul rând ne vom concentra asupra extragerii cuvintelor principale din interiorul mesajului.Având in vedere faptul că, atunci când scriem un mesaj, folosim multe cuvinte de legătură, prepoziţii, pronume, conjucţii şi alte părţi de vorbire care nu sunt relevante, acestea vor fi excluse din algoritm.Pentru a face acest lucru, vom extrage din mesajele noastre cele mai utilizate 250 de cuvinte din limba engleză. Conform studiilor, masa de bază a unui vocabular este în principal formată din puţine cuvinte.Dacă extrăgeam mai multe cuvinte riscam faptul ca, mulţi termeni relevanţi pentru mesajul meu sa nu fie incluşi.

În ceea ce priveşte cuvintele relevante care vor fi folosite, acestora li se va da o pondere.Această pondere reprezintă in principal importanţa pe care cuvântul o va avea în mesajul repsectiv. Cu cât cuvântul apare mai des in mesaj, cu atât el va fi mai important. Ponderea cuvântului se va calcula doar în mesajul respectiv iar durata de execuţie a calculului va fi una foarte rapidă.

Micro-bloggingul asupra căruia se va face filtrarea permite doar mesaje cu lungimea de maxim 500 de cuvinte. Astfel, pentru a se face mai bine inţeles utilizatorul foloseşte link-uri pe care le adaugă in mesaj. Prin aceste link-uri se va redirecţiona cititorul către o altă paginã, paginã care va conţine mai multe informaţii in legatură cu subiectul precizat. Algoritmul utilizat işi propune să preia toata informaţia utilă din link-ul respectiv şi să o adauge mesajului. Astfel limitarea de 500 de cuvinte va putea devenii mult mai mare. De notificat ar mai putea fi şi faptul că mesajul se va adăuga mesajului iniţial, informaţia din acesta rămânând utilă în continuare.Prin informaţia provenită din link-uri vom putea imbunătăţii algoritmul de clasificare.

O altă provocare a proiectului ar fi mesajele în altă limbă. Întodeauna comunicarea a fost limitată la doar o singură limbă.Evoluţiile actuale din domeniul tehnologic au reuşit însă să rupă această barieră şi să facă traduceri între texte dintr-o limbă in alta. In cazul in care mesajul din micro-blogging este intr-o altă limbă decât engleza, se va face traducerea lui iar după traducere se va aplica algoritmul de filtrare.

De foarte puţine ori, in momentul in care comparãm douã mesaje, se întamplã sa gãsim aceleaşi cuvinte in exact aceeaşi formã. Astfel, deşi probabil rãdãcina cuvântului va fi aceeaşi,cuvintele vor diferi. O altã problemã des intâlnitã se referã la faptul cã, pentru a spune acelaşi lucru, persoane diferite folosesc cuvinte diferite. Algoritmul de filtrare işi propune sa poatã trece peste aceste impedimente, şi indiferent dacã cuvântul respectiv este o altã formã a cuvantului de bazã sau chiar şi un sinonim, sã funcţioneze exact ca şi cum cuvintele întâlnite sunt aceleaşi.

Pentru operaţia de filtrare clasificatorii sunt folosiţi pentru a determina care mesaje for fi relevante faţã de mesajul introdus de noi şi care vor fi irelevante.Clasificatorii vor fi construiţi in special în funcţie de ponderile cuvintelor care apar in mesaje. Astfel fiecare mesaj va avea o pondere, pondere care nu va însemna nimic altceva decât ponderea de relevanţã a textului.De notificat ar fi şi faptul cã fiecare din obiectivele descrise mai sus vor fi folosite pentru filtrarea mesajelor.

Sistemul de filtrare a fost construit ca fiind unul dinamic. Fiecare modificare a unui mesaj din baza de date aduce la rândul sãu o modificare ulterioarã a mesajului din filtru insã numai la urmatoarea rulare a filtrului.O altã caracteristicã care face ca sistemul nostru sa fie unul dinamic se referã la faptul cã, ponderea de calcul a mesajelor relevante sã se calculeze dinamic, in funcţie şi de mãrimea mesajului. Teoria din spatele acestei idei constã in faptul cã, conform studiilor,cu cât mesajul este mai lung,oamenii tind sã foloseascã mai puţine cuvinte de specialitate şi cãt mai multe uzuale, care sã il facã pe receptor sã se simtã familiar cu discuţia.În figura 2.1 ne este prezentatã schema generalã a proiectului

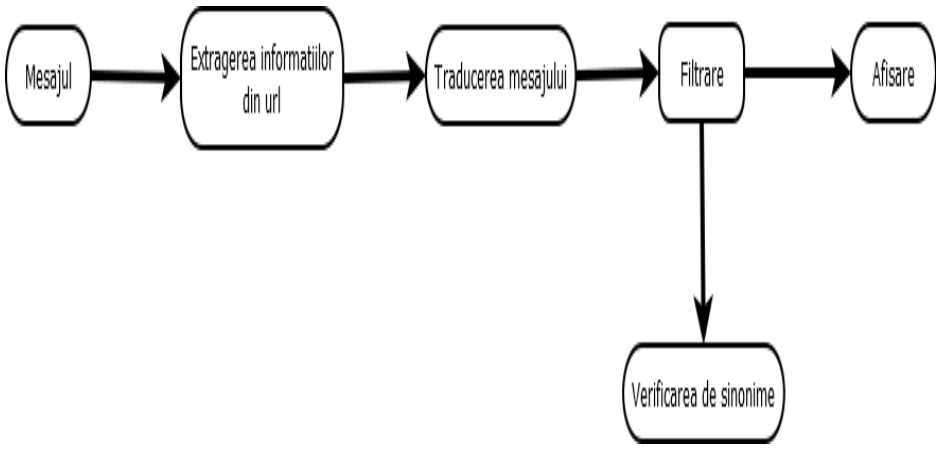

Figura 2.1 Schema generalã a sistemului

2.2.1. Cerinţe funcţionale

Cerinţele funcţionale reprezintã acele funcţionalităţi concrete pe care sistemul trebuie să le ofere astfel încât să răspundă cerinţelor de business. Se bazează pe analiza şi modelarea proceselor de lucru.

Principala cerinţã funcţionalã pe care proiectul trebuie sã o furnizieze este aceea cã trebuie sã ofere rezultate optime la un numãr mare de mesaje. Algoritmul de filtrare trebuie sã fie independent de numãrul de mesaje şi sã afişeze atatea rezultate cate va fi nevoie.

Filtrarea trebuie sa fie de asemenea independentã de mãrimea mesajului. Deşi micro-bloggingul este limitat la 500 de cuvinte, datoritã folosirii url-urilor si datoritã

preluării informaţiei utile din interiorul url-ului, mesajul nostru poate ajunge până la 10 000 de cuvinte sau chiar depăşi acest prag. Algoritmul este unul dinamic, de aceea numărul de cuvinte nu va putea influenţa sortarea.

Utilizatorul va putea introduce spre filtrare un mesaj care va trebui să aibă aceeaşi lungime ca şi mesajele din micro-blogging. În cazul in care mesajul va avea o lungime mai mare decât cea din micro-blogging utilizatorul va fi avertizat cu privire la faptul că a depăşit dimensiunea maximă a mesajului.De asemenea in cazul în care utilizatorul introduce un mesaj gol, acesta va fi avertizat asupra acestui fapt.

O altă caracteristică funcţională se poate referi la faptul că utilizatorul va putea vizualiza mesajele in funcţie de relevanţa lor. Astfel mesajul cel mai semnificativ, respectiv cel care are cea mai mare legătură cu mesajul dat de către noi, va fi afişat primul. Mesajul cel mai puţin semnificativ însă cu o semnificaţie mai mare decât valoarea de prag va fi afişat ultimul.

2.2.2. Cerinţe non-funcţionale

Cerinţele non-funcţionale reprezintă o constrângere de cost, calitate, construcţie, livrare sau operarea unui produs, componentă a unui produs, sau funcţia a unui produs.În domeniul software cerinţele non-funcţionale se referă la acele cerinţe care nu implică realizarea unei funcţionalităţi, dar care sunt necesare pentru ca funcţionalităţile să poată fi utilizate. Includ cerinţe de dimensionare, scalabilitate, securitate, performanţă, uşurinţă de utilizare.

Dimensiunea unei aplicaţii reprezintă numarul de kilobytes pe care aplicaţia o ocupă pe server.**Dimensionarea** reprezintă caracteristica care ne arată cât de compact este programul nostru. Direcţia de evoluţie a programării este aceea în care se insistă pe aplicaţii distributive. Serviciile web, aplicaţile locale, api-urile reprezintă modalitaţile prin care se vor îmbunătaţi atât distributivitatea cât şi dimensionarea. Cerinţa pentru proiectul actual este aceea de a folosi pe cât posibil aceste tehnologii.

Conform glosarului de termeni stiinţifici, **scalabilitatea** este însuşirea unui sistem, echipament sau dispozitiv software de a permite schimbări importante ale dimensiunilor sau capacităţiilor sale, cu costuri acceptabile, fără dificultăţi şi cu păstrarea caracteristiciilor şi a performanţelor iniţiale.O buna diviziune a claselor si o modularizare cât mai bună a proiectului îi poate conferii acestuia o scalabilitate ridicată.

Securitatea reprezintă caracteristica prin care un client are acces doar la datele la care ii permite developer-ul şi nu poate accesa alte date. În cazul aplicaţiilor web, securitatea este cea mai importantă caracteristică. Aplicaţia mea trebuie să respecte atât securitatea datelor din micro-blogging cât şi securitatea codului din filtru de mesaje.

În ingineria software, **performanţa** se caracterizează ca fiind capacitatea sistemului de a răspunde repede si stabil cerinţelor utilizatorului. Având in vedere faptul că proiectul are la bază un algoritm, timpul de răspuns si stabilitatea algoritmului este foarte importantă. Un timp de răspuns lent va face algoritmul inutil si ineficient pe când un timp de răspuns rapid însă greşit va anula întreaga funcţionalitate a proiectului.

Uşurinţa de utilizare se refera la capacitatea unei aplicaţii de a fi înţeleasă şi folosită cu uşurinţă de către un utilizator. Având in vedere faptul că funcţionalităţile aplicaţiei sunt puţine,deoarece ideea de bază a proiectului constă in algoritmul folosit, dar şi faptul că interfaţa grafică este una prietenoasă, uşurinţa de utilizare a clientului este una foarte mare.

2.2.3. Cerinţe de implementare

Cerinţele de implementare se referã la acele tehnologii si limbaje de programare care vor trebui folosite pentru dezvoltarea aplicaţiei.Având in vedere faptul cã aplicaţia mea este un site web, aceasta trebuie sã conţinã atât parte de front-end cât si parte de back-end.

Când discutăm despre front-end , vorbim despre acea parte a site-ului sau a aplicației web, pe care o putem vedea și cu care interacționează vizitatorii.. Pentru afisarea paginii vom folosii pagini jsf, pentru stilizare se vor folosii css-uri iar pentru operaţiile asincrone se va folosii javascript.

Rolul back-endului este cel de management de conținut. Ca şi limbaj de bazã al aplicaţie în care va fi scrisã intreaga logicã a aplicaţiei se va folosii limbajul Java.Pentru partea de bazã de date se va folosi sistemul de gestiune de baze de date MySql.

Ca şi platforme pentru pentru dezvoltarea aplicaţiei se va folosi NetBeans 7.4 iar pentru accesul la baza de date s-a instalat programul XAMPP impreunã cu extensia sa de baze de date, PhpMyAdmin. Pentru rularea micro-blogging-ului se va folosii platforma Eclipse(programul a fost creat pe aceastã platformã).

Alte cerinţe de implementare suplimentare ar fi:documentarea şi comentarea codului(pentru a fi mai uşor de inţeles şi modificat de diferite persoane), testarea manualã si automatã(pentru a ne asigura de faptul cã programul funcţioneazã bine si cu parametrii de performanţã acceptaţi),realizarea unei aplicaţii modulare care sã poatã fi uşor integratã cu alte sisteme.

2.3. Conţinutul lucrării

În cele ce urmează se vor dezvolta şi se va explica in mod mai clar toate cele 8 secţiuni evidenţiate, lucrarea fiind impărţitã in 8 capitole:

- În capitolul I, Introducere – se va specifica contextul în care a fost creat proiectul nostru dar şi diferitele provocări pe care le vom întâmpina în dezvoltarea programului.
- În capitolul II, Obiectivele proiectului s-au descris Specificaţiile si obiectivele aplicaţiei împreunã cu cerinţele funcţionale, non-funcţionale şi de implementare ale aplicaţiei.
- În capitolul III, Studiu bibliografic se va face un studiu bibliografic cu referire la aspectele teoretice folosite în lucrarea de faţã.
- În capitolul IV, Analizã şi fundamentare teoreticã, se descriu algoritmii şi noţiunile teoretice utilizate în aplicaţie. Scopul acestui capitol este de a ne explica principiile funcţionale ale aplicaţiei implementate. La sfârşitul acestui capitol se prezintã soluţia propusã.
- În capitolul V, Proiectare de detailiu şi implementare, se descriu pe larg modulele sistemului de construire a ierarhiilor, funcţionarea lor şi modul în care au fost implementaţi diverşi algoritmi prezentaţi în capitolul anterior.
- În capitolul VI, Testare şi validare se prezintã metodele de testare folosite pentru validarea aplicaţiei.
- În capitolul VII, Manual de instalare şi utlizare, sunt detaliate resursele software şi hardware necesare pentru instalarea şi rularea aplicaţiei, precum şi

o descriere pas cu pas a procesului de instalare. Se descrie şi modul de utilizare al aplicaţiei pas cu pas, din punctul de vedere al utilizatorului.

- În capitolul VIII, Concluzii se prezintã un rezumat al contribuţilor personale, o analizã criticã a rezultatelor obţinute şi o descriere a posibilelor dezvoltări şi îmbunãţiri ulterioare.

Capitolul 3. Studiu Bibliografic

Lucrarea noastrã conţine douã direcţii importante.Prima direcţie se referã la clasificarea si filtrarea textelor. În aceasta parte se va discuta in principal despre filtrarea textelor. În cea de-a doua parte se va discuta despre conceptul de micro-blogging şi se va analiza modul in care micro-bloggingul realizează filtrarea de mesaje, despre modul cum micro-bloggingul local a fost creat şi despre cum se aseamãnã cu alte micro-blogginguri.

3.1. Clasificarea şi filtrarea textelor folosind algoritmi clasici

În literatura de specialitate se gãsesc douã tipuri de sortãri de mesaje: Sortarea dupã conţinutului mesajului si filtrarea dupã componentele sociale.

3.1.1 SIFT

Sortarea dupã conţinutul mesajului este cea mai buna modalitate de sortare. În acest tip de sortare fiecare utilizator se presupune a lucra independent unul faţã de celãlalt. Ca şi rezultat sortarea dupã conţinutul mesajului poate exploata doar documentele.

Programatorul Yan a implementat o filtrare bazatã pe conţinut pentru prima datã pentru un program de ştiri, intr-un sistem intitulat SIFT[3]. Profilele pentru SIFT au fost construite manual prin specificarea cuvintelor care sunt de preferat sã fie adãugate sau ignorate şi trebuia sã fie updatat manual dacã utilizatorul dorea sã le schimbe. Pentru fiecare format, 20 de articole erau disponibile in fiecare zi.Articolele puteau fi alese selectiv folosind World Wide Web.

SIFT oferea douã facilitãţi pentru a ajuta utilizatorul cu crearea profilelor.Yan a dezvoltat SIFT pentru a studia eficient algoritmii pentru filtrarea informaţiilor.În SIFT colecţii mari de informaţii erau comparate cu fiecare articol care sosea în Internet News printr-un server central. Eficienţa era obţinutã prin gruparea profilelor în grupuri în aşa fel încât filtrarea se se poatã face pe grupuri. SIFT nu fãcea deosebire intre locurile în care apãreau cuvintele aşa cã, cuvintele care apãreau in titlu, în corpul mesajului sau în semnãturã aveau de obicei aceeaşi pondere.

3.1.2 Fast Data Finder

Câteva sisteme de filtrarea de texte comerciale se bazeaza pe construcţia manualã a profilelor. Sistemul „Fast Data Finder”[3], un produs al Parcel Inc, foloseşte mii de unitãţi de procesare care sunt optimizate pentru operaţii cum ar fi ponderea termeniilor, constrângerile de proximitate etc. Fiecare unitate de procesare este programatã pentru un singur task iar grupuri de unitãţi de procesare sunt create pentru fiecare profil în parte. Cãutãri simultane pe multiple profile sunt suportate aşadar un un profil multilingual poate fi implementat ca şi un set de profile unilinguale, unul pentru fiecare limbã. Tool-uri automate au fost create pentru a ajuta utilizatorul sã-şi creeze profiluri. Spre deosebire însã de SIFT, algoritmul nu este dinamic, aşa cã profilele de Fast Data Finder nu se adaptează automat.

3.1.3 Info Scope

Tehniciile de filtrare a textelor în mod adaptiv cautã sã facã utilizatorul sã caracterizeze un profil din documentele anterioare şi sã foloseascã acel profil pentru a stabili o bazã de reper pentru noile documente sosite. Stevens a dezvoltat un sistem intitulat „Info Scope" [3] care se baza pe aceastã abordare. Info Scope implementa filtrare adaptivã de text, sugera reguli de funcţie pentru observarea comportamentului utilizatorului şi le oferea pentru acceptare sau uneori modificare. Aceste sugestii aveau la bazã acţiuni simple, observabile cum ar fi timpul petrecut de utilizator pentru a citi stirile sau dacã un document citit a fost salvat pentru referinţe ulterioare.Prin evitarea cerinţelor pentru feedback-ul utilizatoriilor despre articole individuale Info-Scope a fost creat pentru a minimiza încãrcarea de informaţii pe server.

Dacã SIFT trata ştiriile ca fiind o colecţie monoliticã de articole, Info-Scope face filtrãri clare între grupuri, subiecte şi chiar şi persoane individuale. Stevens a construit în acest scop nivele de abstactizare astfel incât acestea sa fie importante pentru utilizator. InfoScope a implementat aceastã idee pentru a dezintegra grupuri de ştiri şi a forma grupuri virtuale individuale. Pentru a defini ştiri pentru grupuri virtuale cu posibilitate de a suprascrie sursele, utilizatorii pot reorganiza informaţia in acordanţã cu modelul personal cognitiv al pãrţilor care participã la comunicare pe care ei doresc sã le observe.

InfoScope nu a fost insã fãrã limitări. Sistemul experimental pe care Stevens l-a dezvoltat a fost capabil sã procese informaţiile doar din header-ul fiecãrui articol.Astfel potenţialul algoritmului de a explora texte mari era limitat aşa cã a fost nevoie de folosirea unei metode care sa aleagã doar cuvintele exacte din headere.

Datorită costului lor redus, disponibilitatea unui volum mare de mesaje, și ușurința de a recunoaște informații noi, Internet News și poștă electronică au fost domenii populare pentru cercetarea informațiilor de filtrare. Din păcate, aceste domenii sunt slab potrivite pentru experimente formale, deoarece rezultatele reproductibile sunt greu de obținut. Din acest motiv, foarte puține se știe despre eficacitatea SIFT-ului sau InfoScope-ului. Stevens a raportat faptul că opt din zece cititori experimentați de Internet News au preferat InfoScope la software-ul lor principal într-un studiu inițial, și că toţi cei cinci utilizatori din a doua evaluare au raportat că mai puține articole neinteresante au fost prezentate în prima săptamâna și mai multe articole interesante au fost citite în a 10-a săptămână decât în prima. Deoarece SIFT a fost dezvoltat pentru a studia eficienta, mai degrabă decât probleme de eficacitate, cu atât mai puțin sunt disponibile informații cu privire la eficacitatea acestuia.

Învăţând mai multe despre eficiența unei tehnici de filtrare a textului prevede ca tehnica să fie evaluate în condiții experimentale controlate. Și pentru că performanța tehnicilor de filtrare de text poate varia semnificativ atunci când are nevoie de informații diferite și sunt folosite colecții de documente, compararea rezultatelor între sisteme este facilitată atunci când acești factori sunt consideraţi constante. Evaluarea de rutare TREC a oferit o întâlnire fără precedent pentru acest tip de evaluare a performanțelor. Realizat anual, incepand din 1992, cea mai recentã evaluare de rutare (TREC-5) a atras participarea de la 14 grupuri de cercetare.

3.1.4 TREC NIST

NIST [3] oferă fiecărui participant cincizeci de subiecte şi un set mare (de obicei sute) de documente de formare şi evaluări relevante pentru fiecare subiect. Participanții îşi dezvoltã sistemele lor de filtrare a textului, folosind aceste date ca şi cum ar reprezenta un feedback explicit cu privire la utilitatea fiecărui document de formare dat de cãtre un utilizator, iar apoi trebuie să îşi înregistreze profilurile lor cu NIST înainte de a primi documentele de evaluare. Profilele sunt apoi utilizate de către sistemele de filtrare a textului care le-au generat pentru a pondera un set nevăzut în prealabil de documente de evaluare, iar primele câteva mii de documente se vor depune la NIST pentru evaluare.

Pentru a obține rezultate reproductibile, este necesar să se facă unele ipoteze foarte puternice cu privire la natura sarcinii de filtrare de informații. În TREC se presupune că judecățile umane cu privire la satisfacerea unei informații de cãtre un document sunt binar evaluate (de exemplu, un document este relevant pentru o informare sau nu este) şi constant,(nu contează cine face această hotărâre sau atunci când se face). Relevanța, conceptul fundamental pe care se bazează această metodologie, de fapt, nu îndeplinește simultan aceste două ipoteze. Judecăta de relevanță umanã prezintă variabilitate semnificativã între mai mulţi evaluatori, dar si variabilitate pentru acelaşi evaluator dea lungul timpului. Mai mult decât atât, evaluatorii găsesc uneori cã este dificil sã facã o judecată de relevanță binarã pe o combinație specifică a unui document şi o nevoie de informare. Cu toate acestea, măsurătorile de performanță bazate pe un set comun de hotărâri oferă o bază de principiu pentru a compara performanța relativă a diferitelor tehnici de filtrare a textului pe bază de conținut.

Evaluarea Trec se bazează pe măsuri de eficacitate, care sunt frecvent utilizate pentru sistemele de filtrare a textului. Eficiența sistemelor de filtrare a textului este de obicei caracterizată prin trei statistici:"Precizie", "Reapelare" şi "Eşec". Precizia este fracțiunea de detectat (şi astfel relevante) din documentele care sunt de fapt relevante pentru utilizator, în timp ce reapelarea este fracțiunea din setul actual de documente relevante care sunt clasificate în mod corect ca fiind relevante de către sistemul de filtrare a textului. Atunci când sunt utilizaţi împreună, precizie şi reapelarea definesc eficacitatea de detecţie. Eşecul (fracțiunea de documente non-relevante care sunt clasificate de către sistemul de drept potențial relevant) este folosită pentru a măsura eficiența respingerii.

În TREC, aproape toate sistemele de filtrare a textului produc rezultate in funcţie de pondere. În consecință, precizia şi eşecul de la mai multe valori de rechemare sunt raportate, iar "precizia medie" (aria de sub curba de precizie-eşec) este raportatã pentru utilizare atunci când este nevoie de o singură măsură de eficacitate. Precizia medie se calculează prin alegerea de seturi succesive de documente cu pondere mare si care au un grad mare de reapelare. Precizie medie se calculeaza prin alegerea de seturi succesiv mai mari de documente din partea de sus a listei pe locul care duce la o reapelare din ce în ce mai mare. Precizia este apoi calculatã pentru fiecare set, o tehnică de interpolare este utilizată pentru a estima precizia între punctele de rechemare observate, iar aria de sub curba interpolată este raportată ca precizia medie pentru o informație individualã. Procesul se repetă pentru mai multe informaţii, iar media valorilor obținute în acest mod este raportat ca precizia medie pentru sistemul testat. Deoarece atât precizia cât şi reapelarea variază între zero şi unu, valoriile mai mari de precizie medie sunt considerate a fi mai bune , valoarea ideală fiind în mod logic unu.

3.2 Clasificatori de tip machine learning

În ultimii ani,managementul documentelor in funcţie de conţinut a luat o amploare deosebitã datoritã creşterii disponibilitãţii documentelor in formã digitalã dar şi nevoii de a le accesa in moduri flexibile. Categorisirea de text(sau clasificarea de text), activitatea de a asocia un document cu o categorie tematicã provenitã dintr-un anumit set este unul dintre aceste task-uri.Un algoritm de tip machine learning presupune faptul cã, in momentul in care are loc o filtrare de mesaje, sistemul „învaţã" anumite stereotipuri astfel incât la urmatoarea filtrare algoritmul sã fie mai bun.

3.2.1 Clasificatori de tip arbori de decizie

Un clasificator de tip arbori de decizie [4] este de fapt o reprezentare simbolicã a unui copac in care nodurile interne sunt termenii, ramurile care pleacã de la nodurile interne sunt ponderile pe care acei termeni îi au în document iar frunzele sunt categorile. Un astfel de clasificator categorizeazã un document de test prin testare recursivã a ponderiilor din nodurile interne si adãugarea lor intr-un vector pânã in momentul în care o ajungem la o frunzã.

Existã un numar foarte mare de pachete standard pentru folosirea clasificatoriilor de tip decision-tree. Printre cele mai populare putem amintii ID3(folosit in Fuhr 1991), C4.5 (folosit în Cohen and Hirsh 1998) şi C5 (folosit în Li si Jain 1998).

O metodã posibilã a algoritmului de învaţare pentru un arbore de decizie si o categorie constã in folosirea strategiei „divide and conquer".Strategia verificã dacã toate exemplele au aceeaşi categorie; dacã nu au se va selecta un termen şi se va diviza arborele nostru in funcţie de termenul ales. Procesul se va repeta recursiv şi pe sub-arbori pânã când fiecare frunzã va fi asignatã unei anume categorii. Pasul cheie al algoritmului este alegerea unui termen potrivit dupã care sã se facã subdivizarea, aceastã alegere fãcundu-se pe baza unui cumul de informaţii sau unui criteriu entropic.Majoritatea algoritmilor care au la bazã arbori de decizie includ metode pentru creşterea mãrimii arboriilor(şi reducerea complexitãţii lor) dar şi pentru tãierea ramurilor inutile(pruning algoritm).

Arborii de decizie au fost folosiţi de-a lungul timpului fie ca algoritm de clasificare primarã, fie ca şi clasificare secundarã sau ca şi membrii ai unei clasificãri.

3.2.2 Clasificatori de tip reguli de decizie

Un clasificator de tip reguli de decizie[4] reprezintã un clasificator creat prin metode din forma normalã disjunctivã. Forma normalã disjunctivã a unei mesaj reprezintã o modalitate de reprezentare a mesajului care foloseşte doar simboluri denumiţi literari.Literarii din premise(mesajele iniţiale) denotã fie prezenţa fie absenţa cuvântului de bazã în document pe când clauzele de decizie denotã dacã documentul poate aparţine categoriei sau nu. Reguliile DNF(forma normalã disjunctã) sunt similare cu cele din arborii de decizie prin faptul cã pot codifica orice funcţie booleanã. Pe de altã parte un avantaj al folosirii DNF l-ar putea reprezenta faptul cã acesta din urmã tinde sa genereze o clasificare mult mai compactã.

Metodele bazate pe reguli tind sã aleagã din toate reguliile pe cele mai bune conform anumitor criterii de minimalitate. În timp ce arborii de decizie au fost construiţi printr-o strategie de tip dop-down sau divide-and-conquer reguliile DNF au fost construite printr-o strategie buttom-up. Iniţial fiecare exemplu este vãzut ca şi o serie de clauze.Acest set de clauze este deja asignat categoriei, însã are un scor redus. Utilizatorul va aplica o serie de procese de generalizare în care regula va fi simplificatã printr-o serie de modificãri(îndepãrtarea premiselor din clauze sau unirea clauzelor), care vor maximiza compactivitate şi nu vor micşora acoperirea clasificatorului. La sfârşitul clasificãrii, un algoritm de tãiere a clauzelor neimportante este aplicat, în cele din urmã rãmânând doar clauzele importante. În acest moment, în funcţie de ponderea obţinutã, putem sã ne dãm seama dacã algoritmul aparţine metodei respective sau nu.

Reguliile DNF variazã larg in ceea ce priveşte formele lor de apariţie(metode, eurisitici şi criterii). Printre reguliile DNF care au fost aplicate la filtrarea de mesaje amintim CHARADE (Moulinier and Ganascia 1996), DL-ESC (Li and Yamanishi 199), RIPPER (Cohen 1995), Scar (Moulinier 1996) si SWAP(Apte 1994).

Metodele de mai sus folosesc logica propoziţionalã insã au fost folosiţi algoritmi in care s-a folosit şi tehinca First-Order-Logic.Cohen a testat comparativ DNF şi FOL şi a ajuns la concluzia cã First-Order-Logic este mai limitat în eficacitate decât DNS.

3.1.1. Reţele neuronale

Un sistem bazat pe reţele neuronale[4] este un sistem care este compus în principal din unitãţi, fiecare unitate de intrare reprezentând un termen, fiecare unitate de ieşire reprezentând categoria de interes iar legãturile reprezintã ponderiile.Pentru a clasifica un document, se vor afişa pe intrãri termenii, se va rula algoritmul, iar pe ieşiri se vor putea vizualiza categoriile. O metodã tipicã în reţelele neuronale este propagarea înspre înapoi. Astfel, termenii vor fi introduşi pe intrare şi dacã o eroare de clasificare are loc, reţelele care conţin aceastã eroare se vor reiniţializa.Prin aceasta modalitate , vom avea costuri de recuperare mici(nu se vor înlocuii decât reţelele afectate) şi securitate sporitã(datoritã apariţiei metodei de recuperare din eroare).

Cel mai simplu tip de clasificare bazat pe reţele neuronalã este „preceptron”. Acesta este un algoritm de clasificare liniar. Alte tipuri de clasificatori neuronali implementeazã o formã de logisticã regresivã şi sunt doar în stadiul de propunere.

Un algoritm non-linear bazat pe reţele neuronale este în schimb o reţea cu unul sau mai multe straturi de unitãţi pe care algoritmul de clasificare îl reprezintã ca şi interacţiuni de tip înalt pe care reţeaua trebuie sã fie capabilã sã-l înveţe.Totuşi comparând algoritmul non-linear cu cel linear s-a ajuns la concluzia cã foarte puţine îmbunãtaţiri au fost aduse.

3.1.2. Asamblorul de clasificatori

Asamblorul de clasificatori[4] este bazat pe ideea cã, în momentul în care avem un task care necesitã expertizã de specialitate pentru a putea fi rezolvat, mai mulţi experţi ar putea fi mai utili în rezolvarea lui decât unul singur. Translatând aceastã idee în filtrarea de mesaje rezultã faptul cã, aplicând o serie de clasificatori asupra aceluiaşi text,

posibilitatea ca textul sa fie clasificat corect este mai mare.Astfel asamblorul de clasificatori este caracterizat de o serie de clasificatori şi un algoritm de decizie.

Problema principalã a acestui algoritm este aceea cã, pentru a asigura eficacitate bunã, clasificatorii ar trebuii sã fie independenţi unul de celãlalt. Aceştia ar putea sã difere atât ca şi conţinut cât şi ca modalitate. Astfel, folosind mai mulţi algoritmi de filtare timpul de execuţie va creşte.

3.2 Micro-blogging

Termenul de blog este o prescurtare a termenului "webblog" şi se referã la un jurnal creat online unde articolele sunt afişate in ordine invers cronologicã, ultimul scris fiind primul care apare. În prezent în lume existã peste 3 milioane de blog-uri având diferite teme.

3.2.1. Microblogging-ul local

Microblogging-ul local este unul de tip open-source. Acesta a fost simplificat deorece tema mea îsi propune sã punã un mai mare accent pe filtrarea de mesaje şi nu pe design-ul şi implementarea microblogging-ului. Totuşi acesta se incadreazã perfect in categoria microblogging-ului şi de aceea am sa-l prezint ca şi un microblog în comparaţie cu celelalte microblogging-uri. Principalul obiectiv a microblogging-ului personal este acela de a permite persoanelor atât postarea cât şi citirea anumitor mesaje.

Prima paginã a micro-blogging-ului este pagina de logare. Aceastã paginã este una asemanãtoare celorlalte pagini de logare obişnuitã de pe alte aplicaţii. Pe lângã logare, utilizatorul va mai putea sã-şi creeze cont pe aplicaţie. Pentru a putea posta mesaje sau pentru a putea citii mesajele celorlalţi utilizatore, utilizatorul nostru trebuie în primul rând sã se logheze. Vizualizarea mesajelor nu este permisã farã logare.

Pentru înregistrarea contului, utilizatorul are nevoie doar de aceleaşi date ca şi pentru logare adica username şi parolã. Aplicaţia a fost importatã doar pentru postarea de mesaje şi de aceea complexitatea sa nu este foarte importantã.

Odatã logaţi în sistemul nostru avem 4 posibilitãţi:Crearea mesajului, gãsirea unui utilizator, vizualizarea postãriilor unui utilizator(follow) oricând acesta le va modifica şi vizualizarea pe moment a postãrilor. Pentru creearea unei postãri avem la dispoziţie 500 de caractere. Restricţia a fost adãugatã deoarece aplicaţia face parte din categoria micro-bloggingului iar micro-bloggingul nu permite decât mesaje scurte. De asemenea în aceastã paginã se pot vizualiza şi mesajele tuturor utilizatorilor care sunt „urmãriţi" de cãtre noi.

În pagina care se ocupã cu gãsirea unui utilizator avem un câmp care va trebuii completat cu numele utilizatorului. Dacã utilizatorul va fi gãsit în sistemul de micro-blogging, utilizatorul nou gãsit va apãrea într-un tabel iar sistemul îi va da posibilitatea utilizatorului curent sã urmãreascã postãrile introduse de cãtre acel utilizator.

În pagina în care putem vedea postãrile utilizatorului, avem, la fel ca şi în pagina anterioarã un input în care putem introduce un nume. Dupã ce utilizatorul a fost gãsit ne vor apãrea în paginã postãriile sale. Spre deosebire de prima paginã, în care ne apar toate postãriile utilizatorilor urmãriţi, în aceastã paginã ne apar doar postãrile utilizatorului

respectiv. Utilizatorul va putea fi schimbat oricând in câmpul de cãutare în tabel urmând sã ne aparã postãrile celuilalt utilizator.

Din punct de vedere tehnic microblogging-ul este o aplicaţie aparent simplã însã cu o complexitatea şi un numãr al claselor destul de mare, clasele fiind special create pentru a îndeplinii funcţia de securitate.A fost creat în limbajul Java şi are la bazã tehnologia Spring. Aplicaţia se mapeaza la 4 tabele în baza de date:authorities, follower, user şi post.

Tabela post este cea mai importantã pentru sistemul nostru de filtrare deoarece din aceastã tabelã se vor extrage mesajele dorite. Aceasta conţine câmpurile de username, password şi enabled. De notificat ar fi aici şi faptul cã parola este salvatã în baza de date sub formã criptatã.În continuare se vor prezenta câteva servicii de micro-blogging şi se vor compara cu blogul local.

3.2.2. Microblogging-uri uzuale

Termenul de microblogging nu se referã la altceva decât un blog în care conţinutul este mai mic atât in ceea ce priveşte mesajele cât şi in ceea ce priveşte dimensiunea fişierelor trimise. Pe scurt, în interiorul unui microblog, utilizatorii trimit unul altuia mesaje scurte, poze individuale sau link-uri cãtre alte site-uri web. În ultimii ani, micro-bloggingul a avut o creştere considerabilã comparativ cu pãrintele sau mai mare blog-ul, in principal datorita simplitãţiilor sale. Printre microblogging-urile mai importante putem amintii: Twitter, Binify, Status Net, Jaiku, Qaiku.

3.2.2.1. Twitter

Twitter[10] este cel mai important serviciu de micro-blogging.Acesta permite utilizatorilor sã inregistreze mesaje care au dimensiunea de maxim 140 de caractere.Microblogging-ul personal pe de altã parte permite utilizarea a 500 de caractere. Twitter este un microblogging multilingval, adicã opţiuniile si conţinutul acestuia poate apãrea în mai multe limbi. Postãriile insã nu vor fi modificate ele putând apãrea in limba care au fost create şi sub modalitatea in care au fost create. Micro-bloggingul personal foloşte doar engleza ca şi limbã de bazã.

O altã caracteristicã a Twitter-ului care poate fi utilizatã în compararea acestuia cu micro-bloggingul personal se referã la conceptul de “tagging”. Un tag este un cuvânt cheie sau un termen asignat unei informaţii care ne poate duce cãtre pagina de unde aceea informţie provine.În Twitter putem da tag unei postãri de Twitter sau oricãrei alte pagini Web. Microblogging-ul personal, fiind o creaţie mult mai simplistã nu permite taging-ul sub nici o formã.

Micro-bloggingurile au de obicei douã tipuri de conversaţii. Un tip de conversaţie este acela public, in care oricine poate vedea mesajele care au fost schimbate intre oameni.Un alt tip de conversaţie poate fi cel privat in care doar persoanele care participã la conversaţie pot vedea mesajele. În ceea ce priveşte Twitter-ul acesta permite atât conversaţii private cât şi conversaţii publice.Microblogging-ul personal permite doar acele tipuri de conversaţii care sunt publice, mesajele postate de cineva putând fi vãzute de oricine urmãreşte aceea persoanã.

Aplicatiile web pot avea servicii de tip open source. Prin intermediul serviciilor de tip open-source putem accesa date din baza de date a acelor aplicaţii. Microblogging-urile find aplicaţii web au posibilitatea de a acorda anumite servicii de tip open source. Atât Twitter-ul cât şi microblogging-ul personal oferã utilizatorilor acces la toate mesajele postate.

Micro-bloggingurile pot avea o serie de funcţionalitãti suplimentare, insã principala lor funcţionalitatea ar fi accea de a posta si de a citi mesaje. Postarea si citirea mesajelor presupune operaţii diferite si pot fi fãcute prin mai multe modalitãţi.În cazul twitter-ului, putem posta mesaje prin web, sms sau api.Postarea mesajelor prin web presupune accesarea paginii web unde vom scrie şi posta mesajul .Postarea prin sms a fost creatã pentru a spori flexibilitatea microblogging-ului şi constã în scrierea unui mesaj pe telefon şi respectarea anumitor indicaţii astfel încât mesajul dorit sã poatã sã ajungã în locul dorit de noi.Postarea prin API este foarte asemãnãtoare cu postarea prin web, insã în loc sã deschidem o paginã web vom accesa o aplicaţie desktop special creatã de Twitter care ne va permite sa scriem mesajele pe pagina doritã. Citirea mesajelor poate fi facutã prin web, sms ,api si feed. Un web feed sau un news feed este un format de date utilizat pentru a oferi utilizatorilor conţinutul actualizat frecvent.În cazul microblogging-ului personal citirea si postarea mesajelor se poate face doar prin web.

3.2.2.2. StatusNet

StatusNet[11] este cel mai complex microblogging existent pe piaţã.Deşii este foarte asemãnãtor cu Twitter si are aproximativ aceleaşi funcţionalitãţi, StatusNet asigurã utilizarea unor dezvoltãri noi.

O primã caracteristicã care ar putea fi folositã in comparaţia cu microblogging-ul personal ar fi faptul cã StatusNet asociazã utilizatorii în diferite grupuri. Grupurile pot fi formate dupã diferite criteri, utilizatorii putând aparţine mai multor grupuri iar un grup poate cuprinde mai mulţi utilizatori.Microblogging-ul personal, spre deosebire de StatusNet nu permite aceasta dezvoltare, fiecare utilizator fiind cosiderat ca şi persoanã diferitã.

RSS este una din dezvoltãrile noi adoptate de cãtre StatusNet. RSS provine de la Rich Site Summary si este un protocol prin care se pot trimite informaţii complexe(imagini, audio) şi care are ca şi scop principal transformarea informaţiei intr-una publicã, informaţia putând fi accesatã din oricare altã paginã web. Microblogging-ul personal nu foloseşte protocolul rss insã oferã accesibilitate la informaţie.

FOAF sau Friend of Friend este o ontologie folositã in special in cãutarea utilizatorilor. Astfel algoritmul incepe de la utilizatorul actual si cautã recursiv prin toţi "prietenii" acestuia pânã când utilizatorul dorit va fi gãsit.Microblogging-ul personal nu permite acest tip de cãutare insã ar putea fi implementatã ca şi o dezvoltare ulterioarã.

Bookmarklet-ul este un bookmark pãstrat in browser care conţine comenzi Javascript pentru a extinde functionalitatea browser-ului.StatusNet foloseşte acest tip de dezvoltare nouã in special pentru a creşte distributivitatea aplicaţiei.Microblogging-ul local, are secvenţe de Javascript integrate direct in cod si nu foloşte bookmarklet-ul.

În ceea ce priveşte postarea şi scrierea de mesaje, StatusNet este superior atât microblogging-ului local cât si Twiter-ului. Postarea mesajelor se poate face prin web, xmpp, sms,mail sau api.Postarea prin xmpp este o dezvoltare nouã a StatusNet. Xmpp

este un protocol special folosit in comunicare care salvează mesajele intr-un format asemănător XML-urilor. Microblogging-ul local folosește în comunicare tehnica clasică prin care mesajele se salvează ca şi şiruri.Postarea mesajelor prin E-mail este posibilă printr-o mapare dintre email-ul destinaţie şi pagina pe care se va face postarea mesajelor.Citirea se face la fel ca şi scrierea prin toate modalitătiile posibile.În tabelul de mai jos se vor prezenta modalităţile prin care se pot posta şi citi mesaje in cele mai populare servicii de micro-blogging. Primele 4 coloane se referă la postarea mesajelor iar ultimele 4 coloane se referă la citirea mesajelor.

Micro-blogging	WEB	XMPP	EMAIL	SMS	WEB	XMPP	EMAIL	SMS
StatusNet	DA	DA	DA	DA	DA	DA	DA	DA
Jaiku	DA	DA	NU	NU	DA	DA	NU	NU
Qaiku	DA	DA	DA	NU	DA	DA	NU	NU
Twitter	DA	NU	NU	DA	DA	NU	NU	DA

Tabela 3.1 Tipurile de comunicaţii acceptate de micro-blogginguri[12]

În concluzie, studierea caracteristicilor fiecărui sistem de micro-blogging în parte a ajutat la înţelegerea scopului micro-bloggingului. Astfel, în acest moment se ştie motivul pentru care micro-bloggingul a fost creat, şi modalităţile prin care mesajele circulă prin interiorul serviciilor de micro-blogging.

Capitolul 4. Analiza şi fundamentarea teoreticã

Filtrarea postãrilor irelevante în legãturã cu un subiect dat poate fi realizatã prin clasificarea fiecãrei postãri ca fiind relevantã sau irelevantã şi afişare pe ecran. Pentru aceasta s-a propus o metodã pentru a identifica o metodã de clasificare care sa filtreze mesajele.

Clasificarea are la bazã o serie de module de transformare şi decizie.Pentru fiecare postare vom returna valoriile care au relevanţa cu mesajul scris de noi. Extragerea cuvintelor importante din interiorul mesajului nostru este foarte important pentru algoritm. Din fericire tehnologiile actuale fac acest lucru mult mai uşor decât în trecut.

Vom folosi o serie de module pre-filtrare care au diferite roluri, rolul principal fiind acela cã se va obţine un mesaj simplificat mult mai uşor de manipulat. Urmeazã apoi etapa de filtrare unde se vor filtra mesajele bazate pe conţinut, urmând ca mai apoi, în etapa finalã sa se afişeze acele informaţii care sunt relevante cu mesajul dat de noi.

4.1. Clasificare bazatã pe conţinutul mesajelor

Scopul clasificării este acela de a filtra mesajele care au texte irelevante faţă de un mesaj dat de noi, elementele textuale sunt principalele caracteristici din sistemul nostru. Caracteristicile textuale ar trebui să reprezinte cât de relevant este subiectul textului pentru mesajul nostru. Fiecare mesaj are un subiect diferit (sau subiecte), de aceea avem nevoie de o metodă pentru a extrage în mod automat temele-cheie din mesajul nostru și măsura similaritatea textului cu acesta.

4.1.1. Module de prelucrare pre-filtrare

Pentru a putea filtra textul avem în primul rând nevoie ca acesta să fie modificat. Textele trebuie sa ajungã la algoritmul de filtrare intr-o formã minimã astfel incât , procesãri ulterioare sa numai trebuiascã facute.În funcţie de momentele de apariţie vor fi prezentate modulele de preprocesare.

4.1.1.1.Modulul de extragere a conţinutului HTML

Micro-bloggingul este limitat la 500 de caractere. Acest lucru poate fi considerat ca un impediment deoarece explicaţiile si detaliile mesajelor sunt restrânse.Astfel, mulţi utilizatori apeleazã la opţiunea de a introduce un url care sa conţinã mai multe informaţii cu legaturã la subiect. Modulul de extragere a conţinutului HTML îşi propune sã extragã informaţia utilã din interiorul url-ului şi sa o apendeze textului.

Pentru a putea extrage datele utile dintr-un html, trebuie mai întâi sã verficãm dacã cuvântul respectiv este sau nu este un url.Clienţii nu au în micro-blogging-ul local nici un loc special în care sã poatã insera url-uri de aceea sunt nevoiţi sa le introducã în interiorul mesajului. Pentru a verifica faptul cã avem un url în mesaj vom parcurge mesajul şi vom aplica un algoritm de verificare de url pe fiecare cuvânt.

Modalitatea de verificare dacã un cuvânt este un url sau nu o reprezintã incercarea de conexiune la acel url. Dacã cuvântul nostru este url , rezultatul browser-ului va avea codul 200. Dacã cuvântul nu este url, rezultatul browser-ului v-a avea alt cod. Aceastã metodã este una mai lentã insã este cea mai eficientã. O altã modalitate ar fi putut fi folosirea expresiilor regulate. În aceastã metodã se lua cuvantul şi se verificau anumite tipare(de exemplu dacã cuvântul incepe cu „http” sau dacã conţine unul din domeniile de internet(ro,com,de etc).Metoda cu expresii regulate ar fi fost mult mai rapidã insã ar fi fost şi mult mai ineficientã.Odatã ce ştim cã cuvantul este un url putem parsa informaţiile din el.

Pentru a parsa url-ul şi pentru a scoate informaţiile utile vom folosii un html parser predefinit de java. În informaticã parsarea este parcurgerea si analizarea unui text, cu identificarea atomiilor care îi corespund în raport cu o gramaticã formalã.În cazul nostru, documentul de parsat este întreaga paginã html.

Algoritmul de parsare a unui text HTML nu este foarte complicat.Acesta ia ,într-o primã fazã, întregul text html şi îl considerã ca fiind un text normal.Fiecare algoritm de parsare conţine un limbaj lexical. Limbajul lexical nu este altceva decât totalitatea cuvintelor cheie ale limbajului respectiv(în cazul nostru este vorba despre HTML).

Ce-a de-a doua fazã a algoritmului constã în parcurgerea mesajului caracter cu caracter si verificarea cu limbajul lexical. Astfel dacã in parcurgerea mesajului vom gãsii caracterul „<” acesta nu indicã alteceva decât inceperea unui tag html. În momentul în care se gaseşte un caracter special se va memora poziţia lui. Caracterul „/” ar putea însemna sfârşitul unui tag html doar dacã este precedat de caracterul special „<.

Ce-a de-a treia fazã a algoritmului reprezintã parcurgerea mesajului cuvânt cu cuvânt. In momentul in care a fost gãsit un cuvânt cheie cum ar fi „body” este pãstratã poziţia sa in mesaj.Dacã de exemplu avem acelaşi element de mai multe ori se va pãstra fiecare locaţie a sa

În ultima faza a algoritmului, html-ul nostru este practic tradus intr-un vector în care fiecare element are un anumit simbol şi o anumitã poziţie. Nu ne rãmâne de fãcut decât sa supunem acel vector unor serii de reguli pentru a scoate cuvintele utile din html. Se considerã text util, textul dintre ultimul tag de „>” şi primul tag de „/” care este precedat de untag de „>”. De exemplu dacã avem urmatoarea sintaxã html „<tr><td> Acesta este un mesaj </td></tr>” textul „acesta este un mesaj” va fi considerat ca şi text datoritã regulilor descrise. Bineînţeles acesta este un exemplu în care a fost descrisã o regulã simplã de aflare a conţinutului. În figura 1.1 avem schema modului de extragere de text dintr-o paginã html.

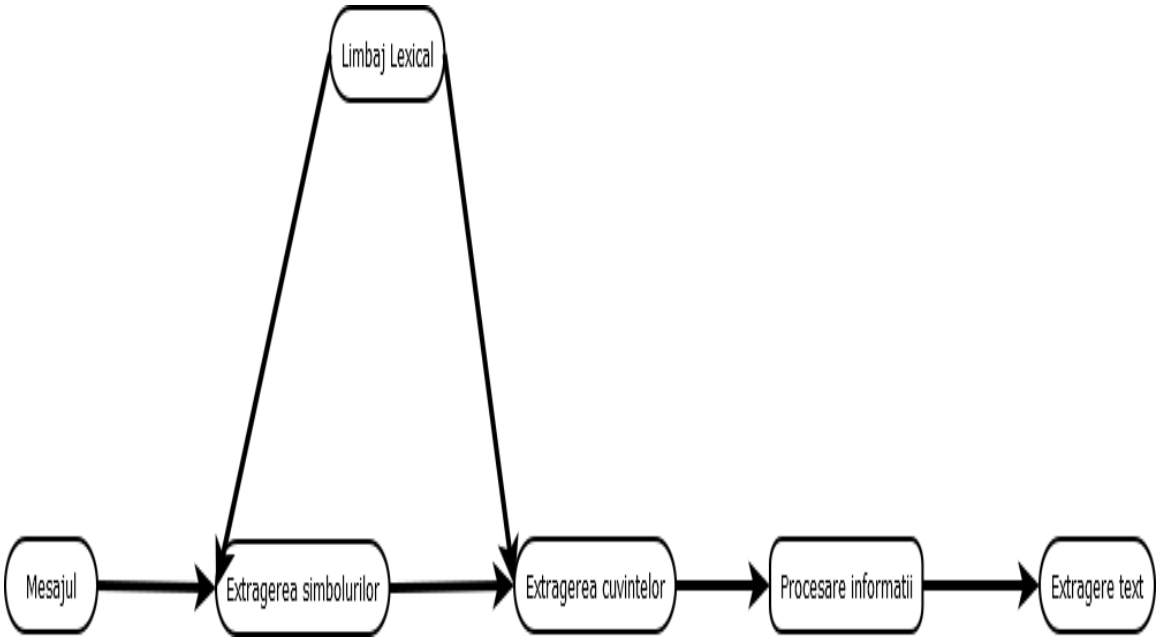

Figura 4.1 Schema modului de extragere de text html

4.1.1.2.Modulul de traducere al textului

Textele scrise în limbi diferite au fost pentru mult timp o restricţie. Multe dintre infomaţii se pierdeau doar pentru cã nu exista acces la ele. În zilele noastre însã, in principal datoritã creşterii distributivitãţii aplicaţiilor, putem folosii texte din alte limbi ca şi cum acestea au fost scrise în limbiile proprii. Ca şi limbã comuna tuturor textelor am ales engleza pentru cã aceasta este documentatã mai bine, sporiind eficienţa algoritmului.

Ca şi în cazul anterior, înainte de a face o traducere avem nevoie de o detecţie a limbii utilizate. În cazul în care am fi avut mesajul nostru în englezã şi noi am fi încercat sã traducem mesajul în englezã, toate atributele de calitate ale proiectului ar fi scãzut considerabil.Pentru traducerea de mesaje am folosit un detector de limbaj. Acesta conţine cele mai utilizate cuvinte dintr-o anumitã limbã si face cãutarea cuvintelor in mod secvenţial.Astfel dacã propoziţia mea este „Vreau sa iau licenţa” acesta va cãuta in baza lui de date cuvintele în funcţie de ordinea de apariţie in propoziţie. Dacã la cuvântul „vreau”, gãseşte ca şi limbã, limba românã se opreşte din cãutat şi returneazã ca şi limbã descoperitã limba românã. Dacã gãseşte mai multe limbi, acesta va traduce urmãtoarele cuvinte pânã când se va ajunge la o detecţie de limbã.Dacã dupã terminarea tuturor cuvintelor din propoziţie nu se va detecta nici o limbã, algoritmul va selecta una din ele dupã anumite criterii de relevanţã.

Algoritmul de traducere de texte este unul foarte complex care se face de către mai multe computere. Aceste computere utilizează un proces numit ‘traducere automată statistică’, acesta fiind de fapt un mod elevat de a spune că traducerile sunt generate de computerele noastre pe baza modelelor găsite în cantităţi mari. Dar să revenim puţin asupra lucrurilor. Dacă aţi dori să învăţaţi pe cineva o limbă nouă, aţi începe prin a-l învăţa cuvinte din vocabular şi reguli gramaticale pentru a explica modul de construire al propoziţiilor. Un computer poate învăţa o limbă străină în acelaşi mod – utilizând un vocabular şi un set de reguli. Dar limbile sunt complicate şi, după cum vă poate spune

orice persoană care învaţă o limbă străină, există excepţii de la aproape orice regulă. Când se încearcă includerea tuturor acestor excepţii şi a excepţiilor de la respectivele excepţii într-un program de computer, calitatea traducerii începe să se deterioreze.

Algoritmul de traducere are o abordare diferită. în loc să încercăm să învăţăm computerele toate regulile unei limbi, lăsăm computerele să descopere singure regulile. Computerele realizează acest lucru analizând milioane şi milioane de documente care au fost deja traduse de traducători reali. Aceste traduseri provin din cărţi, de la organizaţii ca ONU şi de pe site-uri web din întreaga lume. Computerele scanează aceste texte căutând modele importante din punct de vedere statistic, şi modele create între traducere şi textul original cu probabilitate redusă să fi apărut în mod întâmplător. După ce computerul găseşte un model, îl poate utiliza pentru a traduce texte similare pe viitor. După ce acest proces se va fi repetat de miliarde de ori, se vor obţine miliarde de modele şi un program de computer foarte inteligent. Totuşi, pentru unele limbi avem la dispoziţie mai puţine documente traduse şi prin urmare mai puţine modele detectate de aplicaţia noastră software. Din acest motiv, calitatea traducerii variază în funcţie de limbă şi de perechea de limbi. Suntem conştienţi că traducerile nu sunt întotdeauna perfecte, dar furnizând constant noi texte traduse, putem crea programe mai inteligente şi traduceri mai bune.

4.1.1.3. Modulul de eliminare a semnelor de punctuaţie si a cuvintelor uzuale

În momentul în care se scrie un text, majoritatea oameniilor nu sunt atenţi asupra modului in care spaţiile dintre cuvinte şi semnele de punctuaţie sunt puse. De aceea, semnul de punctuaţie poate apãrea înaintea cuvântului sau dupã. În acest sens un algoritm de eliminare a semnelor de punctuaţie este necesar.Algoritmul verificã dacã cuvântul începe sau se terminã cu un semn de punctuaţie. În oricare din aceste situaţii semnul de punctuaţie se va inlãtura.

Dacã dorim sa facem o filtrare de mesaje este ştiut faptul cã, cuvintele de legãturã au o prezenţa mai mare în fiecare propoziţie. Prezenţa lor mare va determina ca ele sã aibã conform algoritmului o pondere foarte mare şi sã încurce foarte mult filtrarea.

Conform studiilor, masa de bazã a unui vocabular este în principal formatã din puţine cuvinte.Pentru a face acest lucru, vom extrage din mesajele noastre cele mai utilizate 250 de cuvinte din limba englezã.Dacã extrãgeam mai multe cuvinte riscam faptul ca, mulţi termeni relevanţi pentru mesajul meu sa nu fie incluşi.Din algoritm au fost excluse pãrţi de vorbire precum prepoziţii, conjucţii, pronume, interjecţii, verbe uzuale(am, has)etc. În tabelul de mai jos ne sunt date toate cuvintele care fac parte din masa vocabularului englez şi au fost excluse din algoritmul meu.

the	be	to	of	and	a	in	that	have	I
it	for	not	on	with	as	you	do	at	this
but	his	by	from	they	we	say	her	she	or
an	will	my	one	all	would	there	their	what	so
up	out	if	about	who	get	which	go	me	when
make	can	like	time	no	just	him	know	take	people
into	year	your	good	some	could	them	see	other	than
then	now	look	only	come	back	after	use	two	how

Tabela 4.2 Cele mai întâlnite 80 de cuvinte din vocablarul englez[5]

4.2. Modulul de comparare a cuvintelor

Modulul de comparare a cuvintelor nu este un modul pre-filtrare ci aparţine algoritmului de filtrare de aceea a fost descris separat.Acest modul se ocupã cu compararea a douã cuvinte şi verificã dacã acestea reprezintã acelaşi lucru.

În filtrarea de mesaje se întâmplã foarte rar sã se gãseascã cuvinte identice. De exemplu cuvinetele “filtering” şi “filter” sunt douã forme ale aceluiaşi verb “to filter”, insã deoarece nu sunt identice nu s-ar considera egale în comparare. Dupã cum putem observa, cuvântul de bazã din cele douã verbe este filter. Aceastã ar fi şi ideea care ar putea combate aceastã problemã, şi anume , folosirea unui algoritm de extragere a bazei cuvântului.

Pentru aceasta s-a instalat pe calculatorul propriu un dicţionar al limbii engleze cu care s-a fãcut legãtura. Acest dicţionar conţine în baza lui de date toate mapãriile dintre cuvinte şi cuvintele lor de bazã. Ideea de a folosii aceastã metodã de comparare consider cã este foarte bunã deoarece cuvântul de bazã poate conţine şi ideea de bazã a semnificaţiei lui. De exemplu substantivul “filter” şi verbul “filtering” au aceeaşi idee de baza(filtrarea).

Oamenii comunicã deseori aceeaşi idee însã în moduri diferite. În acest caz extragerea ideii de bazã nu este suficientã ci este nevoie şi de un analizator de sinonime. De exemplu “people” şi “humans” sunt sinononime insã in momentul comparãrii lor acestea nu vor fi pozitiv comparate niciodatã.

Dicţionarul ne-a fost util şi în momentul acesta deoarece acesta conţine in baza lui de date legãturiile dintre toate sinonimele. Algoritmul de comparare verificã intr-o primã etapã dacã cuvintele sunt egale(„filtering cu filtering”). În ce-a de-a doua etapã se va verifica daca cuvintele de bazã din cele douã cuvinte sunt egale („filter” cu „filtering”) urmând ca în a treia etapã sã se verifice dacã sinonime ale cuvintelor de bazã sunt identice.

Deşii pare riscant algoritmul nostru are ca scop gãsirea oricãror douã cuvinte care au ceva comun între ele. Şansele ca mai multe cuvinte dintr-o propoziţie sã aibã legaturã cu mai multe cuvinte dintr-o altã propoziţie fãrã ca logica propoziţiilor sã aibã ceva comun sunt infime.

De notificat ar mai fi aici şi faptul cã, în momentul în care se eliminã cuvintele de legãturã, se foloseşte algoritmul de comparare între cuvântul respectiv şi cuvântul de eliminat.

4.3. Algoritmul de clasificare

În abordarea tradiţionalã, frecvenţa termeniilor sau inversul frecvenţei termenilor sunt mãsurãtori folosite în clasificarea documentelor. Aceste mãsuratori indicã cât de important este un cuvânt pentru un document într-o anumita sintaxa. În frecvenţa inversã a termeniilor, importanţa unui cuvânt pentru un mesaj este cu atât mai mare cu cât cuvântul este folosit de mai multe ori şi este cu atât mai micã când cuvântul este folosit de mai puţine ori. Similaritãţiile celor douã documente pot fi calculate cu ajutorul similaritãţiior ponderilor cuvintelor din fiecare document.

Inversul frecvenţei termenilor dupã pondere este o modalitate popularã de sortare şi este folositã atât în motoare de cãutare cât şi în diferiţi algoritmi de clasificare de text. Cu ajutorul algoritmului care are la bazã inversul frecvenţei termeniilor, cuvintele care se întamplã foarte des in propoziţie , de exemplu „the” sau „end” vor avea ponderi mai mici.Totuşi în cazul în care textul conţine foarte multe cuvinte de legaturã şi foarte puţine cuvinte, acesta ar putea apãrea ca fiind relevant. De aceea s-a ales ca aceste cuvinte sã se elimine complet din algoritm şi sã ia ponderea 0.

Având in vedere faptul cã aceste metode nu sunt potrivite complet cu clasificarea noastrã vom propune o nouã metodã care va calcula proximitatea unui text faţa de postãriile din micro-blogging.Principala provocare aici ar putea fi care cuvinte importante sunt cu adevãrat importante în contextul de faţa şi care vor fi doar cuvinte de legãturã.Modulul de eliminare a cuvintelor comune prezentat anterior s-a ocupat de aceastã problemã, în mesajele de clasificat rãmânând doar cu cuvinte cheie.

Spre exemplu sã presupunem cã avem o serie de postãri în micro-blogging şi cã avem un mesaj care va sorta mesajele noastre ca fiind relevante sau irelevante cu topicul de discuţie. Sã presupunem de asemenea faptul cã cuvântul MySql va apãrea des in propoziţia noastrã deoarece postarea noastrã se referã la baze de date.În lista de postãri extrase din microblogging vor putea fi sute de mesaje iar dintre acestea doar puţine se vor referi la baze de date. Aşadar frecvenţa apariţiei cuvântului MySql este mai micã deoarece ponderea cuvintelor în care a apãrut cuvântul „MySql” este mai micã. Dacã însã vom extrage un cuvânt comun precum „children” sau „tomorrow”, este de aşteptat ca aceste cuvinte sã aibã o frecvenţã de apariţie mai mare şi sã aparã mai desc decât cuventul MySql. Algoritmul nostru ar trebuii ca sã identifice faptul ca „tomorrow” este un cuvânt comun şi sã-i atribuie o pondere cât mai micã. Cuvintele precum „and” şi alte cuvinte de legãturã nu vor fi luate in considerare.

Prin folosirea ideii de comparare dupã frecvenţa termeniilor, vom putea calcula cât de relevant este cuvântul respectiv pentru mesajul nostru.Vom considera mesajul nostru ca fiind unul local, iar mesajul provenit din server-ul de micro-blogging unul global. În figura de mai jos ne este prezentat modul in care se va calcula ponderea unui cuvânt în mesajul local:

$$ponderea\ localã = \frac{numãrul\ de\ apariţii\ al\ cuvântului}{numãrul\ total\ de\ cuvinte}$$

Figura 4.3 Modul de calcul al mesajului local

unde deîmparţitul va reprezenta numãrul de apariţii a cuvântului în mesajul local iar impãrţitorul va reprezenta suma apariţiilor tuturor cuvintelor în mesajul local.

Relevanţa unui text faţã de o postare nu va fi aşadar altceva decât suma ponderilor din accea postare raportatã la numãrul de cuvinte din postarea respectivã.

$$relevanţa = \frac{totalul\ ponderilor}{numărul\ total\ de\ cuvinte}$$

Figura 4.4 Relevanţa unei postări

Dimensiunea textului pentru fiecare postare este limitatã la 500 de caractere în microblogging-ul local ceea ce face algoritmul de clasificare mult mai greu. Pentru a trece peste aceastã problemã se folosesc link-uri care conţin text. În zilele noastre este un lucru comun sã se foloseascã url-uri pentru o mai bunã documentare a informaţiei prezentate. Conţinututl url-ului va fi extras cu ajutorul unui serviciu de extragere de conţinut HTML şi adãugat conţinutului fiecarei postări.

Prin algoritmul nostru, vom putea sã calculãm relevanţa unui text sau a unei postări. Scorul are limita inferioarã 0 iar limita superioarã 1. Pentru fiecare mesaj adãugat ponderea mesajelor se schimbã, algoritmul trebuind reluat din nou. Mesajele ambigue pot crea cele mai multe erori de clasificare pe când mesajele clare, cu cuvinte de topic specifice, pot face ca sistemul sã dea cele mai bune rezultate. Vom folosi tipul mesajului ca şi un indicator din care sistemul trebuie sa inveţe, în funcţie de acesta dându-se valoarea pragului de relevanţã.

4.4. Pseudocodul algoritmului

În figura 4.5 este prezentatã partea de atribuire de ponderi. Prezentarea este fãcutã în pseudocod şi are rolul de a prezenta mai bine algoritmul folosit.

Iniţial vom avea textul local, venit din interfaţa graficã şi textele globale, textele aparţinând bazei de date a micro-bloggingului.Textul local va fi iniţial despãrţit în cuvinte. Dupã despãrţirea în cuvinte se va verifica fiecare cuvânt în parte. Dacã cuvântul nostru este cuvânt comun se va trece mai departe. Dacã cuvântul nostru a mai fost gãsit în mesaj, atunci se va lua ponderea sa şi se va incrementa cu 1. Dacã cuvântul nostru nu este cuvânt comun şi este folosit pentru prima datã în mesajul nostru, atunci se va insera în lista de ponderi cuvântul nostru cu ponderea 1.

Dupã ce toate cuvintele au fost parcurse, se va lua lista de ponderi şi se va împãrţi fiecare cuvânt la numãrul total de cuvinte. Vom face acest lucru pentru ca ponderea fiecãrui cuvânt sã fie dupã formula descrisã în figura 4.3 .În figura 4.4 va fi prezentatã o monstrã a algoritmului folosit.

```
var textLocal=ui.getTestLocal();
var lista<texte>=database.getTexteFromMicroblogging();

vector cuvinteLocale=textLocal.split();
listaPonderi=null;

for(i in cuvinteLocale)
        if(nuEsteCuvantComun(i))
                if(listaPonderi.get(i)==0)
                        listaPonderi.add(i,1);
                else
                        pondere=listaPonderi.get(i);
                        pondere=pondere+1;
                        listaPonderi.add(i,pondere);
                endif
        endif
endfor

for(i in listaPonderi)
        pondereFinala=listaPonderi[i]/nrCuvinte
endfor
```

Figura 4.5 Pseudocod aplicare ponderi

Dupã ce sunt atribuite ponderile pentru fiecare cuvânt va urma ca toate mesajele din baza de date sã fi parcurse.Fiecare mesaj va fi despãrţit la rândul sãu în cuvinte.Dupã aceea urmeazã ca fiecare cuvânt din lista de ponderi sã se compare cu fiecare cuvânt din mesaj. Dacã cuvintele sunt identice se va incrementa o variabilã cu valoarea ponderii cuvântului.Dupã calcularea tuturor ponderilor, se va împãrţi ponderea rezultatã la numãrul de cuvinte al textului global.Dacã aceastã valoare este mai mare decât un anumit prag atunci mesajul este considerat relevant.În figura 4.6 este prezentat algoritumul de atribuire a ponderilor mesajelor globale şi verificarea relevanţei acestora.

De notificat ar mai putea fi şi faptul cã, modulele de prefiltrare nu au fost introduse în pseudocod.Astfel, se considerã cã textele globale nu conţin url(conţinutul url-ului fiind deja extras) şi sunt în limba englezã.De asemenea egalitatea din pseudocudul nostru se va face cu ajutorul utilitarului Wordnet creat de universitatea din Princeton.Mesajele care vor trece pragul de relevanţã se vor trece într-o listã separatã însã pentru înţelegerea algoritmului s-a adãugat doar un comentariu sugestiv.

În concluzie folosirea pseudocodului a avut rolul de a clarifica modalitatea de funţionare a algoritmului şi punerea în practicã a formulelor utilizate.

```
var textLocal=ui.getTestLocal();
var lista<texte>=database.getTexteFromMicroblogging();

for(i in lista<texte>)
        pondereTextGlobal=0;
        cuvinte textGlobal=i.split();
        for(j in listaPonderi)
                for(k in textGlobal)
                        if(j.cuvant==k)
                                pondereTextGlobal=pondereTextGlobal+j.pondere;
                        endif
                endfor
        endfor
        pondereTextGlobal=pondereTextGlobal/nrCuvinteTextGlobal;
        if(pondereTextGlobal)>valoare de prag
                //mesajul este relevant
        endif
endfor
```

Figura 4.6 Pseudocod algoritm filtrare

4.5. Protocolul HTTP

Hypertext Transfer Protocol sau cunoscut sub numele de HTTP este un protocol la nivel de aplicație ce constituie fundamentul pentru World Wide Web (www). Acesta are la bază protocolul de comunicare TCP/IP folosit pentru a trimite date pe Internet. Aceste date pot fi fișiere HTML, fisiere audio sau video, poze sau pur și simplu text. Specificația HTTP exemplifică modul în care sunt construite și trimise datele clienților către server și modul în care serverele răspund acestor cereri.

Există trei particularități care fac protocolul HTTP să fie simplu, dar și puternic în același timp:

- HTTP este “connectionless”: se referă la faptul că browser-ul clientului inițiază o cerere server-ului după care se deconectează de la server și așteaptă răspunsul. Server-ul procesează cererea, se reconectează cu clientul și îi trimite răspunsul acestuia.
- HTTP este “media independent”: se referă la faptul că prin intermediul protocolului se pot trimite orice fel de formate media atâta timp cât server-ul și clientul știu să proceseze conținutul acestora.
- HTTP este “stateless”: din prima caracteristică menționată mai sus rezultă faptul că atât server-ul cât și clientul au cunoștință unul de celălalt doar pe perioada unei cereri, după furnizarea răspunsului acestia nu mai au nicio legătură.

În Figura 4.7. de mai jos este prezentată arhitectura de bază a tutoror aplicațiilor care se bazează pe protocolul HTTP.

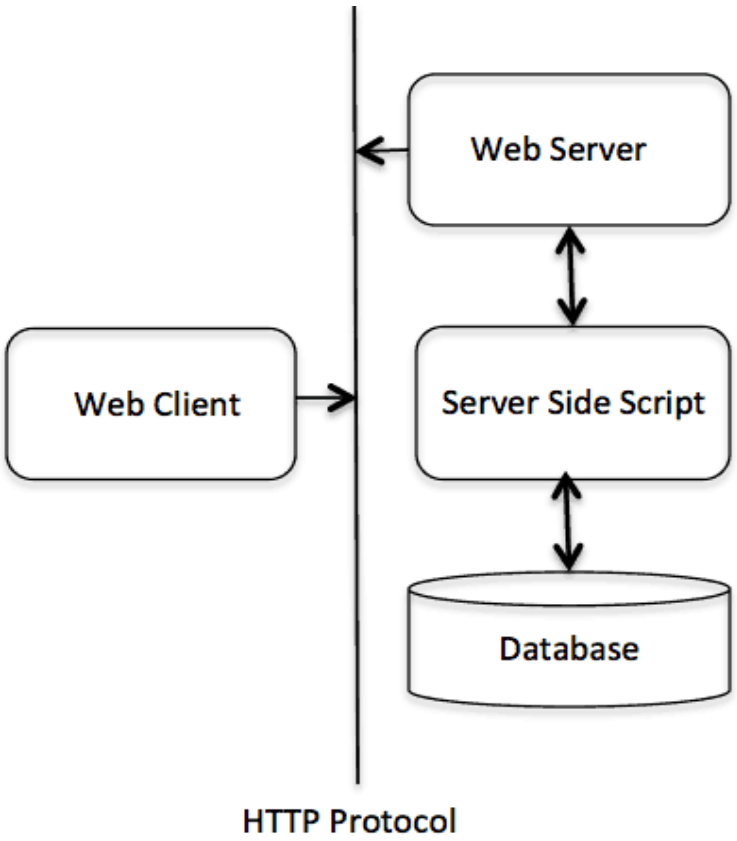

Figura 4.7 Arhitectura aplicațiilor bazate pe HTTP

Clientul trimite o cerere server-ului prin intermediul protocolului HTTP cu toate informațiile de care acesta are nevoie. Server-ul procesează datele, interacționează eventual cu o bază de date după care îi răspunde clientului cu un cod de succes sau de eroare și cu informațiile cerute în caz de succes.

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol sunt descrise specificaţiile aplicaţiei, arhitectura generalã şi diagramele implicate în realizarea proiectului.

5.1. Modul de definire al sistemului

În figura 5.1 este prezentat un curs complet al execuţiei principale aplicaţiei.O primã acţiune în momentul pornirii este aceea de a insera un text care va fi apoi considerat ca şi text de pornire pentru filtrarea de mesaje. De notificat este faptul ca acest fişier poate fi scris doar în englezã, trebuie sa aibã o lungime de minim 15 cuvinte şi nu poate conţine url-uri.

5.1.1. Cursul de execuţie al aplicaţiei

Aplicaţia conţine 2 fâşii de execuţie care sunt independente pânã in momentul clasificãrii, comunicarea dintre ele în acel moment devenind crucialã.

Dupã ce utilizatorul apasã butonul „Search” mesajul de cãutat va avea de suferit o serie de transformãri.Prima dintre ele este aceea cã îi vor fi eliminate semnele de punctuaţie din interiorul mesajului. Mesajul trebuie sã conţinã doar cuvinte. Urmeazã mai apoi partea de traducere. Aici se va verifica dacã mesajul de tradus este în limba englezã. În cazul în care limba detectatã este alta decât engleza se va traduce mesajul în limba englezã. Urmãtorul modul este acela de extragere de cuvinte uzuale. Dupã ieşirea din acest modul, mesajul nostru nu va conţine decât cuvintele relavante cuvintele de legãturã fiind eliminate .În acest moment se va putea aplica algoritmul de atribuire a ponderilor mesajului nostru.

De cealaltã parte vom avea partea de modulele care se vor ocupa cu mesajele provenite din server-ul de micro-blogging. Într-o primã etapã se vor extrage mesajele din baza de date. Pentru ca aceasta sã poate fi executatã va fi nevoie de o conexiune la baza de date a micro-bloggingului respectiv. Odatã ce mesajele sunt salvate din baza de date şi preluate într-o listã, se vor apela aproximativ aceeaşi paşi de pre-clasificare ai mesajului ca şi în cazul anterior.Singura diferenţã notabilã ar fi aici modulul de extragere de informaţie din url. În acest modul se verificã dacã textul conţine un url, iar dacã acesta va conţine se va appenda la textul nostru mesajul din interiorul url-ului.

Odatã ajunse in etapa de clasificare, mesajele vor fi parcurse pentru compararea cuvintelor şi atribuirea ponderilor. Dupã aceastã etapã se va verifica dacã suma ponderiilor din fiecare text este mai mare decât un anumit prag iar mesajele relevante se vor afişa pe ecran. În acest moment toţi algoritmi au luat sfârşit, operaţia putând fi executatã din nou, fie pentru acelaşi mesaj ,fie pentru mesaje diferite.

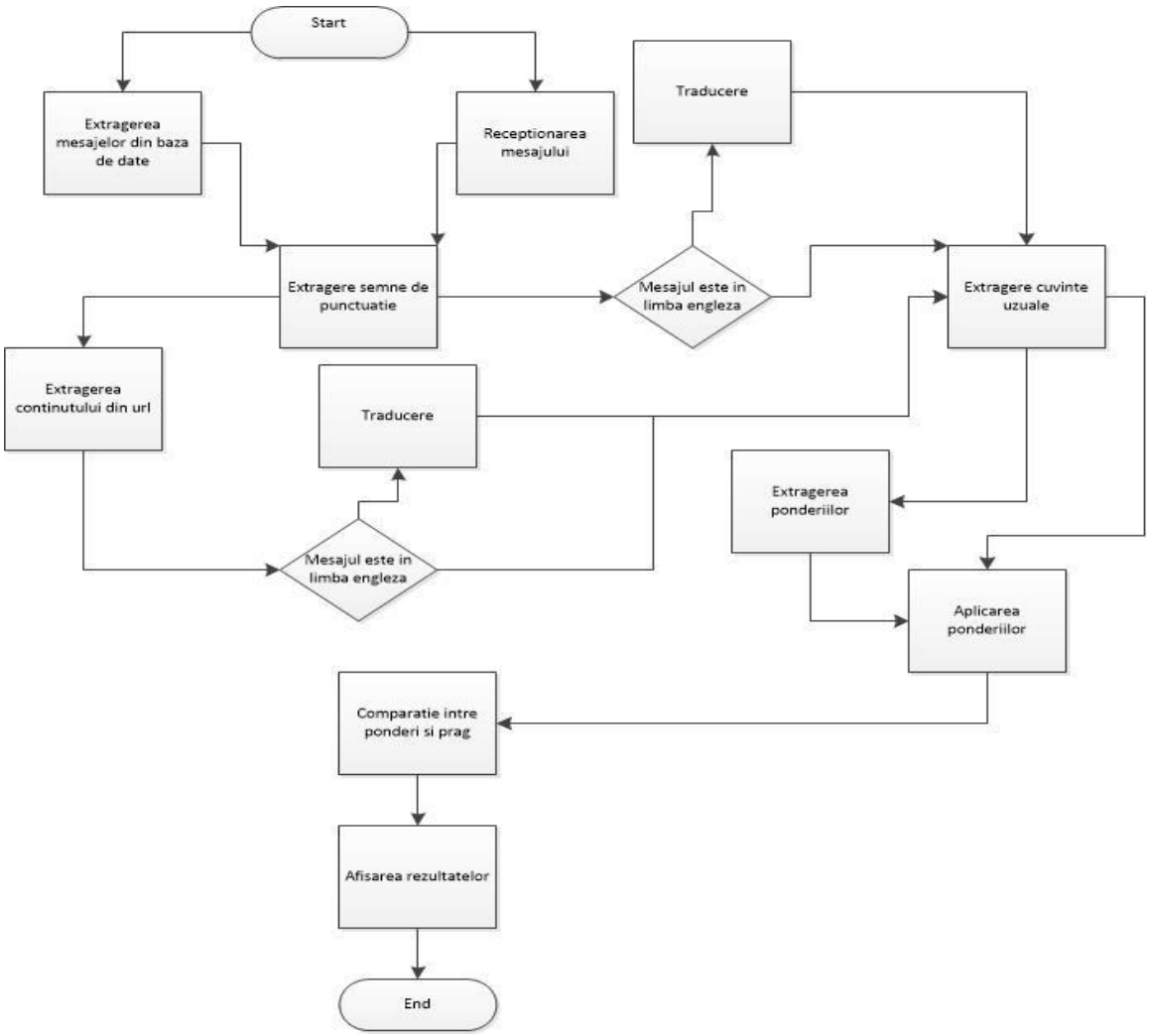

Figura 5.1 Flow-ul de execuţie al aplicaţiei

5.1.2. Arhitectura conceptualã generalã

Aplicaţia noastrã are ca scop principal filtrarea mesajelor provenite dintr-un serviciu de micro-blogging şi de aceea am considerat ca în diagrama conceptualã sã aparã şi aplicaţia de micro-blogging. De asemenea, filtrul nostru mai foloseşte şi o bazã de date de dicţionar de aceea aplicaţia de dicţionar de la wordnet va mai apãrea şi ea in diagramã. Nu in ultimul rând aplicaţia mai foloseşte şi un serviciu de translate folosit de microsoft care va apãrea si el în în diagramã.

Utilizatorul va putea interacţiona cu toate cele 3 aplicaţii, aplicaţia de dicţionar fiind disponibilã doar in varianta desktop. De notificat ar fi şi faptul cã linia dublã din desen semnificã interacţiune pe când linia simplã reprezintã apel.

Aplicaţia de dicţionar este una simplã şi este compusã din partea de „User Interface” şi de clasele care se ocupã cu maparea la baza de date. Dacã se va dorii faptul

ca un nou cuvânt sau o nouã bazã a unui cuvânt va trebuii sã se interacţioneze cu aceastã aplicaţie.

Aplicaţia de micro-blogging este cea pe care se va face filtrarea de mesaje. Ca şi în cazul anterior, pentru a putea crea un mesaj va trebui ca mesajul sã fie inserat în aplicaţia de micro-blogging. Aplicaţia de micro-blogging are la bazã tehnologia Spring. În figurã au fost prezentate pachetele din aplicaţie intr-o formã simplistã pentru uşurinţa înţelegerii.

Aplicaţia de filtrare este aplicaţia centralã a diagramei. Motivul pentru care baza de date este apelatã din pachetul de servicii este acela cã, baza de date este apelatã prin intermediul unei librãrii. Legãtura cu aplicaţia de micro-blogging se va face doar la nivelul bazei de date şi nu va exista cod comun între cele douã aplicaţii. O ultimã componentã folositã este api-ul oferit de microsoft pentru traducerea mesajelor.Acesta nu se aflã pe calculatorul local de aceea este doar apelat în aplicaţia noastrã el rulând în cloud.

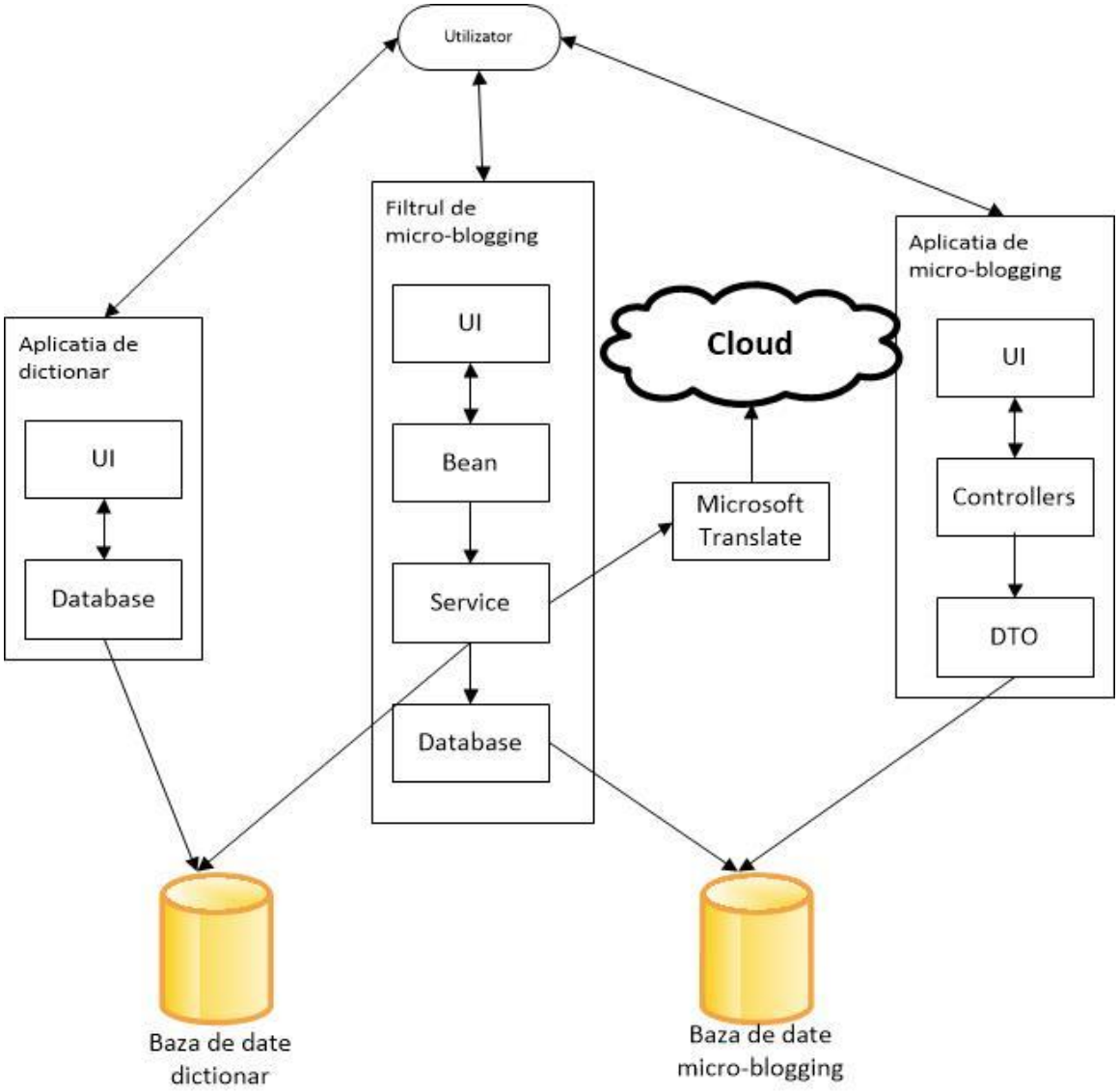

Figura 5.2 Diagrama conceptualã a aplicaţiei

5.2. Modelul cazurilor de utilizare

Sistemul propus şi dezvoltat este caracterizat de urmãtoarele cazuri de utilizare:

- extragerea url-ului din mesaj
- traducerea mesajelor care nu sunt în englezã
- eliminarea semnelor de punctuaţie si cuvintelor uzuale
- traducerea textelor în mod corect chiar dacã acestea nu conţin aceleaşi forme ale cuvintelor

5.2.1. Extragerea url-ului din mesaj

Actorul primar:Sistemul
Participanţi şi interese: Utilizatorul setează un mesaj de filtrare
Precondiţii: Utilizatorul cunoaşte aplicaţia şi scopul ei
Postcondiţii:Afişarea mesajului împreunã cu url-ul.
Scenariul principal de succes:

- utilizatorul porneşte sistemul de filtrare
- sistemul gãseşte mesajul care conţine url-ul
- sistemul extrage conţinutul url-ului şi îl apendează mesajului
- mesajul este afişat împreunã cu url-ul.

Fluxuri alternative(de eroare)

- în caz cã sistemul nu poate identifica în mod corect url-ul se va continua algoritmul de filtrare iar url-ul va fi considerat ca şi un cuvânt obişnuit.

5.2.2. Traducerea mesajelor care nu sunt în englezã

Actorul primar:Sistemul
Participanţi şi interese: Utilizatorul doreşte sã vizualizeze şi informaţiile care nu sunt în englezã
Precondiţii: Utilizatorul cunoaşte aplicaţia şi scopul ei
Postcondiţii:Afişarea mesajelor din alte limbi decât engleza.
Scenariul principal de succes:

- utilizatorul porneşte sistemul de filtrare
- sistemul gãseşte un mesaj care nu este in englezã
- sistemul traduce mesajul şi îl pune în locul mesajului existent
- mesajul este afişat în limba englezã

Fluxuri alternative(de eroare)

- în caz cã sistemul nu cunoaşte cuvântul de tradus, se va traduce cu ceea mai apropiatã formã a sa.

5.2.3. Eliminarea semnelor de punctuaţie si a cuvintelor uzuale

Actorul primar:Utilizatorul
Participanţi şi interese: Utilizatorul setează un mesaj de filtrare
Precondiţii: Utilizatorul cunoaşte aplicaţia şi scopul ei
Postcondiţii:Sortarea mesajelor în mod corect
Scenariul principal de succes:

- utilizatorul porneşte sistemul de filtrare
- sistemul a întâlnit un semn de punctuaţie sau un cuvânt uzual
- sistemul elimină din cuvânt semnul de punctuaţie respectiv elimină cuvântul uzual din mesaj
- sortarea mesajelor a fost făcută în mod corect

Fluxuri alternative(de eroare):

- în cazul în care punctuaţia face parte din cuvânt, cuvântul nu va fi supus în mod corect filtrării
- în cazul în care cuvântul uzual este un cuvânt relevant pentru aplicaţie(extrem de rar), ponderea cuvântului se va pierde.

5.2.4. Traducerea cuvintelor în mod corect chiar dacă acestea nu conţin aceleaşi forme ale cuvintelor.

Actorul primar:Utilizatorul
Participanţi şi interese: Utilizatorul setează un mesaj de filtrare
Precondiţii: Utilizatorul cunoaşte aplicaţia şi scopul ei
Postcondiţii:Sortarea mesajelor în mod corect
Scenariul principal de succes:

- utilizatorul porneşte sistemul de filtrare
- sistemul desparte mesajele din baza de date şi le compară cu cuvintele relevante din mesajul de filtrat
- sistemul verifică la comparare dacă cuvintele de comparat au acelaşi cuvânt de bază sau sunt sinonime

Fluxuri alternative(de eroare)

- cuvântul de bază lipseşte din dicţionarul local
- cuvântul de comparat nu apare ca şi sinonim în dicţionarul local

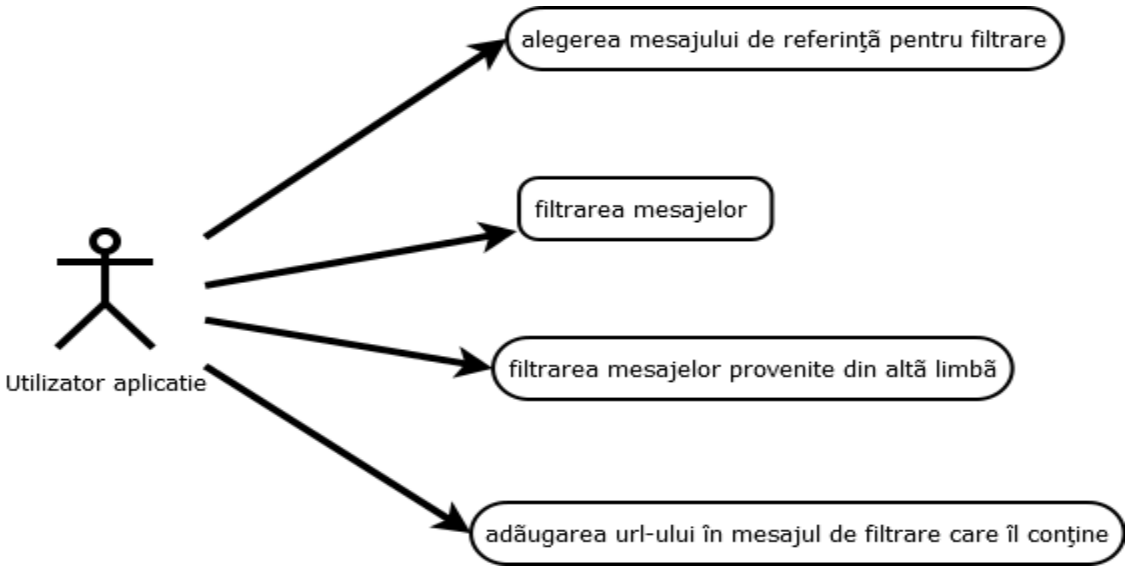

Figura 5.3 Diagrama use-case a aplicaţiei

5.3. Diagrama cazurilor de utilizare

În figura 5.2 este prezentată diagramă cazurilor de utilizare. Actorul este utilizatorul care poate efectua una din urmatoarele operaţiuni:

- alegerea mesajului de referinţă pentru filtrare
- filtrarea mesajelor
- filtrarea mesajelor provenite din altă limbă
- adăugarea url-ului în mesajul de filtrare care îl conţine

5.4. Diagrama de pachete

Lucrarea de faţă a fost creată folosiind tehnologia JSF ca şi tehnologie de bază iar ca şi tehnologiie de back-end clasele au fost grupate în pachete astfel incât să se respecte regullile unei aplicaţii pe layere.Primul pachet este cel de view în care avem paginiile JSF ale aplicaţiei. Cel de-al doilea pachet este pachetul de styles în care vom păstra fişierele css ale aplicaţiei. De notificat ar fi aici şi faptul că aceste două clase sunte cele care sunt responsabile cu interfaţa grafică a utilizatorului.

Cel de-al treilea pachet este cel de bean. Acesta este cel care face legătura între evenimentele care se produc în interfaţa grafică şi logica programului.Pachetul service este pachetul responsabil cu logica din spatele programului urmând iar pachetul database sa facă legatura la baza de date.

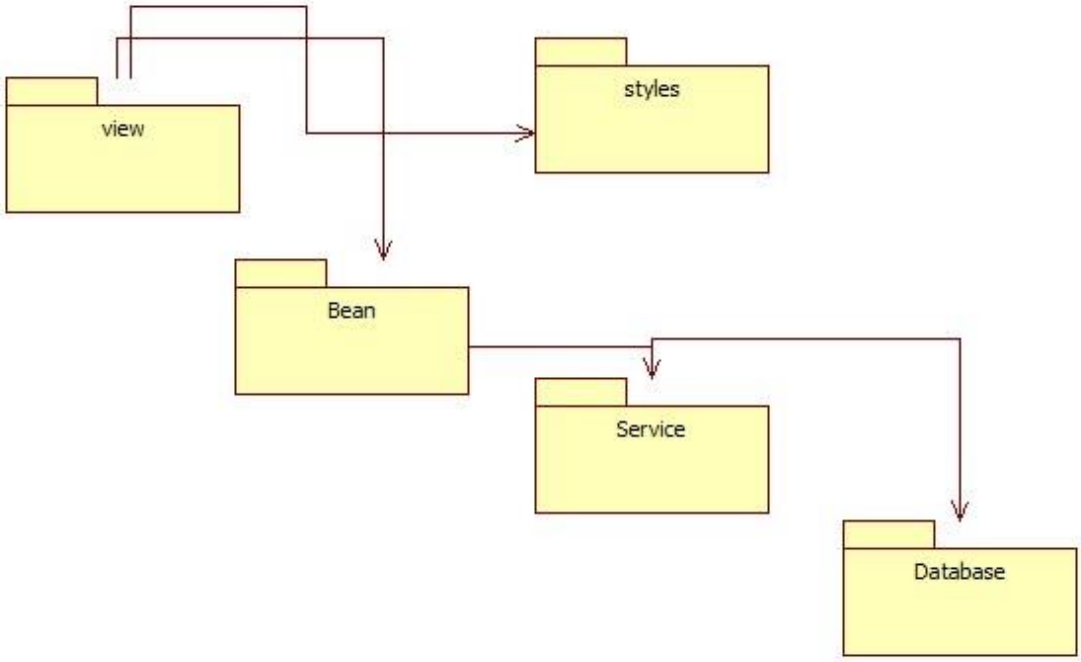

Figura 5.4 Diagrama de pachete

5.5. Diagrama de clase

În figura 5.5 se evidențiază diagrama de clase. Se poate observa faptul că este respectată o arhitectură pe layere.În cele ce urmează se vor explica clasele împreuna cu metodele sale cele mai importante

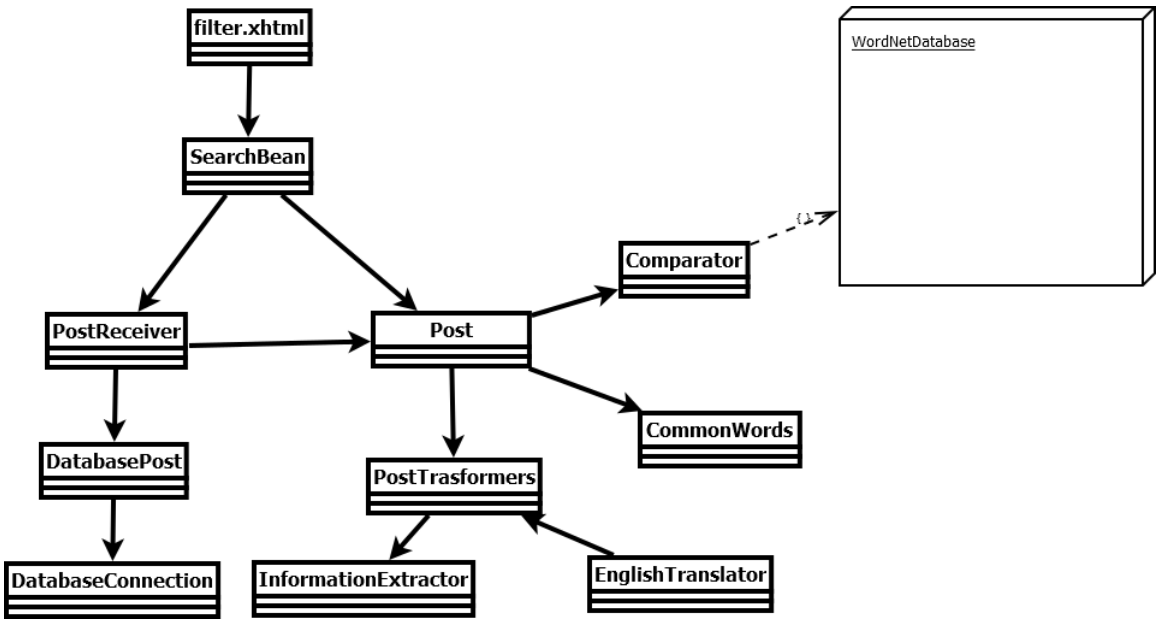

Figura 5.5 Diagrama de clase

5.5.1. filter.xhtml

Acest fişier este un fişier jsf specific in care este creatã interfaţa graficã a aplicaţiei.Conţine practic o intrare de tip input în care utilizatorul poate insera textul dorit spre filtrare şi un button care sã trimitã textul mai departe. De notificat ar fi faptul cã interfaţa graficã nu este punctul forte al aplicaţiei, punctul forte constând în aplicarea algoritmului.

5.5.2. SearchBean

Framework-ul de JSF presupune existenţa a cel puţin douã entitãţi.O paginã specificã JSF şi un Java Managed Bean. În cazul nostru, SearchBean nu este altceva decât un Managed Bean. Acesta are un atribut de instanţã searchText care se va updata cu valoare din pagina jsf. De asemenea în interiorul acestui bean mai existã şi metoda de listener a butonului.

În momentul în care butonul este apãsat, executã douã operaţii importante. Prima dintre ele este aceea cã se apeleazã clasa PostReceiver care va aduce postãrile din baza de date. A doua operaţie importantã este aceea cã, se va instanţia o clasa Post cu mesajul din interfaţa graficã şi se va porni algoritmul de filtrare de mesaje.

5.5.3. PostReceiver

Este clasa care se ocupã cu transformarea obiectelor provenite de la baza de date, în instanţe alea clasei Post.În aceastã clasã , toate datele din baza de date vor fi încapsulate într-o listã de obiecte de tip „Post”.De notificat aici ar mai putea fi şi faptul cã acestã clasã s-a adãugat şi din nevoia de respectare a arhitecturii pe layere, din managed bean nefiind îngãduitã apelarea unei clase din pachetul de baze de date.

5.5.4. DatabasePost şi DatabaseConnection

Aceste clase fac parte din pachetul „database” şi sunt practic clasele care aduc datele din baza de date.Clasa „DatabaseConnection” , dupã cum sugestiv îi spune numele se ocupã cu conectarea la baza de date a micro-bloggingului iar în clasa „DatabasePost” avem metoda getAllPosts() care aduce toate datele din baza de date. De notificat aici ar mai putea fi şi faptul cã, la crearea clasei DatabaseConnection s-a folosit design-patternul Singleton

5.5.5. Post

Aceasta este practic clasa de bazã a aplicaţiei deoarece conţine atât metode de instanţiere a variabilelor cât şi algoritmul de filtrare propiu-zisã a mesajului. Din SearchFilter se va instanţia mai întâi clasa Post cu mesajul din interfaţa graficã.În momentul în care s-a setat mesajul algoritmul de atribuirea a ponderilor începe.

În metoda splitMessage nu numai cã se va despãrţi mesajul nostru în cuvinte insã le va fi atribuitã şi o pondere cuvintelor respective. Astfel, dupã despãrţirea mesajului în cuvinte se va apela metoda removePunctuation(). Aceastã metodã este una simplã, care

verificã dacã cuvântul începe sau se terminã cu un semn de punctuaţie şi îl eliminã. După îndepartarea semnelor de punctuaţie se va apela clasa CommonWords care verificã dacã cuvântul este unul uzual sau nu. Dacã nu este uzual şi este primul cuvânt din propoziţie cuvântul va fi adaugat unui vector şi va avea ponderea 1. Dacã cuvântul este unul uzual atunci algoritmul îl va ignora complet şi va merge mai departe. De asemenea dacã cuvântul nu este primul din propoziţie , atunci cuvântul se va compara, folosiind clasa Comparator, cu toate cuvintele din vector. Dacã compararea nu este una pozitvã, cuvântul va fi şi el adãugat in vector cu ponderea 1. În momentul în care compararea este una pozitivã, cuvântul din vector cu care s-a fãcut compararea va fi incrementat cu 1. Prin aceastã modalitate, cuvintele care apar de cele mai multe ori vor avea o pondere mai mare decât cele care apar de mai puţine ori. La final, vectorul se va mai parcurge odatã, de aceastã datã ponderile fiind impãrţite la numãrul de cuvinte utile(care nu sunt comune).

În metoda seeRelevant() se va face practic algoritmul de sortare în funcţie de ponderile calculate în metoda de mai sus. În aceastã metoda se va lua fiecare postare in parte şi se va extrage mesajul. Prima operaţie care se va executa este aceea de tranformare a mesajului(mesajul nu este pe englezã sau conţine un url). După transformarea mesajului, se vor compara cuvintele din mesajul local cu cuvintele din mesajul din micro-blogging. Compararea se va face şi de aceastã datã cu ajutorul clasei Comparator. Dacã compararea va returna un rezultat pozitiv se va reţine ponderea localã. La sfârşitul parcurgerii se va va face o sumã pe toate ponderiile reţinute şi se va impãrţii ponderea rezultatã la numãrul total de cuvinte.

5.5.6. PostTransformers

Aceastã clasã are ca scop principal crearea unei legãturi dintre algoritmul de filtrare şi feature-urile de extragere de conţinut url şi de traducere de text. Mai întâi se va apela extragerea de conţinut url deoarece existã şansa ca informaţia utilã care va fi extrasã de pe url sã nu fie în limba englezã fiind nevoie de o traducere suplimentarã pentru aceasta.

5.5.7. InformationExtractor

Clasa InformationExtractor este clasa care se preocupã cu extragerea informaţilor utile din url. Aceastã clasã conţine trei metode: verifyIfWebsite extractInformation şi getInformationFromUrl.

În metoda verifyIfWebsite se va verifica dacã stringul introdus este un url. Programul va încerca sa creeze o conexiune pentru fiecare cuvânt. Daca codul returnat de server este altul decât 200 atunci cuvântul introdus de mine nu este website. Acest algoritm este considerat a fi lent unul lent insã este foarte eficient. O altã abordare ar putea fi folosirea de expresii regulate. Deşii pare mult mai rapid eficienţa algoritmului va scãdea considerabil.

În metoda getInformationFromUrl se va face extragerea propriu-zisã a url-ului. Pentru a extrage datele utile s-a folosit librãria Java Jsoup. Motivul pentru care s-a folosit aceastã librarie este uşurinţa in folosire(nu trebuie decât introduşi parametrii în locul potrivit) dar şi eficienţa sa.

Metoda extractInformation este metoda de legãturã dintre cele douã metode. Astfel, în aceastã metodã se vor evalua cuvintele url-ului unul câte unul şi se va verifica dacã cuvântul respectiv este sau nu url. Dacã este url atunci se va extrage informaţia. De notificat ar mai putea fi şi faptul cã dacã mesajul conţine mai multe url-uri atunci se vor extrage toate url-urile şi se vor concatena mesajului.

5.5.8. EnglishTranslator

Clasa EnglishTranslator este clasa care verificã dacã mesajul este scris într-o altã limbã diferitã de limba englezã, in acest caz traducându-l. Spre deosebire de clasa de mai sus aceastã clasã conţine o singurã metodã intitulatã sugestiv translate. În aceastã metodã se va verifica mai întâi dacã mesajul scris este intr-o limbã strãinã.Verificarea se va face folosiind o librărie de detecţie de cod. Această librărie nu conţine decât o mare parte din vocabularul limbii respective iar pentru ca aceasta sã poatã detecta corect este nevoie de minimum 15 cuvinte.

Traducerea textului în limba englezã s-a fãcut prin apelarea api-ului „bing translate" oferit de firma Microsoft. Acest api este de tip open-source şi ne oferã pânã la doua milioane de caractere de tradus. Pentru a-l putea folosii însã este nevoie de introducerea unui id şi unei parole secrete.

5.5.9. Common words şi comparator

Este clasa care conţine într-o variabilã staticã cele mai frecvente 250 de cuvinte din limba englezã. Clasa a fost fãcutã staticã pentru a putea suporta eventuale dezvoltări ulterioare(adãugarea unui cuvânt care apare mult prea des în contextul proiectului deşi nu face parte din cele mai întâlnite cuvinte). De notificat ar mai fi faptul cã, clasa a fost creatã în special pentru o mai bunã observare a logicii algoritmului.

Clasa comparator este o altã clasã mai aparte. Această clasã are o logicã proprie însã şi face legatura cu o altã aplicaţie de dicţionar. Aplicaţia „Dicţionar" este o aplicaţie de tip open-source. Pentru folosirea ei vom avea nevoie de baza de date a aplicaţiei(şi implicit de aplicaţia propriu-zisă) şi de o librărie oferitã tot de aplicaţie. Librăria va crea un obiect de tip WordNetDatabase care va fi de fapt un obiect care mapează toate relaţiile bazei de date. Ca şi parametru de creare a metodei va trebuii introdusã locaţia localã a bazei de date.

Astfel, metoda compare va instanţia un obiect de tip WordNetDatabase. Metoda noastrã are douã cuvinte ca şi parametrii. Intr-o primã etapã metoda comparã cele douã cuvinte pentru a vedea dacã sunt egale. Acest lucru, in caz de egalitate reduce foarte mult timpul de execuţie şi sporeşte eficienţa algoritmului. Dacã cuvintele nu sunt egale, cu ajutorul obiectului WordNetDatabase se vor extrage formele de bazã ale cuvintelor. Apoi tot cu ajutorul obiectului de bazã de date se vor extrage intr-o listã toate sinonimele cuvântului de bazã. Într-o ultimã etapã se vor parcurge listele şi se vor compara termenii. Dacã termenii coincid la un moment dat algoritmul se va oprii şi va returna un rãspuns pozitiv. Dacã niciunul din termeni nu coincid, algoritmul va întoarce un rãspuns negativ. Deşii pare un algoritm primejdios acesta va detecta dacã este vreo legãturã între cuvintele comparate.

5.6. Diagrama secvenţialã

În figura 5.6 este prezentatã digrama secvenţialã a aplicaţiei. Motivul pentru care metodele şi declanşatorii au fost scrişi cu cifre este acela de a explica diagrama in detaliu pentru a putea fi înţeleasã de cãtre toatã lumea.În cele ce urmeazã voi scrie o mini-legendã a metodelor folosite mai jos

1.Utilizatorul introduce textul de convertit şi apasã butonul de „Cautã”

2.Datoritã folosirii framework-ului de JSF, evenimentul din cadrul metodei cautã() este activat

3.Se apeleazã clasa PostReceiver pentru aducerea mesajelor din baza de date

4.Se apeleazã clasa PostTransformers care mapeaza clasa Post la baza de date

5.Se apeleazã clasa DatabaseConnection pentru a obţine o conexiune la baza de date

6.Se returneazã conexiunea la baza de date

7.Se returneazã o listã cu toate mesajele din baza de date.

8.Se returneazã o listã care conţine obiecte de tip Post mapate exact pe baza de date

9.Se apeleazã clasa Post pentru calcularea ponderiilor.

10.Se verificã dacã cuvântul de filtrat este cuvânt comun

11.Se returneazã adevãrat dacã cuvântul este cuvânt comun sau fals în caz contrar

12.Se comparã cuvântul cu alte cuvinte care sunt conţinute în mesajul meu şi nu sunt comune.

13. Se returneazã adevãrat dacã cuvintele au aceeaşi bazã sau sunt sinonime ori fals in caz contrar

14.Se apeleazã clasa PostTransformers care se ocupã cu algoritmi de pre-clasificare

15.Se apeleazã clasa InformationExtractor care are ca şi rol extragerea url-ului.

16.Se returneazã mesajul concatenat cu informaţiile utile din url. Dacã mesajul nu conţine url, se va returna mesajul gol.

17.Se apeleazã clasa EnglishTranslator care are ca şi rol traducerea textelor în englezã.

18.Se returnezã textul tradus în englezã sau acelaşi text, în cazul în care textul este deja în englezã.

19.Se returneazã o variantã transformatã a mesajului.

20.Se verificã dacã postarea este relevantã.Pentru aceasta fiecare cuvânt relevant din postarea localã va fi comparat cu fiecare cuvânt din postarea provenitã din baza de date.

21. Se returneazã adevãrat dacã cuvintele au aceeaşi bazã sau sunt sinonime ori fals in caz contrar

22.Se returneazã o listã cu postãrile relevante

23.Datoritã frame-workului de JSF, postãrile relevante vor apãrea pe interfaţa graficã.

24.Utilizatorul va poate vizualiza mesajele dorite.

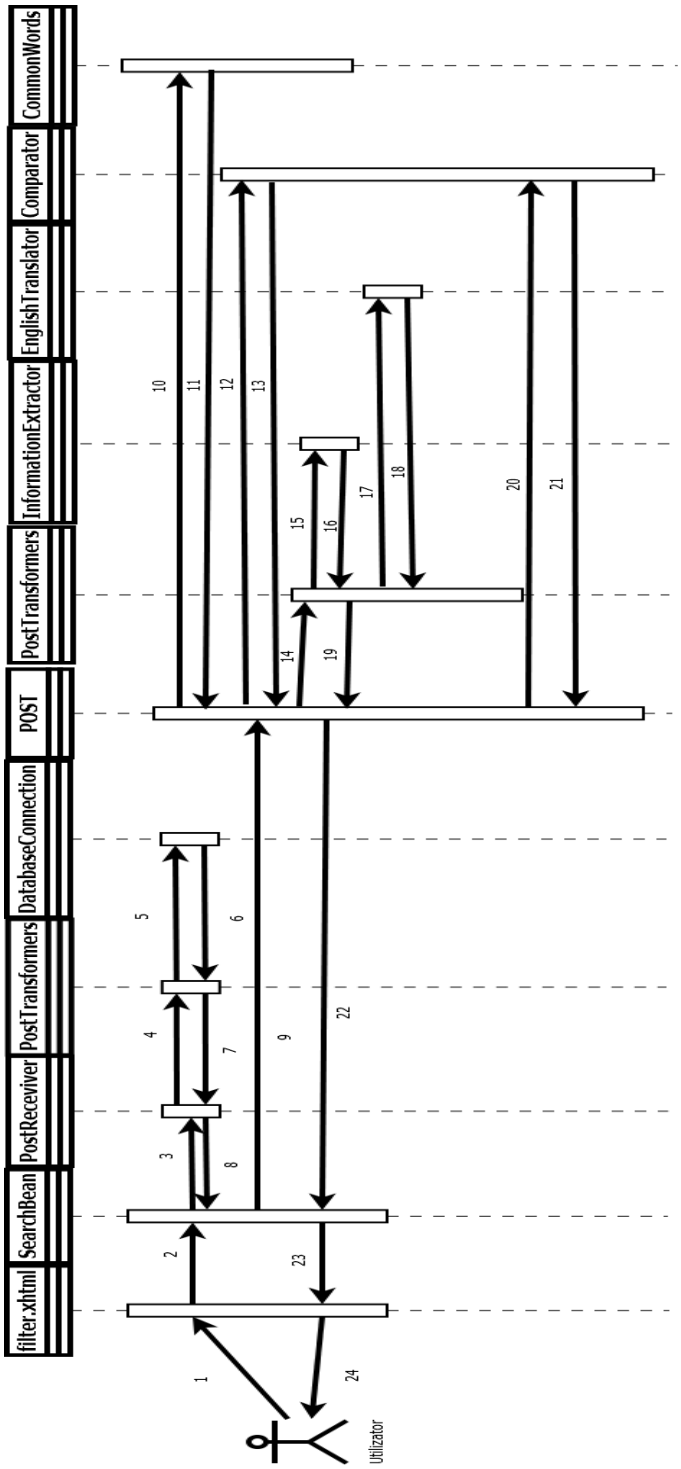

Figura 5.6 Diagrama secvenţialã

5.7. Tehnologii folosite în aplicaţie

Tehnologiile folosite în aplicaţie pot fi grupate în două mari categorii: tehnologii de front-end şi tehnologii de back-end.

5.7.1. Tehnologiile de front-end

Când discutăm despre *front-end* , vorbim despre acea parte a site-ului sau a aplicației web, pe care o putem vedea și cu care interacționează vizitatorii. Front-End-ul are două părți: design-ul (partea creativă) și dezvoltarea interfeței (partea de cod sau implementare HTML, CSS).

5.7.1.1. Javascript

JavaScript (JS) este un limbaj de programare orientat obiect bazat pe conceptul prototipurilor.Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul Javascript din aceste pagini fiind rulat de către browser. Limbajul este binecunoscut pentru folosirea sa în construirea siturilor web, dar este folosit și pentru acesul la obiecte încastrate (embedded objects) în alte aplicații.

Cea mai des întâlnită utilizare a JavaScript este în scriptarea paginilor web. Programatorii web pot îngloba în paginile HTML script-uri pentru diverse activități cum ar fi verificarea datelor introduse de utilizatori sau crearea de meniuri și alte efecte animate.

Browserele reţin în memorie o reprezentare a unei pagini web sub forma unui arbore de obiecte și pun la dispoziție aceste obiecte script-urilor JavaScript, care le pot citi și manipula. Arborele de obiecte poartă numele de Document Object Model sau DOM. Există un standard W3C pentru DOM-ul pe care trebuie să îl pună la dispoziție un browser, ceea ce oferă premiza scrierii de script-uri portabile, care să funcționeze pe toate browserele. În practică, însă, standardul W3C pentru DOM este incomplet implementat. Deși tendința browserelor este de a se alinia standardului W3C, unele din acestea încă prezintă incompatibilități majore, cum este cazul Internet Explorer. [6]

În proiectul nostru, Javascript are rolul principal de a nu lasă utilizatorul să caute un mesaj gol. Dacă utilizatorul va încerca să introducă un mesaj gol acesta va fi avertizat asupra faptului că mesajul este gol şi că nu poate fi procesată o filtrare asupra unui mesaj gol.

5.7.1.2. CSS

CSS (Cascading Style Sheets) este un limbaj care permite programatorilor să definească aspectul unei pagini web separând conținutul paginii de design-ul acesteia. Efectul de cascadă, după cum îi spune și numele, este ceea ce face ca acest limbaj să fie unic. Acest efect reprezintă parcurgerea fișierelor de la început până la sfârșit extrăgând informațiile necesare. De exemplu dacă pentru un anumit titlu este definită o culoare de două sau mai multe ori, ultima definiție va fi luată în considerare.

Includerea unor stiluri definite în limbajul CSS într-o pagină JSF este una foarte simplă și se poate realiza în 3 moduri:

- Se creează un fișier separat cu stilurile definite după care acesta se include în fișierul JSF.
- Se scriu blocuri de stiluri direct în fișierul JSF.
- Se scriu stiluri pentru fiecare element în parte acestea fiind aplicate doar acelui element JSF.

Deşi au apărut unele deficienţe de compatibilitate intre browsere, majoritatea proprietăților CSS3 au fost implementate cu succes in variantele browserelor noi.

Elementul HTML div este definit de urmatoarele proprietăți CSS: dimensiunea in lungime este redată de valoarea in pixeli a proprietății width, folosește o bordură de 2 pixeli, o bordură solidă de culoare gri-inchis definită de culoarea HEX #333333. Culoarea de fundal este gri deschis definită de HEX #dddddd. Bordura rotunjită este de 25 pixeli pentru toate cele 4 colțuri.

În proiectul nostru, css-urile au fost folosite pentru a oferii interfeţei grafice un aspect cât mai plăcut.Imaginile si stilurile folosite în aplicaţie sunt de tip open source. Având in vedere faptul cã aplicaţia este una care se ocupã cu filtrarea mesajelor s-a încercat adoptarea unui stil care sã simuleze faptul cã se foloseşte un stil didactic. Astfel , inputul a fost amplasat astfel încât sã dea impresia utilizatorului cã se scrie pe o tablã. De asemenea, tot în acest scop, culoarea font-ului a fost schimbatã la alb pentru a simula scrierea cu cretã pe tablã. Butonului i-a fost adãugat un stil aparte care îl face mai interesnt şi mai uşor de integrat în aplicaţia noastrã.Secţiunea de div a fost introdusã deoarece css-uril din aplicaţia mea folosesc într-un procent foarte mare div-uri pentru a importa stiluri aplicaţiei.

5.7.2. Tehnologiile de back-end

Back-End-ul de obicei constă în trei părți: un server, o aplicație de interfață și o bază de date. Rolul principal al acestuia este cel de management de conținut.

5.7.2.1. Java

Java [7] este un limbaj de programare de nivel înalt, dezvoltat de JavaSoft, companie în cadrul firmei Sun Microsystems.

Limbajul împrumută o mare parte din sintaxă de la C și C++, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos. Un program Java compilat, corect scris, poate fi rulat fără modificări pe orice platformă care e instalată o mașină virtuală Java (engleză Java Virtual Machine, prescurtat JVM). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format standard numit cod de octeți (engleză byte-code) care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

Dintre caracteristicile principale care m-au fãcut sã aleg limbajul Java ca şi limbaj de bazã în folosirea aplicaţiei amintim:

- **simplitate**, eliminã supraîncãrcarea operatorilor, moştenirea multiplã si toate "facilitãţiile" ce pot provoca scrierea unui cod confuz.

- **robusteţe**, eliminã sursele frecvente de erori ce apar in programare prin eliminarea pointerilor, administrarea automatã a memoriei si eliminarea fisurilor de memorie printr-o procedurã de colectare a 'gunoiului' care rulează în fundal
- **complet orientat pe obiecte** - eliminã complet stilul de programare procedural
- **usurinţa in ceea ce priveşte programarea in reţea**
- **securitate**, este cel mai sigur limbaj de programare disponibil în acest moment, asigurând mecanisme stricte de securitate a programelor concretizate prin: verificarea dinamicã a codului pentru detectarea secvenţelor periculoase, impunerea unor reguli stricte pentru rularea programelor lansate pe calculatoare aflate la distanta, etc

Java este un limbaj modern, din el putând deriva alte tehnologii şi metode folosite in aplicaţie. Framework-urile şi tehnologiile care au la bazã Java şi care au fost folosite în cadrul aplicaţiei noastra vor fi descrise în secţiuniile urmãtoare..

5.7.2.1.1. JSF

Java Server Faces[13] este un framework folosit pentru dezvoltarea de aplicaţii web.Acesta permite crearea de UI folosind componente standard, reutilizabile. Sintaxa lui se aseamãnã mult cu sintaxa html însã oferã şi anumite facilitãţi în plus. O altã caracteristicã importantã a paginiilor JSF este aceea cã asigurã persistenţa stării al nivel de WEB(permite crearea de UI folosind componente, salveazã starea componentelor UI cand clientul face o solicitare pentru o nouă pagină şi o restaurează când solicitarea este returnată)

Motivul pentru care acest framework este util pentru aplicaţia mea este acela cã, paginile JSF au o logicã simplã, sunt uşor de folosit şi întreţinut. Pentru un algoritm simplu de filtrare acestea se mulează exact pe ce avem nevoie. De asemenea, framework-ul JSF oferã o organizare proprie, care face codul mai uşor de înţeles şi urmărit.

5.7.2.1.2. Arhitectura Layers

Stilul Layers [14] (pe niveluri): adecvat aplicatiilor care pot fi descompuse in grupuri de subtaskuri, fiecare grup reprezentand un anumit nivel de abstractizare. Fiecare nivel utilizeaza servicii furnizate de nivelul(urile) de sub el.Arhitectura foloseşte un stil simplist , robust, sigur şi de aceea se potriveşte cerinţelor aplicaţiei mele

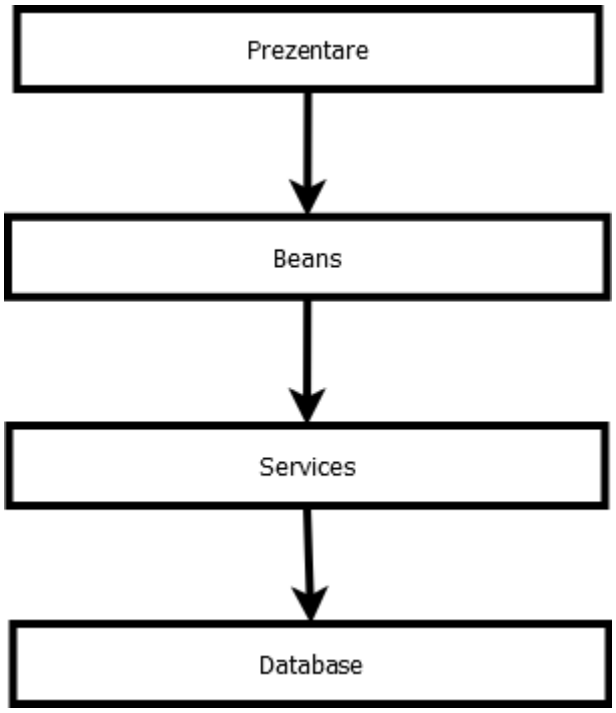

Figura 5.7 Modul de mapare al arhitecturii layers pe aplicaţie

5.7.2.1.3. Singleton

Design patterns(sau proiectarea modelelor) sunt un un set de soluţii ce pot fi aplicate la diverse probleme ce se intâlnesc frecvent in cadrul programării orientate pe obiect.

Modelul Singleton[15] este probabil cel mai simplu model. El este făcut pentru a permite accesul la o singură resursă care nu va fi niciodată duplicată, dar care trebuie facută disponibilă în orice moment al execuţiei aplicaţiei. Implementarea unui Singleton trebuie să satisfacă cele două condiţii de acces global si instanţiere singulară a resursei respective. Are nevoie de un mecanism de acces la clasa Singleton fără a instanţia un obiect şi un mecanism de a păstra informaţia asupra resursei ce urmează a fi folosită. Astfel acest model poate fi cel mai usor implementat prin crearea unei clase ce va conţine o metoda care va crea o nouă instanţă a clasei, daca aceasta nu există deja, iar dacă exista o va returna pe aceasta. Pentru a fi siguri că acea clasă nu va fi instanţiată in alt fel, constructorul este declarat protejat. Singleton este folosit în aplicaţia noastră pentru a instanţia clasa DatabaseConnection(nu dorim ca să facem mai multe conexiuni la baza de date în cadrul aceleiaşi aplicaţii).

5.7.2.1.4. Librăria JSOUP

Librăria JSOUP[16] este o librărie Java care se ocupă în principal cu parsare de String-uri. Motivul pentru care s-a folosit librăria JSOUP în aplicaţia noastră este pentru a parsa un site web şi a scoate informaţiile utile din interiorul său.

Pentru ca metoda să funcţioneze se presupune că avem un url sub forma unui String şi că dorim să extragem informaţiile utile din acel url. String-ul poate proveni

dintr-un fişier de intrare sau dintr-un alt site web. Librăria va oferii suport pentru toate aceste tipuri de proveniență.

Metoda care parsează string-ul şi returnează conţnutul url-ului este Jsoup.parse(String html) sau Jsoup.parse(String html, String baseUri) dacă pagina provine de pe un alt site şi se doreşte sa se ia url-ul absolut

Metoda parse(String html, String baseUri) parsează rul-ul într-un document nou. Argumentul baseUri este folosit pentru a transforma url-ul relativ într-unul absolut. Dacă se cunoaşte cu siguranţă faptul că url-ul este o cale absolută atunci se va putea folosii metoda parse(String html).

Atâta timp cât se va parsa un string care nu este null, este garantat faptul că vom avea o parsare efectuată cu succes şi sensibilă(cu un document care conţine cel puţin un tag de <head> şi un tag de <body>).

Odată ce avem documentul putem accesa data din interiorul documentului folosind metodele definite în clasa document cu argumente specifice xml-urilor cum ar fi Element si Node.

5.7.2.2. MySql

MySQL[17] este un sistem de gestiune a bazelor de date relaționale, produs de compania suedeza MySQL AB și distribuit sub Licența Publică Generală GNU. Este cel mai popular SGBD open-source la ora actuală, fiind o componentă cheie a stivei LAMP (Linux, Apache, MySQL, PHP).

Deși este folosit foarte des împreună cu limbajul de programare PHP, cu MySQL se pot construi aplicații în orice limbaj major. Există multe scheme API disponibile pentru MySQL ce permit scrierea aplicațiilor în numeroase limbaje de programare pentru accesarea bazelor de date MySQL, cum are fi: C, C++, C#, Java, Perl, PHP, Python, FreeBasic, etc., fiecare dintre acestea folosind un tip spefic API. O interfață de tip ODBC denumită MyODBC permite altor limbaje de programare ce folosesc această interfață, să interacționeze cu bazele de date MySQL cum ar fi ASP sau Visual Basic. În sprijinul acestor limbaje de programare, unele companii produc componente de tip COM/COM+ sau .NET (pentru Windows) prin intermediul cărora respetivele limbaje să poată folosi acest SGBD mult mai ușor decât prin intermediul sistemului ODBC. Aceste componente pot fi gratuite (ca de exemplu MyVBQL) sau comerciale.

Licența GNU GPL nu permite încorporarea MySQL în softuri comerciale; cei care doresc să facă acest lucru pot achiziționa, contra cost, o licență comercială de la compania producătoare, MySQL AB.

MySQL este componentă integrată a platformelor LAMP sau WAMP (Linux/Windows-Apache-MySQL-PHP/Perl/Python). Popularitatea sa ca aplicație web este strâns legată de cea a PHP-ului care este adesea combinat cu MySQL și denumit Duo-ul Dinamic. În multe cărți de specialitate este precizat faptul ca MySQL este mult mai ușor de invățat și folosit decât multe din aplicațiile de gestiune a bazelor de date, ca exemplu comanda de ieșire fiind una simplă și evidentă: „exit” sau „quit”.

Pentru a administra bazele de date MySQL se poate folosi modul linie de comandă sau, prin descărcare de pe internet, o interfață grafică: MySQL Administrator și MySQL Query Browser. Un alt instrument de management al acestor baze de date este aplicația gratuită, scrisă în PHP, phpMyAdmin.

MySQL poate fi rulat pe multe dintre platformele software existente: AIX, FreeBSD, GNU/Linux, Mac OS X, NetBSD, Solaris, SunOS, Windows 9x/NT/2000/XP/Vista.

În aplicaţie, MySql este folosit pentru a reţine datele din aplicaţia de micro-blogging. Datele din baza de date vor trebui sa poatã fi extrase atât din interirul filtrului cât si din interiorul aplicaţiei de aceea sistemul de gestiune al bazei de date trebuie sã fie flexibil. O altã caracteristicã importantã a MySql este securitatea. Doar aplicaţia de micro-blogging şi aplicaţia de filtrare de mesaje pot avea acces la baza de date restul aplicaţilor având acces interzis. Acest lucru se va face folosind un username şi o parola secretã bazei de date.

5.7.3. Alte tehnologii

Deşii aceste tehnologii face parte din back-endul aplicaţiei nu pot fi considerate tehnologii specifice de back-end şi de aceea vor fi tratate separat. Tehnologiile descrise în acest capitol sunt tehnologiile cele mai moderne pe care aplicaţia le deţine

5.7.3.1. Aplicaţia WordNet

WordNet[8] este o bază de date lexicalã din limba engleză. Substantive, verbe, adjective și adverbe sunt grupate în seturi de sinonime cognitive (synset-uri), fiecare exprimând un concept distinct. Synset-urile sunt interconectate prin intermediul unor relații conceptuale-semantice și lexicale. Rețeaua rezultatã de cuvinte și concepte semnificativ legate poate fi navigatã cu browser-ul. WordNet este, de asemenea, în mod liber și la dispoziția publicului pentru descărcare.

WordNet este o aplicaţie care instalează pe calculatorul local o aplicaţie de sine stătãtoare care simulează funcţionearea unui dicţionar explicativ în limba englezã. Aplicaţia însã pune la dispoziţia utilizatoriilor o serie de librării prin care poate fi accesatã baza de date a aplicaţiei.

Pentru a accesa baza da date însã avem nevoie sã setãm calea pânã la baza de date. Dupã ce setãm calea, putem folosii metodele din librăriile date pentru a obţine acces la formele de bazã şi sinonimele cuvintelor.

Prima metodã care v-a fi apelatã este metoda getSynsets. Aceastã metodã primește ca și parametru un cuvânt și returnează totalitatea sinonimelor din baza de date ale cuvântului. Aceastã metodã se va aplica pentru ambele cuvinte.

A doua metodã care se va apela este metoda getWordForms. Aceastã metodã interogheazã baza de date a dicţionarului şi returnează forma de bazã a cuvântului respectiv. Baza de date a aplicaţiei WordNet este o bazã de date relaţionalã , bazã de date a cãrei calitate importantã este aceea cã face query-urile sa fie mult mai rapide.

5.7.4. Integrarea cu alte sisteme

Application Programming Interface (prescurtat API) reprezintă denumirea în limba engleză a unei interfețe pentru programarea de aplicații. De obicei este vorba despre interfața dintre programele de aplicație și sistemul de operare, care stabilește în

amănunt modul în care programele de aplicație pot accesa (apela) serviciile sistemului de operare sub care rulează.Pe scurt un API nu este nimic altceva decât o resursă de internet facută disponibilă si care poate fi accesată din codul tău local.

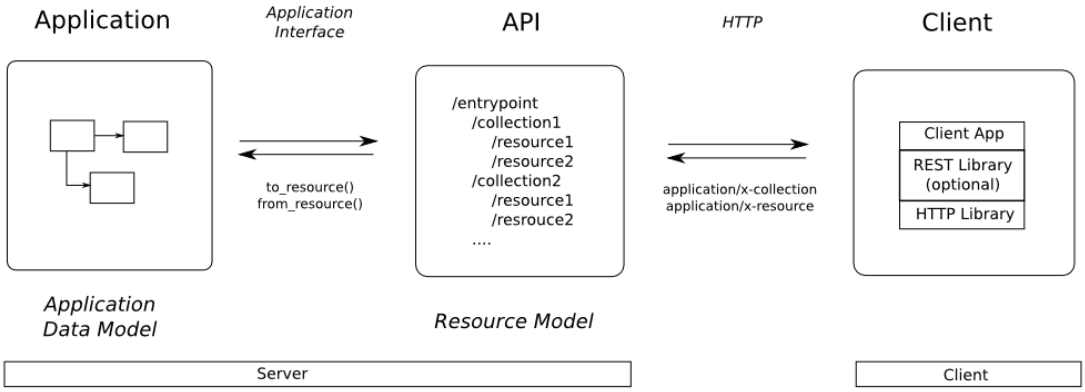

Figura 5.8 Desigul principal al unui API

5.7.4.1. Bing Translator API

Microsoft Translator oferă un set puternic de API-uri de servicii web pe care dezvoltatorii le pot utiliza pentru a profita de tehnologia sa de traducere în propria lor aplicații, servicii sau site-uri web. Acest API poate fi apelat într-un număr de moduri inclusiv un HTTP REST, un serviciu AJAX apelabile și un serviciu SOAP Web.

Traducătorul API Microsoft [9], este un serviciu on-line, şi este disponibil direct de la Windows Azure Marketplace. Microsoft Translator API este vândut ca un abonament lunar în funcție de numărul de caractere de text apelate din API. API-ul este disponibil gratuit pentru utilizare de până la 2 milioane de caractere pe lună.

Metodelele pe care API-ul le oferă dezvoltator-ului sunt enumerate mai jos:

- *Translate –traduce textul dintr-o limbã in alta*
- *TranslateArray – traduce un şir de texte dintr-o limbã în alta*
- *Speak – Returneazã o monstrã audio a textului vorbit în limba doritã*
- *Detect – Detectează limba mesajului*
- *DetectArray- Detectează limbiile dintr-o listă de string-uri*
- GetTranslations- *Returnează un vector cu traduceri alternatice ale textului*
- GetTranslationsArray-*Returnează un vector cu traducer alternative la un vector de texte*
- GetLanguageNames, GetLanguagesForSpeak,GetLanguagesForTranslate-*returnează o listă cu limbile suportate*.
- AddTranlation, AddTranslationArray- *permite developeri-lor să adauge translatările proprii.*

În plus față la utilizarea API-ul, developerii pot utiliza Microsoft Translator Widget pentru a obține traduceri instantanee de pagini web fără a fi nevoie de a scrie orice cod. Widget-ul este o mică bucată de cod HTML, care poate fi plasatã oriunde pe site, şi oferă utilizatorului final, o opțiune de a traduce textul de pe site în oricare dintre limbile acceptate. Widget-ul include caracteristici ale traducerii care vã vor permite sã colaboraţi, sã furnizaţi traduceri alternative, vă permit să editați traducerile automate și votaţi cea mai bună traducere pentru a îmbunătaţii cadrul pentru comunitate.

Capitolul 6. Testare şi validare

În cadrul acestui capitol se va prezenta modul de testare a aplicației precum și principalele clase de validare a datelor.

Pe parcursul dezvoltării unui sistem software, testarea este foarte importantă deoarece ajută la prevenirea erorilor și la rezolvarea acestora. De fiecare dată când o funcționalitate nouă este integrată în sistem este necesar să testăm atât acea funcționalitate în particular cât și întreg sistemul după integrarea acesteia pentru a observa modul în care aplicația se comportă. Pe parcursul dezvoltării proiectului au fost testate funcționalitățile și prevenite eventualele erori.

6.1. Validarea datelor

Validarea datelor se face în aplicaţia noastrã cu ajutorul pãrţii de front-end.Astfel, în momentul în care utilizatorul doreşte sã aleagã ca şi mesaj de referinţã un mesaj gol va fi avertizat asupra faptului cã mesajul ales este gol. De asemenea dacã dimensiunea mesajului va depãşi limita de 500 de caractere, utilizatorul va fi avertizat din nou asupra faptului cã mesajul este prea lung.

Principalul avantaj al validării datelor pe partea de front-end îl consideră faptul că întregul proces nu va funcţiona dacă nu are date. Astfel, dacă mesajul nu va respecta cerinţele utilizatorul va fi avertizat asupra acestui lucru iar întregul sistem va fi pus în aşteptare. Sistemul va ieşi din aşteptare atunci când utilizatorul va introduce date potrivite.Un alt avantaj al folosiri validării pe partea de front-end ar putea fi considerat şi costul redus dar şi performanţa. Cu doar câteva linii de cod(cost mic) utilizatorul va putea obţine o aplicaţie sigurã.

6.2. Testarea aplicaţiei

Testarea aplicației face referire la verificarea componentelor cu scopul ca aceastea să îndeplinească cerințele funcționale propuse. În procesul de testare am verificat periodic modul în care se comportă aplicația în diferitele navigatoare pe care le-ar putea folosi utilizatorii (Mozilla, Chrome, Internet Exporer sau Safari).

Cel mai important de testat în cadrul aplicaţiei noastre va fi algoritmul de testare. Acesta, în cazul în care nu va funcţiona sau va funţiona greşit, va produce rezultate greşite. Pentru a putea testa cât de bun este algoritmul nostru se vor introduce 9 mesaje care conţin informaţii despre subiecte diferite.Pentru testarea algoritumului s-a folosit şi feed-backul dat de anumiţi utilizator.Utilizatorul clasifica mesajele ca fiind relevante sau irelevante în funcţie de condiţile sale proprii.

Pentru a putea să ne dăm seama dacă algoritmul nostru este unul bun sau ne trebuie să definim anumite metrici. Aceste metrici sunt:

$$Acurateţe = \frac{număr\ relevant\ corect + număr\ irelevant\ corect}{numărul\ total\ de\ mesaje}$$

Figura 6.1 Definirea acurateţii

Acurateţea reprezintã raportul dintre numãrul de clasificãri corecte şi numãrul total de clasificãri. În figura 6.1 am notat cu numãr relevant corect ca fiind numãrul de mesaje pe care algoritmul nostru l-a clasificat corect iar cu numãr irelevant corect am notat numãrul de mesaje irelevante pe care algoritmul l-a clasificat corect.

$$Precizie = \frac{numãr\ relevant\ corect}{numãrul\ total\ de\ mesaje}$$

Figura 6.2 Definirea preciziei

Precizia mãsoarã raportul dintre numãrul de mesaje relevante clasificate corect şi numãrul total de mesaje. De notificat este faptul cã numãrul irelevant de mesaje va fi ignorat.

$$Reapelare = \frac{numãr\ irelevant\ incorect}{numãrul\ total\ de\ mesaje}$$

Figura 6.3 Definirea reapelãrii

Reapelarea este opusul preciziei şi reprezintã raportul dintre numãrul de mesaje irelevante clasificate corect şi numãrul total de mesaje.

$$FScore = \frac{2 * Precizie * Reapelare}{Precizie + Reapelare}$$

Figura 6.4 Definirea FScore

Scorul F sau FScore reprezintã cea mai complexã metricã de mãsurã a eficienţei alogoritmului şi reprezintã media armonicã dintre precizie şi reapelare.

În tabelul 6.5 se pot vedea subiectele de dicuţie şi valoriile care au fost luate pentru metricile discutate.

Subiectul Mesajului	Acurateţe	Precizie	Reapelare	FScore
Sarcinã-pãrinţi	.855	.895	.833	.899
Film	.792	.89	.709	.789
Ecologie	.792	.787	.947	.86
Market	.878	.896	.977	.935
Programatori	.695	.722	.585	.646
Ştiri fructe	.791	.844	.882	.863
Actori comedianţi	.462	.498	.572	.531
MySql	.694	.684	.57	.621
Bazã de date	.882	.923	.873	.897

Tabela 6.5 Eficienţa sistemului mãsuratã în metrici

Capitolul 7. Manual de instalare şi utilizare

În acest subcapitol sunt prezentate cerinţele pentru instalare şi utilizare a aplicaţie precum şi paşii necesari pentru aplicaţie.

7.1. Instalarea aplicaţiei

După cum se ştie deja, pentru ca filtrul sã poatã funcţiona este nevoie de instalarea altor module suplimentare respectiv aplicaţia de micro-blogging şi aplicaţia de dicţionar.

7.1.1. Aplicaţia de micro-blogging

Micro-bloggingul local este o aplicaţie de tip open-source. Aceasta se gãseşte pe site-ul github si poate fi descãrcatã gratis.Pentru a instala aplicaţia avem nevoie de mediul virtual Eclipse şi sistemul de gestiune a bazelor de date MySql. De asemenea pentru a putea rula programul avem nevoie de Java Development Kit 1.6 sau o versiune mai nouã. Pentru a se putea crea baza de date a micro-bloggingului trebuie modificat fişierul context.xml cu username-ul şi parola bazei de date locale. În acest moment aplicaţia poate fi rulatã cu succes.

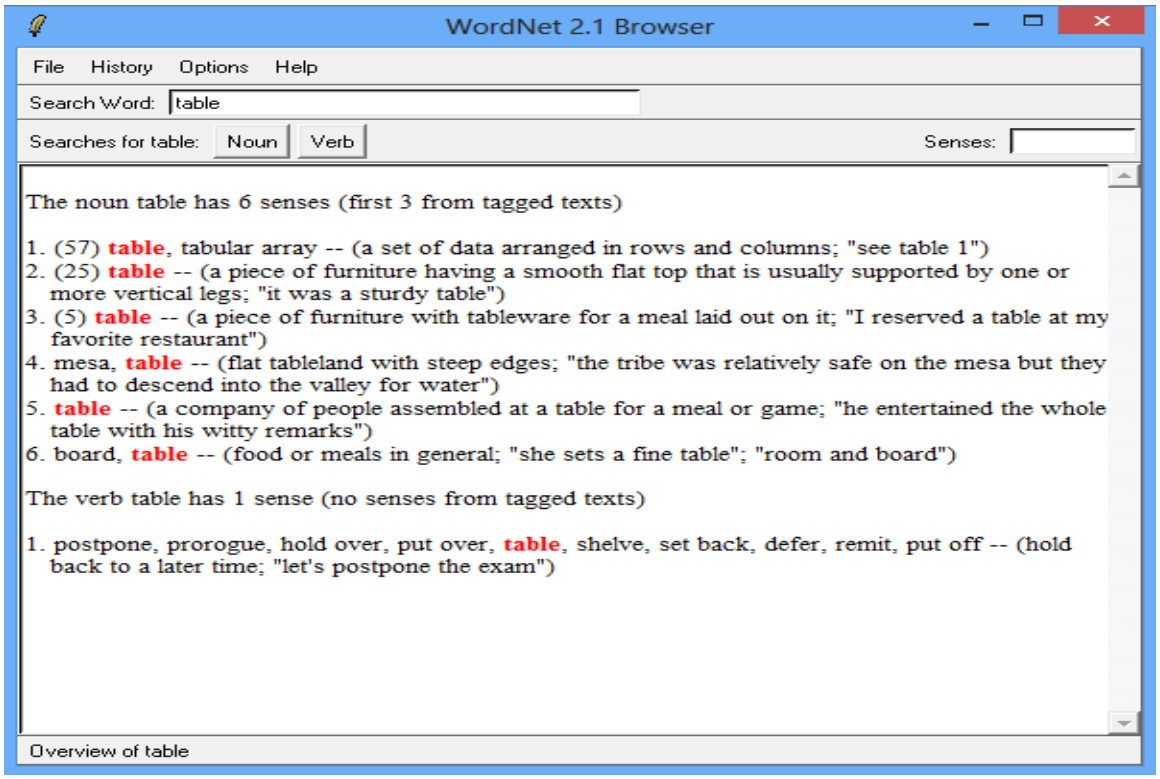

Figura 7.1 Aplicaţia WordNet

7.1.2. Aplicaţia de dicţionar

Dicţionarul nostru local este de fapt aplicaţia oferită de WordNet pentru dicţionar.Aceasta se găseşte ca fiind open source şi poate fi descărcată de pe site-ul universităţii din Princeton.Pentru ca aplicaţia de micro-blogging să poată funcţiona trebuie ca dicţionarul să fie instalat precum în fişierul de instalare pe care îl conţine(„C:\Program Files \WordNet”).În figura 7.1 ne este prezentată interfaţa grafică a dicţionarului.

7.1.3. Aplicaţia de filtrare de mesaje

Pentru ca aplicaţia de micro-blogging să poată funcţiona avem nevoie ca şi in cazul aplicaţiei de micro-blogging de Java Development Kit 1.6 sau o versiune mai nouă. De asemenea aplicaţia a fost creată în mediul de dezvoltare NetBeans. Nu în ultimul rând, pentru a putea accesa şi api-ul de traducere oferit de Microsoft vom avea nevoie şi de o conexiune la Internet.

7.2. Utilizarea aplicaţiei

Aplicaţia noastră se ocupă cu filtrarea mesajelor. Pentru a putea filtra mesajele avem nevoie ca mai întâi să avem anumite mesaje în aplicaţia de micro-blogging. Inserarea mesajelor nu se va putea face decât prin intermediul aplicaţiei de micro-blogging. Pentru a putea insera un mesaj nu este nevoie decât să ne logăm(în cazul în care nu avem un cont, acesta va trebui creat) şi să postăm mesajul. În figura 7.2 ne este prezentat modul în care se poate introduce un mesaj în aplicaţie.

Figura 7.2 Aplicaţia de micro-blogging

Dupã ce avem mesaje în interiorul micro-bloggingului vom putea sã aplicãm şi procedura de filtrare de mesaje.

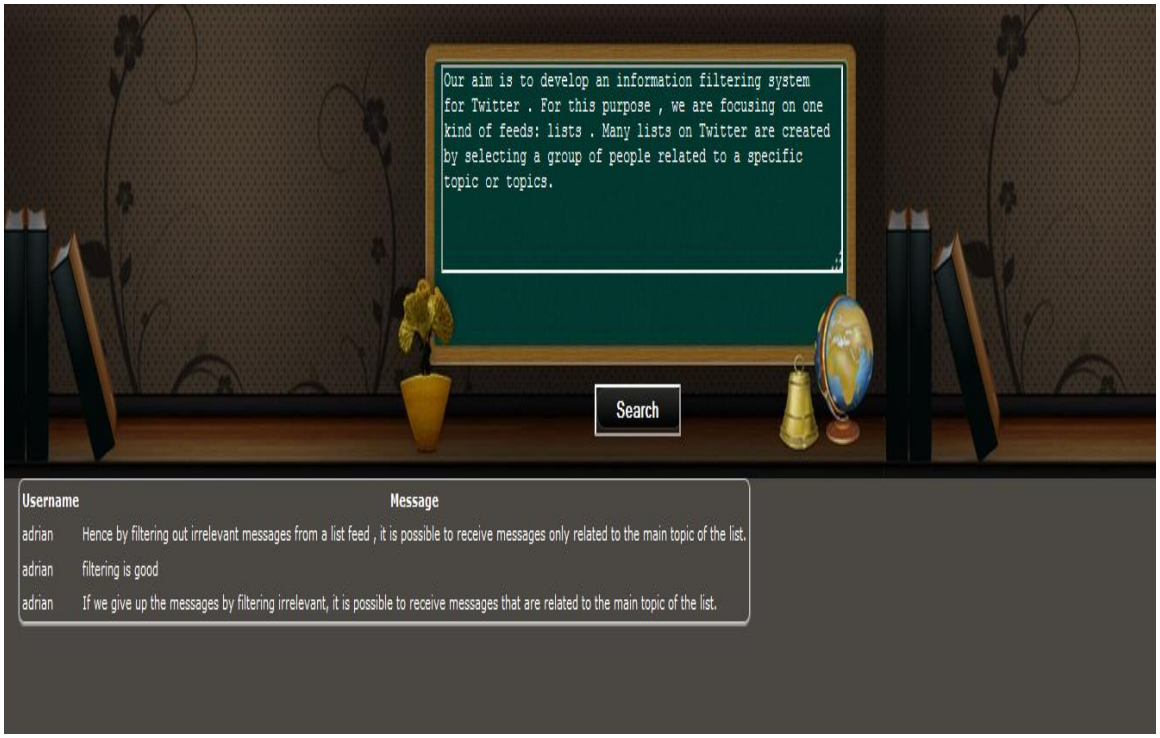

Figura 7.3 Filtrul de micro-blogging

Dupã introducerea în cãsuţa a textului dorit spre filtrare şi apãsarea butonului de search în tabel ne vor apãrea mesajele relevante. Primul mesaj este considerat a fi relevant deoarece, dupã cum se poate observa apar foarte multe cuvinte care sunt asemãnãtoare ca formã cu textul nostru. În cel de-al doilea mesaj, deşi este unul scurt, ponderea cuvântului filtering este una mare şi de aceea mesajul nostru va fi considerat ca fiind unul relevant.Dupã cum se poate observa, cel de-al treilea mesaj din figura 7.3 este traducerea în englezã a primului mesaj din figura 7.2 Astfel, sistemul nostru a tradus mesajul, a aplicat algoritmul şi a considerat cã informaţia conţinutã este una relevantã. Ca şi concluzie se poate observa faptul cã toate traduceriile au fost fãcute în mod corect.

Capitolul 8. Concluzii

În această secţiune se prezintă rezultatele în concordanţă cu obiectivele propuse. Toate obiectivele propuse iniţial au fost îndeplinite, astfel:

- Aplicaţia are abilitatea de a filtra textele provenite de la un serviciu de micro-blogging in texte relevante şi texte irelevante
- Aplicaţia poate distinge în momentul sortării cuvintele uzuale cum ar fi prepoziţiile, conjuncţiile de cuvintele folosite pentru sortare
- Algoritmul funcţionează in mod corect, ponderile fiind atribuite în funcţie de frecvenţa cu care acestea apar în mesaj
- Aplicaţia permite extragerea conţinutului provenit din link-uri, adăugarea acestuia în mesaj şi sortarea lui
- Aplicaţia este capabilă să traducă mesajele provenite din altă limbă şi să le adauge la algoritmul de filtrare
- Algoritmul de sortare trebuie să fie capabil ca, în momentul în care avem de comparat 2 sinonime compararea sa aibă loc ca şi când am avea de comparat 2 cuvinte identice
- Cerinţele funcţionale, nefuncţionale şi de implementare au fost respectate cu succes.

8.1. Contribuţii personale

Ca şi o contribuţie am creat un algoritm de clasificare care se bazează pe atribuirea ponderilor cuvintelor din interiorul mesajelor, atribuire bazată pe frecvenţa mesajelor care apar în text.Am folosit aceaste ponderi pentru a sorta mesajele în mesaje relevante şi mesaje irelevante.În plus am folosit şi sortarea mesajelor provenite din url-uri dar şi mesajele provenite din alte limbi. Experimentele au arătat faptul că aplicaţia noastră este una destul de eficientă.

Am arătat în aplicaţie şi faptul că link-urile incluse în mesaj sunt o sursă bună pentru informarea asupra subiectului respectiv.Am tratat mesajele cu link-uri prin extragerea informaţiei utile din url şi aplicarea filtrului de mesaje asupra sa. Experimentele au arătat faptul că eficienţa filtrării are o pondere foarte mare.

Cu ajutorul experimentelor din filtrarea de mesaje am ajuns la valori ale acurateţii între 75% şi 85%. Mai departe am observat că fiecare tip de mesaj din cadrul experimentelor are un anumit topic şi un anumit grad de acurateţe(în funcţie de cât de specific este fiecare tip de mesaj).

În final, există o versiune online pentru filtrarea de mesaje. Acesta reţine ponderea cuvintelor din mesajul filtrat şi calculează pentru fiecare mesaj introdus valoarea sa de relevanţă.

8.2. Dezvoltări ulterioare

Aplicaţia de faţă dispune de mai multe direcţii de dezvoltări ulterioare cum ar fi:

- Posibilitatea de a alege din mai multe aplicaţii de micro-blogging. Utilizatorul va putea să-şi aleagă din interfaţa grafică micro-bloggingul pe care se doreşte să se facă filtrarea de mesaje. Dacă nici un micro-blogging nu va fi ales utilizatorul se va deduce faptul că utilizatorul doreşte informaţii de pe toate serviciile de micro-blogging.
- Posibilitatea de a insera în câmpul de căutare un url. Astfel, se vor afişa mesajele relevante cu conţinutul acelui url.
- Posibilitatea introducerii în interiorul câmpului a unui mesaj într-o altă limbă diferită de engleză.Utilizatorul ar trebui să primească informaţia în limba în care a fost scris şi texul pentru căutarea informaţiei.
- Adoptarea unor algoritmi şi tehnologii care să mărească viteza de execuţie a aplicaţiei.
- Introducerea unor modele de tipar(“template-uri”) pentru tipuriile uzuale de mesaje folosite

Bibliografie

[1] „Blog” [Online], http://ro.wikipedia.org/wiki/Blog

[2] „Proceedings of the 16th international conference on World Wide Web” New York, SUA , 2007.

[3] Douglas W. Ouard, *The state of art in text filtering* , University of Maryland,2009

[4] Fabrizio Sebastiani,*Machine learning in automated text categorization* , Consiglio Nazionale delle Ricerche,Italia

[5] „Most Common Words in English”[Online], http://en.wikipedia.org/wiki/Most_common_words_in_English

[6] „Javascript”, [Online], http://ro.wikipedia.org/wiki/JavaScript

[7] „Java”, [Online], http://profs.info.uaic.ro/~acf/java/curs/1/introducere.html

[8] „Wordnet”, [Online], http://wordnet.princeton.edu/

[9] „Bing Translator”, [Online], http://www.microsoft.com/en-us/translator/developers.aspx

[10] „Twitter”, [Online], http://ro.wikipedia.org/wiki/Twitter

[11] „StatusNet”,[Online], http://en.wikipedia.org/wiki/StatusNet

[12] „Comparasion of micro-blogging services”,[Online], http://en.wikipedia.org/wiki/Comparison_of_microblogging_services

[13] „JSF”,[Online], http://en.wikipedia.org/wiki/JavaServer_Faces

[14] Elliot Rusty Harold , *Java Network Programming*,O’Reilly

[15] Alan Shalloway, James R. Trott, *Design Patterns Explained*, Pearson Education,2009

[16] „JSOUP”, [Online], http://jsoup.org/

[17] „MySql”, [Online], http://ro.wikipedia.org/wiki/MySQL

Anexa 1- Lista de figuri

Anexa 2- Lista de tabele