UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Colectarea si analizarea datelor de pe reddit

propusă de

Cristian-Andrei Ursu

Sesiunea: *iulie, 2019*

Coordonator științific

Conf. Dr. Anca Vitcu

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Colectarea si analizarea datelor de pe reddit

Cristian-Andrei Ursu

Sesiunea: *iulie, 2019*

Coordonator științific

Conf. Dr. Anca Vitcu

Avizat,

Îndrumător Lucrare de Licenţă

Titlul, Numele şi prenumele _____

Data _____ Semnătura _____

DECLARAŢIE privind originalitatea conţinutului lucrării de licenţă

Subsemntatul(a) ……

domiciliul în ……..

născut(ă) la data de ……., identificat prin CNP ……., absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de …… specializarea ……., promoţia ……., declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: _____

_____

_____elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, …… Semnătură student ……

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul „*Colectarea si analizarea datelor de pe reddit*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iaşi, să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, *data*

Absolvent *Prenume Nume*

_____

(semnătura în original)

Contents

Contributia proprie in dezvoltarea aplicatiei 3
Serverul si arhitectura acestuia 4
Construirea si antrenarea retelei neuronale. 9
Exemplificarea folosirii API-ului celor de la Reddit si a crawler-ului. 16
Exemplificarea si argumentarea folosirii 17
API-ului de la google. 17
Baza de date 18
Conexiunea cu baza de date 19
Clientul si arhitectura acestuia 22
Concluzii 25
Bibliografie 28

Introducere

RCruncher este o aplicatie web ce se ocupa cu colectarea si analiza datelor de pe reddit. Functionalitatea aplicatiei este impartita in doua arii: colectarea si analiza datelor pentru utilizatorii de reddit si colectarea si analiza datelor pentru asa numitele subreddits.

In ceea ce priveste utilizatorii, functionalitatea consta in a le oferi subreddits in care acestia ar putea fi interesati bazat pe activitatea anterioara in cadrul siteului si de a le indica utilizatorii cu interese asemanatoare. In ceea ce priveste postarile propriu-zise de pe reddit, postarea este analizata din punct de vedere semantic si din punct de vedere a analizei sentimentelor.

Din punct de vedere al originalitatii, reddit implementeaza unele dintre functionalitatile pe care le propune RCruncher dar nu ofera niciun fel de aplicatie externa spre acest scop. Ca aplicatii externe oferite de redidt, ce nu folosesc protocolul OAuth, majoritatea sunt wrappers peste API-ul expus de reddit.

Pentru a putea furniza solutii la problema propusa, aceea de a oferi recomandari si de a analiza anumite postari, am construit o aplicatie web ce consta in server-client ce serveste acest scop. Pentru inceput, se foloseste API-ul expus de catre reddit pentru a colecta datele ce tin de utilizatori, spre exemplu: posturile create si in ce subreddits si comentariile lasate si la ce posturi. Aceste date sunt mai tarziu salvate intr-o baza de date. Pe datele colectate, cu ajutorul unei retele neuronale, se ofera predictiile, recomandariile si utilizatorii inruditi. Pentru cea de a doua parte, se foloseste un crawler si API-ul extern pentru a obtine continutul propriu zis al postarilor dupa care datele sunt analizate de catre API-ul de limbaj natural al celor de la google, datele fiind salvate in baza de date anterior mentionata.

Ca si capitole ale acestei lucrari vor fi abordare urmatoarele:

- Serverul si arhitectura acestuia.
- Construirea si antrenarea retele neuronale.

- Exemplificarea folosirii API-ului celor de la Reddit si a crawler-ului.
- Exemplificarea si argumentarea folosirii API-ului de la google.
- Baza de date.
- Conexiunea cu baza de date.
- Clientul si arhitectura acestuia.

Contributia proprie in dezvoltarea aplicatiei

In dezvoltarea aplicatiei, asupra partii de server a fost concentrate mare parte din efort. Astfel in ceea ce priveste serverul, singura dependenta este NestJs, tot ce tine de designul architectural al serverului, controlerele si metodele respective acestora, domeniul si arhitectura lui, serviciile folosite, toate sunt aproape in totalitate munca proprie.

In ceea ce priveste reteaua neuronala, alegerea unei retele potrivite pentru setul de date disponibil, pregatirea mediului favorabil, alegerea si pregatirea datelor pentru antrenare, alegerea optiunilor pentru creare, documentarea asupra functiei de distanta si alegerea acesteia, integrarea si adaptarea retelei sunt munca proprie, depinzand totusi de un pachet extern ce reprezinta o baza a retelei neuronale de tip Kohonen.

Designul bazei de date, crearea tabelelor, a dependentei dintre acestea si a operatiilor efectuate asupra acesotra sunt dependente de tooluri externe, insa contributia proprie este semnificativa.

In ceea ce priveste colectarea datelor si analiza cu instrumentele de la google: instantierea apelurilor, validarea si prelucrarea datelor atat primite cat si trimise, reprezinta principala contributie in acest domeniu.

Pe partea de client, pentru partea de vizualizare sunt folosite unele librarii externe insa pentru care este necesara partea de conexiune cu serverul ce consta in serviciile de la nivelul clientului si prelucrarea si validarea datelor.

Serverul si arhitectura acestuia

Partea de server este construita dupa principiul dupa DDD(domain-driven design), astfel incat concentrearea este directionata catre domeniu si logica acestuia. Pe partea de server, am optat sa folosesc node.js si typescript in detrimentul PHP-ului.

Printre motive se enumera:

- Folosirea aceluiasi limbaj de programare la niveul clientului si al serverului
- Moulele incarcate de catre node sunt descarcate si dupa initializate, ulterior fiind disponibile constant
- Se lucreaza mai usor cu fisiere de dimensiuni mari
- Caracterul strongly-typed al typescriptului fata de javascript
- Numeroasele module oferite de catre Node.js
- Debugging in real time
- Crawlere folosite primesc raspunsuri HTML randate total

Serverul este construit cu ajutorul lui NestJS, un framework specializat pe constructia de servere, care poate fi descris cel mai bine ca un wrapper peste Express. Hostarea acestuia are loc pe portul 3000 si reprezinta punctul de start al aplicatiei.

```
async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  app.enableCors();
  await app.listen(3000);
}
bootstrap();
```

Fig. 1 *Hostarea aplicatiei*

Aplicatia contine un singur modul in care se intializeaza dependentele acesteia, de exemplu: modulul de CQRS, conexiunea cu baza de date, controllerele si serviciile de care depinde modulul.

```
@Module({
  imports: [TypeOrmModule.forRoot(), CqrsModule],
  controllers: [RedditUsersController, RedditPostsController],
  providers: [
    ...CommandHandlers,
    ...QuerryHandlers,
    RedditDataService,
    TextEnchancerService,
    NaturalLanguageService,
    CrawlerService,
    PageContentRequester,
  ],
})
export class AppModule {
  constructor(private readonly connection: Connection) { }
}
```

Fig. 2 *Modulul principal al aplicatiei si dependentele acestuia*

API-ul expus de catre server este unul de tip REST. Acesta expune doua controlere:

- Pentru userii de reddit

```
@Controller('reddit-users')
```

- Pentru posturile de pe reddit

```
@Controller('reddit-posts')
```

Fig. 3 *Decoratorul folosit pentru a crea cele doua controlere*

Majoritatea metodelor expuse definite in controller-ul pentru utilizatori iar amandoua controller-ele contin metode doar de tipul POST si GET.

Pentru controller-ul ce corespunde utilizatorilor urmatoarele metode sunt expuse:

- Create, de tipul POST care primeste numele unui utilizator de reddit, creeaza entitatea corespunzatoare si o salveaza in baza de date.

- GetUserData, de tipul GET, care returneaza din baza de date un utilizator si toate datele legate de acesta.
- PredictUser, de tipul GET, care indica pozitia utilizatorului in cadul retelei neuronale.
- GetUserTopics, de tipul GET, care returneaza topicurile unui user bazate pe comentariile lasate de acestea in scopul construirii unui nor de cuvinte.
- GetTrainingSet, de tipul GET, care returneaza pozitiile tuturor utilizatorilor in cadrul retelei neuronale, utilizatori ce au facut parte din setul de initial de antrenament.
- RefreshComments, RefreshSubmitted, RefreshTopics, de tipul GET, care ii comunica serverului ca pentru un anumit utilizator se doreste innoirea respectivelor date.
- GetApplicationData, de tipul GET, prin care se returneaza statisticile aplicatiei.

Pentru controller-ul ce corespunde postarilor urmatoarele metode sunt expuse:

- CreatePost, de tipul POST, ce primeste URL unui post de pe reddit, il trimite catre analiza si ulterior catre salvarea lui in baza de date impreuna cu analiza corespunzatoare.
- GetPostWithData, de tipul GET, ce primeste URL unui post de pe reddit si returneaza date despre acesta impreuna cu analiza corespunzatoare a postarii.

Fiecare serviciu de la nivelul serverului, mai putin reteaua neuronala, este de tip Injectable, ceea ce inseamna ca, similar unui singleton, fiecare serviciu Injectabil v-a exista intr-o singura instantiere pe parcursul unei sesiuni de viata a serverului.

Toate functiile asincrone de pe server, cum ar fi aducerea datelor de pe reddit sau returnarea unor date la nivelul API-ului, folosesc tipul de date Observable din libraria Rx.js, ce reprezinta o inbunatatire de la Promise, sau patternul async, await ce sunt

deseori folosite in aplicatii bazate pe node.js atunci cand este nevoie de tratarea asincronismului.

Pentru operatiilor asupra bazei de date se foloseste sablonul CQRS, care vizeaza segregarea comenzilor de interogari. In acest scop, pentru fiecare interogare sau comanda exista cate un handler si un query sau respectiv command. Astfel clasa command sau query create contine datele necesare procesarii, iar in clasa handler are loc procesearea propriu zisa a comenzilor. De exemplu, pentru a colecta comentariile unui utilizator de reddit in baza de date, au fost create urmatoarele clase:

- AddCommentsForFirstTimeRedditUserCommand
- AddCommentsForFirstTimeRedditUserHandler

Astfel, comanda contine numele utilizatorului reddit pentru care se colecteaza comentariile, care este unic:

```
export class AddCommentsForFirstTimeRedditUserCommand {
  constructor(public readonly redditUser: RedditUserModel) { }
}
```

Fig. 4 *Comanda pentru adaugarea unui utilizator*

Clasei din urma ii revine responsabilitatea de a colecta datele, de ale prelucra, si de ale salva in baza de date. Pentru interogari structura este asemanatoare.

Pentru a executa aceste comenzi si interogari, sunt folosite urmatoarele doua servicii injectabile: CommandBus si QueryBus, folosite in ambele controlere.

Chiar daca, un asemenea bus pare sa aiba la inceput doar responsabilitatea de a lega comenzile si interogarile de handlere, utilitatea acestuia este mult mai larga. Acesta valideaza datele din interiorul unei comenzi, incapsuleaza handlerul intr-o tranzactie a bazei de date si pastreaza ordinea comenzilor/interogarilor, respectand astfel sablonul lantului de responsabilitate.

In continuare voi prezenta o diagrama UML, usor simplificata, ce reprezinta arhitectura serverului.

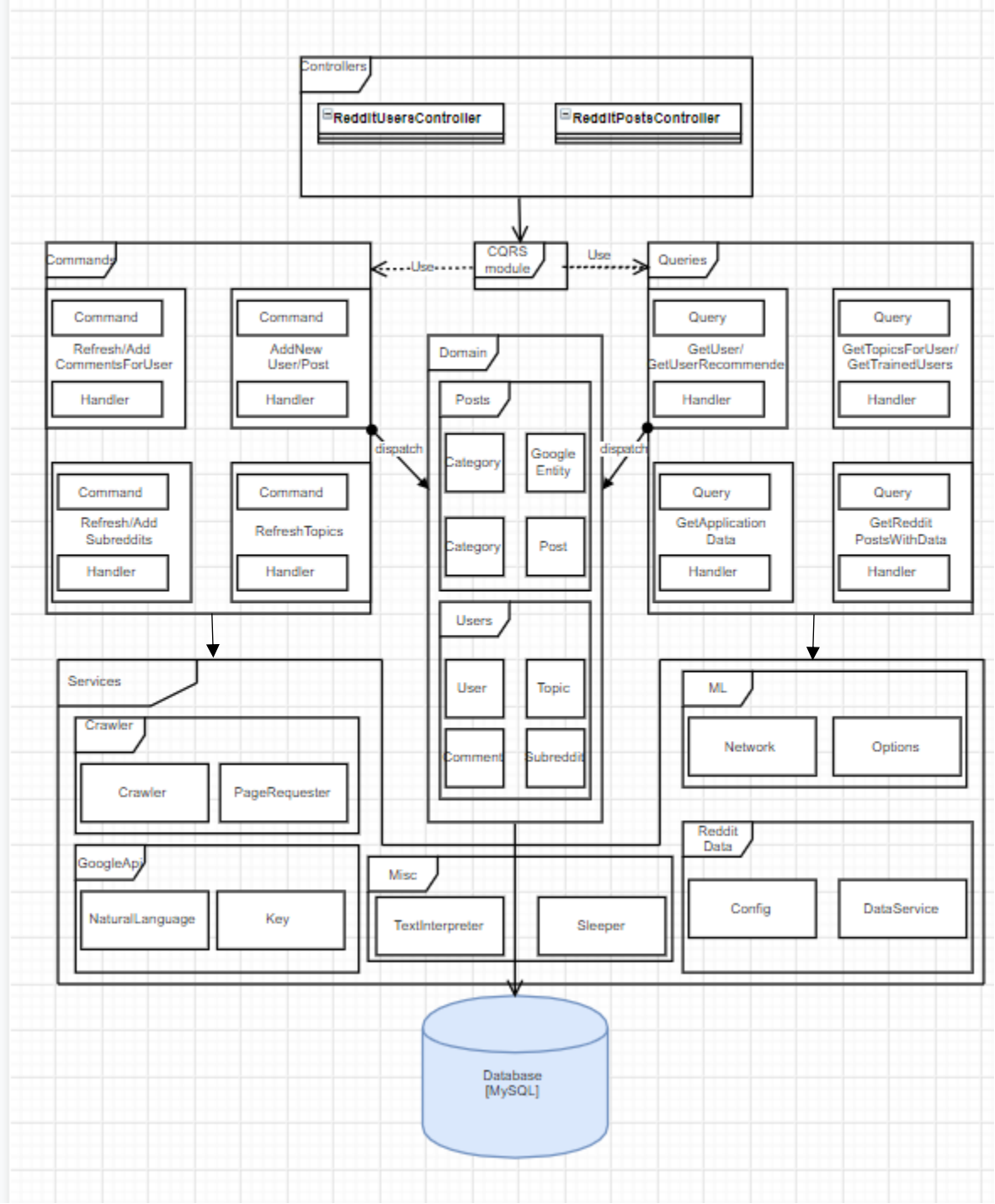


Fig. 5 *Diagrama arhitecturii serverului*

Construirea si antrenarea retelei neuronale.

In ceea ce priveste reteaua neuronala am optat pentru una de tip SOM(self-organizing map)/ Kohonen din mai multe motive:

- Reprezinta un tip de retea neuronala cu invatare nesupervizata.
- Este disponibila alegerea unei functii de distanta customizata.
- Tipul de invatare oferit de retea este : invatare competitive in opozitie cu invatarea bazate pe corectare de erori. Am optat pentru aceasta varianta deoarece cu greu se poate vorbi de erori in cazul de recomandari sau predictii.
- Este indicata in cazul vrem sa vizualizam predictii pentru date foarte mari. Pentru o retea neuronala construita din 100 de utilizatori si 633 de sub topics reteaua neuronala, salvata sub forma unui fisier de tipul .JSON ajunge la o marime de aprox. 85 MB, in conditiile in care, in martie 2019, reddit a inregistrat 542 de milioane de utilizatorii.

Chiar daca este privita ca un serviciu, am ales sa nu ii ofer retelei neuronale caracterul de clasa Injectabila, datorita timpului lung de incarcare si antrenare a acesteia, ci mai degraba sa o instantiez la crearea controllerului pentru utilizatori de care este strans dependenta. Astfel reteaua va fi incarcata la pornirea serverului si nu cand va fi nevoie de ea propriu-zis.

Setarile dupa care este construita reteaua sunt urmatoarele, urmand sa fie explicate in urmatoarele cateva paragrafe:

```
export class KohonenOptions {
    public fields;
    public iterations = 100;
    public learningRate = 0.1;
    public xValue = 100;
    public yValue = 100;
```

Fig. 6 *Optiunile retelei neuronale*

Astfel, reteaua neuronala consta in 100*100 de celule. Iteratiile sunt de asemenea 100, ceea ce inseamna ca se va itera de 100 * Lungimea datelor de antrenament in faza de antrenare a retelei. Coeficentul de invatare este 0.1, acesta fiind transmis mai departe catre algoritmul ce se ocupa cu invatarea propriu-zisa

Astfel, pentru a putea antrena reteaua sau pentru a putea produce o recomandare/prezicere este nevoie de asesarea datelor in urmatorul format: pentru fiecare utilizator este creat un json cu urmatorul tip de inregistrari: numeSubreddit : [0, PMAX], unde PMAX reprezinta numarul maxim de postari facute de orice utilizator in acel subreddit. Astfel pentru fiecare utilizator for fi parcuse toate subredditurile si create inregistrarile necesare. Daca un utilizator nu a postat niciodata intr-un subreddit, activitatea acestuia in subredditul corespunzator este considerate nula.

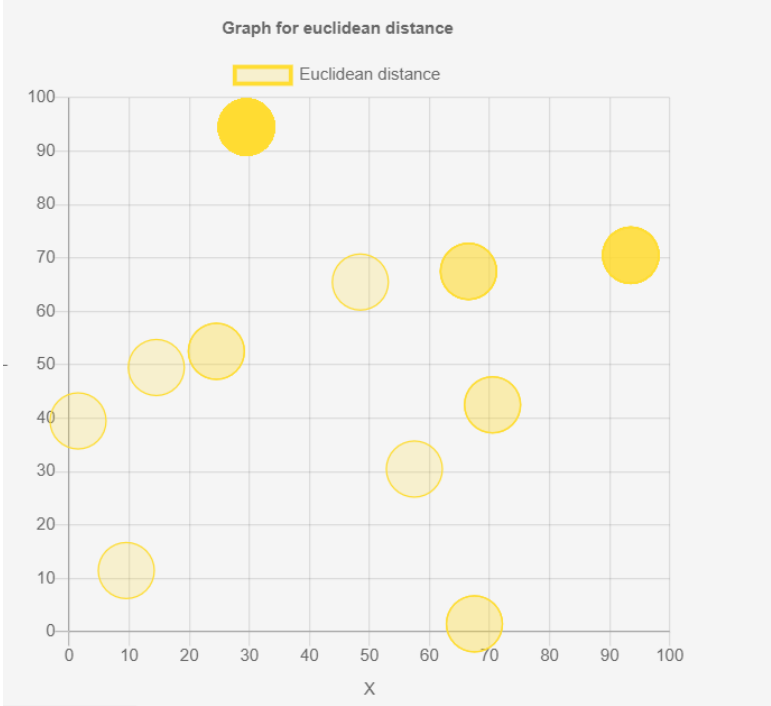
```
public buildUserTrainingSet(currentUser: RedditUserEntity, allSubbreddits: UserSubredditEntity[]): Observable<any> {
    const newPromise = new Promise(async (resolve) => {
        const userDataSet = {};
        for (const subreddit of allSubbreddits) {
            const foundSubredditForUser = await UserSubredditEntity.findOne(
                {
                    where: { origin: subreddit.origin, owner: currentUser },
                },
            );
            if (this.baseFields.includes(subreddit.origin)) {
                if (foundSubredditForUser !== undefined) {
                    userDataSet[subreddit.origin] = subreddit.numberOfAppearances;
                } else {
                    userDataSet[subreddit.origin] = 0;
                }
            }

        }
        resolve(userDataSet);
    });
    return from(newPromise);
}
```

Fig. 7 *Construirea datelor pentru un singur utilizator*

In continuare vom aborda functia de distanta dintre utilizatori si procesul alegerii acesteia.

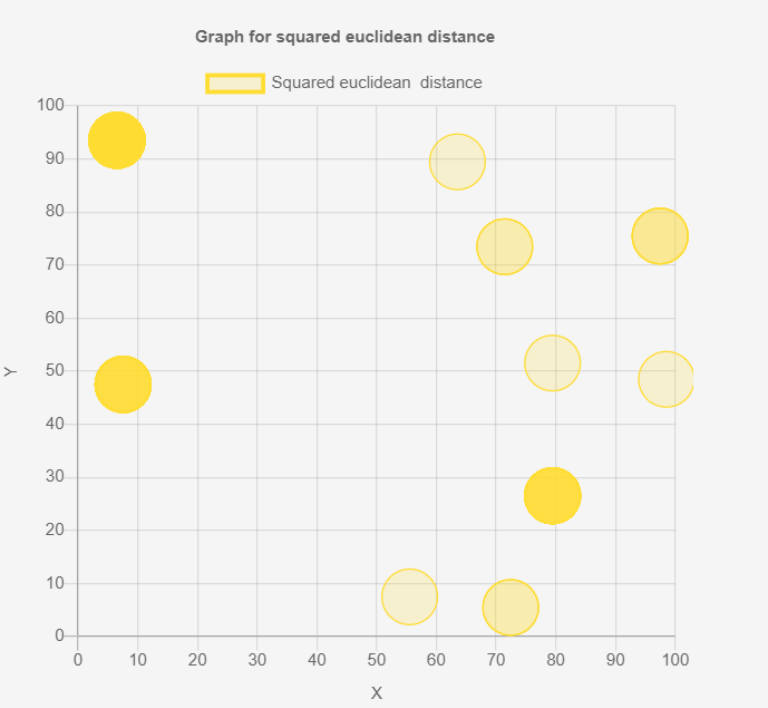
In urmatoarele grafice este evidentiata distrbutia a 100 de utilizatori cu 633 de subreddituri, singura schimbare fiind functia de distanta.



$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Fig. 8 *Distributia setului de antrenament pentru distanta euclidiana*

In cazul distantei euclidiene putem observa o dispersie inadecvata datelor, formandu-se doar doua clustere fixe, unul dintre ele fiind pentru utilizatorii pentru care nu se inregistreaza niciun fel de postare in cadrul redditului.



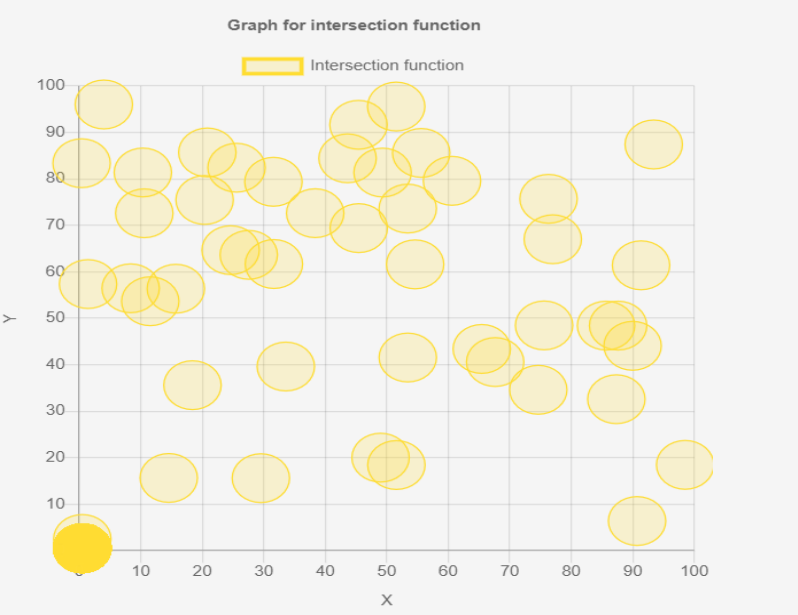
$$\sum_{i=1}^{n}(q_i - p_i)^2$$

Fig. 9 *Distributia setului de antrenament pentru patratul distantei euclidiene*

In cazul patratului distantei euclidiene, se poate constata aparitia unui al 3 cluster, tot fix, dispersia datelor in grafic fiind inca inadecvata.

In ciuda faptului ca aceste doua distante euclidiene reprezinta distantele de baza ce se folosesc in cadrul multor algoritmi de invatare automata, aici se poate observa cu usurinta gradul ridicat de inadecvare a acestora din constructia datelor pentru fiecare utilizator.

In continuare am apelat tot la o functie de baza, si anume, intersectia intre doua inregistrari.



$$\sum_{i}^{n} \min(q_i, p_i)$$

Fig. 10 *Distributia setului de antrenament pentru distanta calculata cu ajutorul intersectiei*

In cazul intersectiei se observa o dispersie mai buna a datelor, un singur cluster fix, si altele 2-3 clustere mai dispersate.

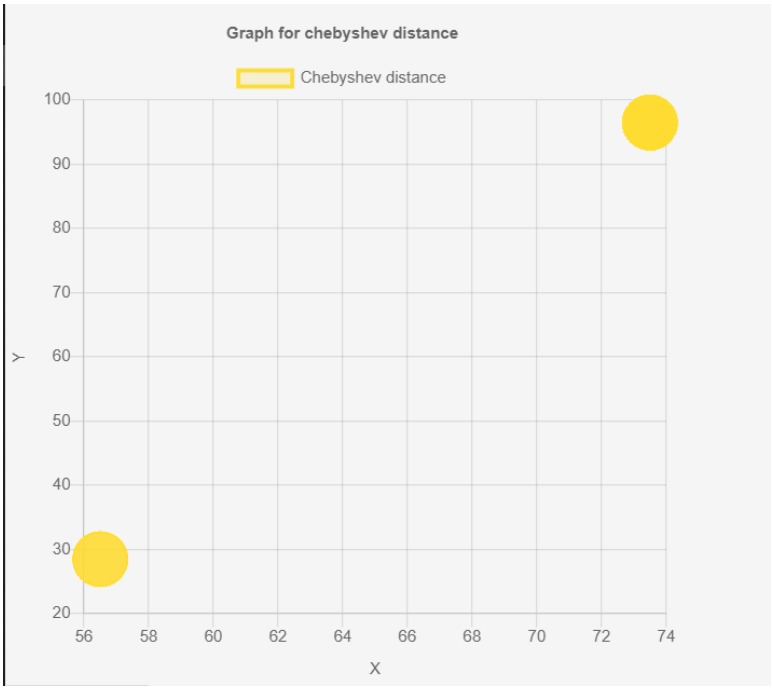


Fig. 11 *Distributia setului de antrenament pentru distanta chebyshev*

$$max_i(q_i - p_i)$$

Cazul distantei Chebyshev este unul nefavorabil formandu-se doar doua clustere fixe la extremitati, din pricina modului in care se calculeaza distanta.

Urmatoarea distanta folosita a fost distanta statistica, rezultatele fiind destul de satfisfacatoare.

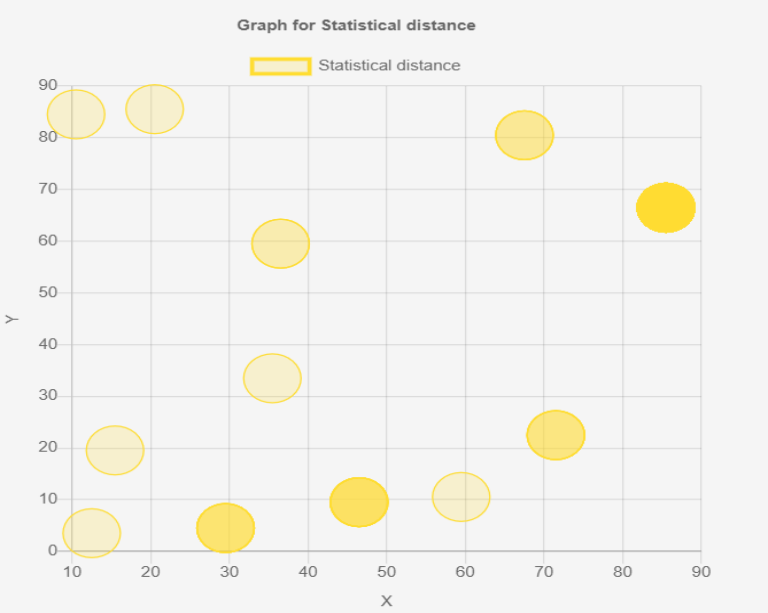


Fig. 12 *Distributia setului de antrenament pentru distanta statistica*

$$\sum_{i=0}^{n} (p_i - q_i)^2 \Big/ (p_i + q_{i)}$$

In continuare am folosit distanta Canberra insa cu mici sane de success deoarece aceasta este bazata pe distantele manhattan, ne mai fiind folosita semnificativ in invatare automata.

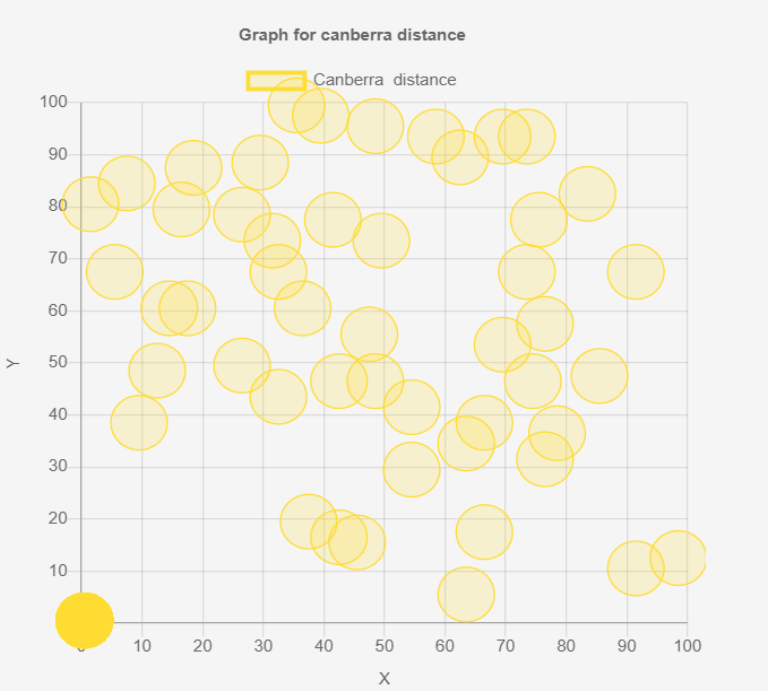
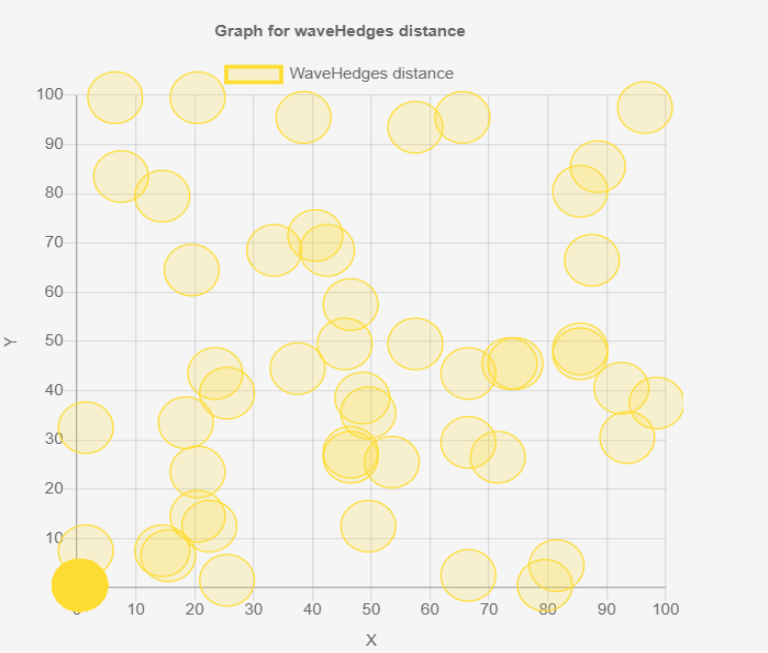


Fig. 13 *Distributia setului de antrenament pentru distanta canberra*

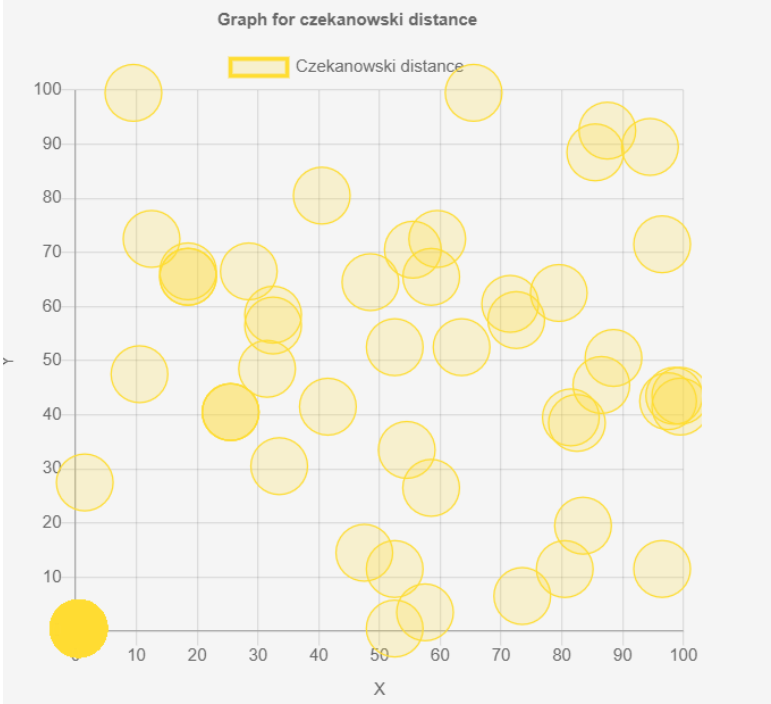
$$\sum_{i=0}^{n} p_i - q_i \Big/ |p_i| + |q_i|$$

Urmatoarele doua distante sunt inrudite, amandoua facand parte din familia distantelor ce folosesc intersectia.



$$\sum_{i=0}^{n} 1 - \frac{\min(p_i, q_i)}{\max(p_i, q_i)}$$

Fig. 14 *Distributia setului de antrenament pentru distanta WaveHedges*



$$\frac{\sum_{i=0}^{n} |p_i - q_i|}{\sum_{i=0}^{n} (p_i + q_i)}$$

Fig. 15 *Distributia setului de antrenament pentru distanta czekanowski*

In finala distanta alesa pentru reteaua neuronala a fost cea Czeakanowski, din pricina distributiei datelor si buna clusterizare a acestora.

Astfel a fost creata reteaua Kohonen, cu ajutorul careia se vor oferi recomandari si preziceri.

Reteaua, optiunile pentru aceasta si datele din setul de antrenament au fost folosite o singura data, dupa care reteaua a fost salvata la nivelul serverlui sub format json, urmand ca la fiecare pornire a acestuia, reteaua sa fie reincarcata din fisier si nu inceperea procesului de la capat.

```
@Controller('reddit-users')
export class RedditUsersController {
    private kohonenNetwork = new KohonenNetwork();
    constructor(private readonly commandBus: CommandBus, private readonly queryBus: QueryBus) {
        this.kohonenNetwork.loadNetwork().subscribe(() => {
            console.log('Trained');
            sleeper(10000).then(() => {
                this.kohonenNetwork.trainNetwork();
            });
        });
    }
```

Fig. 16 *Incarcarea retelei neuronale din fiser*

Exemplificarea folosirii API-ului celor de la Reddit si a crawler-ului.

Avand in vedere ca colectarea datelor reprezinta un pas de baza din aplicatia propusa, in continuare voi exemplifica modul in care sunt obtinute datele si API-ul corespunzator al siteului reddit.

Deoarece reddit considera ca datele precum comentariile sau posturile create ar trebui sa fie publice, nu a fost nevoie de urmarea protocolului OAuth pentru a obtine aceste date. Pentru utilizatori a fost folosit numai API-ul folosit de catre reddit. Datorita volumului de date existent pe platforma, am cosiderat ca cele mai semnificative date se

afla in ultimele 6 luni de activitate a utilizatorului. Astfel, cand un utilizator este creat, cele mai vechi comentarii si postari dateaza cu maxim 6 luni in urma momentului actual.

```
private numberOfMonthsToFetch: number = 6;
```

```
const boundaryDate = new Date();
boundaryDate.setMonth(boundaryDate.getMonth() - this.numberOfMonthsToFetch);
const unixDate: number = parseInt((boundaryDate.getTime() / 1000).toFixed(0), 10);
do {
    await sleeper().then(() => this.getOneBatchOfComments(
        redditCommentOwner, afterId).then((data) => { commentDate = data[0]; afterId = data[1]; }));
} while (commentDate > unixDate);
```

Fig. 17 *Schema popularii tabelului comentariipentru ultimele 6 luni de activitate*

Aceste date nu sunt singurele ce pot fi colectate de catre aplicatie, fiind posibil ca, ulterior, dupa adaugarea unui user, comentariile sau postarile acestuia sa fie actualizate pana la momentul current.

Pentru datele ce tin de postarile de pe reddit este folosit atat crawlerul cat si API-ul.

Datele propriu-zise dintr-o postare sunt obtinute cu ajutorul apelurilor catre interfata propusa de catre reddit iar crawlerul incearca sa gaseasca si sa analizeze alte postari ce sunt poate accesibile din linkul curent.

Exemplificarea si argumentarea folosirii API-ului de la google.

Daca in ceea ce priveste invatarea automata in cardrul analizei utilizatorilor de reddit folosim o retea neuronala, pentru a extrage date ce tin de limbajul natural, vom folosi API-ul Google pentru.

Acest API foloseste pentru autorizare doar un singur key ce se instantiaza in cadrul consolei google.

Toate apelurile catre google folosesc aceasta cheie privata. Cele trei apeluri catre google sunt folosite pentru:

- Analiza entitatilor.
- Analizarea sentimentelor.
- Si clasificarea textului.

Datele sunt validate si inainte sa fie trimise si dupa ce au fost primite de la modulul de invatare automata a google-ului.

Baza de date

Baza de date este o baza de date relationala, mai exact MySQL si consta in 8 tabele ce vor fi mentionate sumar:

- reddit_user_entity, ce se ocupa cu stocarea utilizatorilor si datelor despre acestia.
- reddit_comment_entity, care stocheaza pentru fiecare utilizator comentariile colectate.
- reddit_topic_entity, in care se salveaza topicurile extrase din comentarii.
- user_subreddit_entity, se ocupa cu stocarea postarilor create de fiecare utilizator
- reddit_post_entity, in care se salveaza posturile cu datele asociate acestora
- google_natural_language_sentence, in care se salveaza datele primite de la google in legatura cu propozitile componente ale postarii
- google_natural_language_entity, stocheaza entitatile din postare, entitati primte de la google
- google_natural_language_category, in care se salveaza categoriile din care face parte o postare, categorii obtinute dupa apelarea modulului de limbaj natural al google-ului.

Conexiunea cu baza de date

Pentru conexiunea cu baza de date a fost folosit un object-relational mapping, si mai exact typeorm. Setarile cu care a fost creata conexiunea sunt urmatoarele:

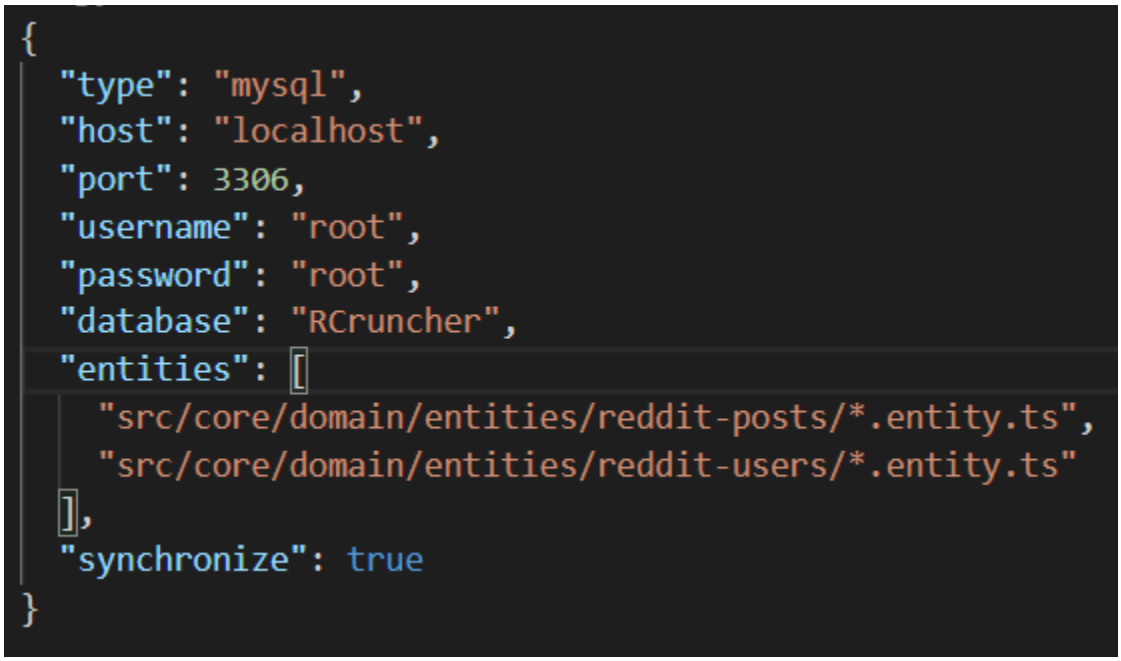


Fig. 18 *Setarile pentru conexiunea cu baza de date*

Baza de date se numeste RCruncher iar tabelele sunt reprezentate de catre entitatile aflate in cele doua foldere: “entities/reddit-posts” si “entities/reddit-users”. Fiecare entitate are decoratorul de Entity si extinde BaseEntity, comportandu-se astfel, cand este nevoie, ca un repository pattern.

Entitatile sunt impartite in doua categorii: cele care tin de utilizatori si cele care tin de postari. Voi incepe cu cele ce tin de postari.

RedditPostEntity, este entitatea ce se ocupa cu postarile de pe reddit. Cele mai importante campuri sunt cele ce retin textul postarii, daca postarea a fost procesata anterior, magnitudinea si scorul sentimentului. De asemenea in acest tabel se tin cheile primare si ale celorlate trei entitati ce au legatura cu analiza Google. Pentru a conecta cele trei entitati de RedditPostEntity se foloseste o relate de tipul OneToMany.

```
@OneToMany(type => GoogleNaturalLanguageSentence, googleNaturalLanguageSentence => googleNaturalLanguageSentence.owner)
sentences: GoogleNaturalLanguageSentence[];

@OneToMany(type => GoogleNaturalLanguageEntity, googleNaturalLanguageEntity => googleNaturalLanguageEntity.owner)
entities: GoogleNaturalLanguageEntity[];

@OneToMany(type => GoogleNaturalLanguageCategory, googleNaturalLanguageCategory => googleNaturalLanguageCategory.owner)
categories: GoogleNaturalLanguageCategory[];
```

Fig. 19 *Relatiile OneToMany din clasa RedditPostEntity*

GoogleNaturalLanguageSentence este necesar stocarii propozitiilor si sentimentul asociat acestora. GoogleNaturalLanguage category este folosit pentru a putea pastra categoriile din care face parte fiecarte text iar GoogleNaturalLanguageEntity este folosit pentru a salva entitatile asociate textului colectat.

Pentru a face legatura cu RedditPostEntity se foloseste o relatie de tipul OneToMany pentru toate cele trei clase mentionate anterior.

```
@ManyToOne(type => RedditPostEntity, redditPostEntity => redditPostEntity.sentences)
owner: RedditPostEntity;
```

Fig. 20 *Relatie de tip ManyToOne din clasa GoogleNaturalLanguageSentence*

A doua categorie de entitati este cea care tine de utilizatori si de datele colectate ale acestora. Entitatea principala este RedditUserEntity si stocheaza mai multe date esentiale rularii aplicatiei: numele utilizatorului, daca utilizatorul face sau nu din setul de antrenare a retelei neuronale, pozitia utilizatorului in cadrul retelei neuronale, id-urile comentariilor si a posturilor create cat si vectori de chei straine pentru urmatoarele trei entitati: RedditCommentEntity, RedditTopicEntity si UserSubRedditEntity. Relatia folosita este OneToMany.

RedditCommentEntity este folosita pentru salvarea comentariilor in tabelul corespunzator. Printre datele pastrate se stocheza textul acesteia, daca aceasta a fost parsata din punct de vedere a extragerii subiectelor, id-ul de reddit si cheia primara a

utilizatorului. Chiar daca la momentul actual comentariile sunt folosite doar pentru a extrage subiectele si ulterior nu mai sunt relevante, am optat sa le pastrez in baza de date in vederea dezvoltarii ulterioare a aplicatiei.

RedditTopicEntity este folosita pentru stocarea subiectelor extrase din comentarii si a aparitiei. Acestea sunt folosite pentru a construi diagrama de tipul polar-area de pe partea de client.

In final, UserSubRedditEntity, este folosita in vederea stocarii frecventei unui user de reddit in cadrul unui subreddit. Aceste entitati sunt folosite in antrenarea retelei neuronale pentru setul de antrenament si pentru predictiile si recomandarile oferite utilizatorilor ce nu fac parte din setul de antrenament. De asemenea legatura intre aceste ultime trei entitati si entitatea ce se ocupa cu utilizatorii se face printr-o relatie de tipul ManyToOne.

In privinta motivarii folosirii acestui pachet, typeorm, aduc urmatoarele argumente: Este asemanator cu alte mari orm-uri de acest fel precum Hibernate sau Entity Framework, astfel codul poate fi citit si usor inteles de un dezvoltator ce nu este familiar cu typeorm. Este actualizat mereu in functie de versiunea de javascript/typescript curenta. Este unul dintre cele mai prolifice orm-uri pentru NodeJs. Suporta atat baze de date relationale cat si non-relationale.

Clientul si arhitectura acestuia

Pentru partea de client, am optat sa folosesc arhitectura SPA(singe-page application) deoarece chiar daca volumul datelor este ridicat pe partea de server, pe partea de client trebuie doar reprezentate visual acestea.

Limbajul folsit este tot typescript, coincizand astfel cu cel de pe server. Frameworkul folosit este Angular, framework ce reprezinta o solutie viabila pentru aplicatii de tip SPA.

Astfel, partea de client este formata din cateva pagini ce vor fi prezentate in continuare:

- Pagina de home, contine datele aplicatiei, cati utilizatori, postari, comentarii etc au fost colectate. De asemenea, in pagina de home se pot adauga noi utilizatori si postari.

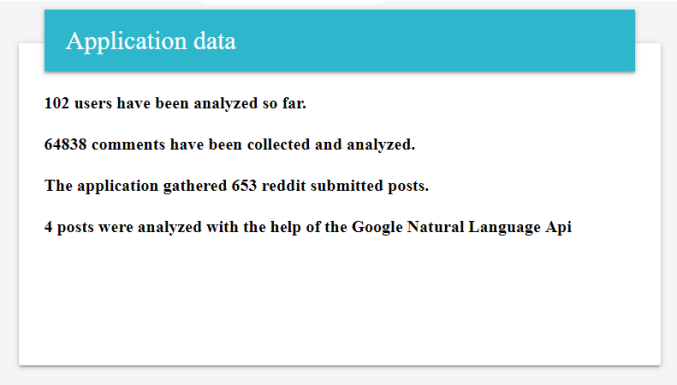


Fig. 21 *Statisticile aplicatiei*

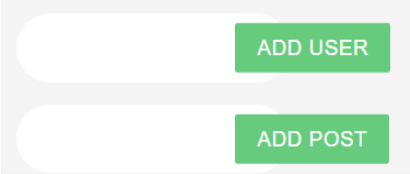


Fig. *22 Butoane pentru adaugare de entitati*

- Pagina de network, contine mijloacele de a viziona datele utilizatorilor, reteaua neuronala, recomandarile pentru utilizatorul ales si statistica subiectelor folosite.

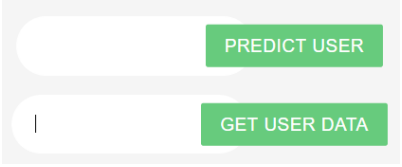


Fig. 23 *Butoane pentru vizualizarea datelor ce tin de utilizator*

Nodurile ce fac parte din setul initial de antrenament sunt reprezentate cu galben iar cele ce sunt adaugate pentru predictie sunt reprezentate cu albastru deschis.

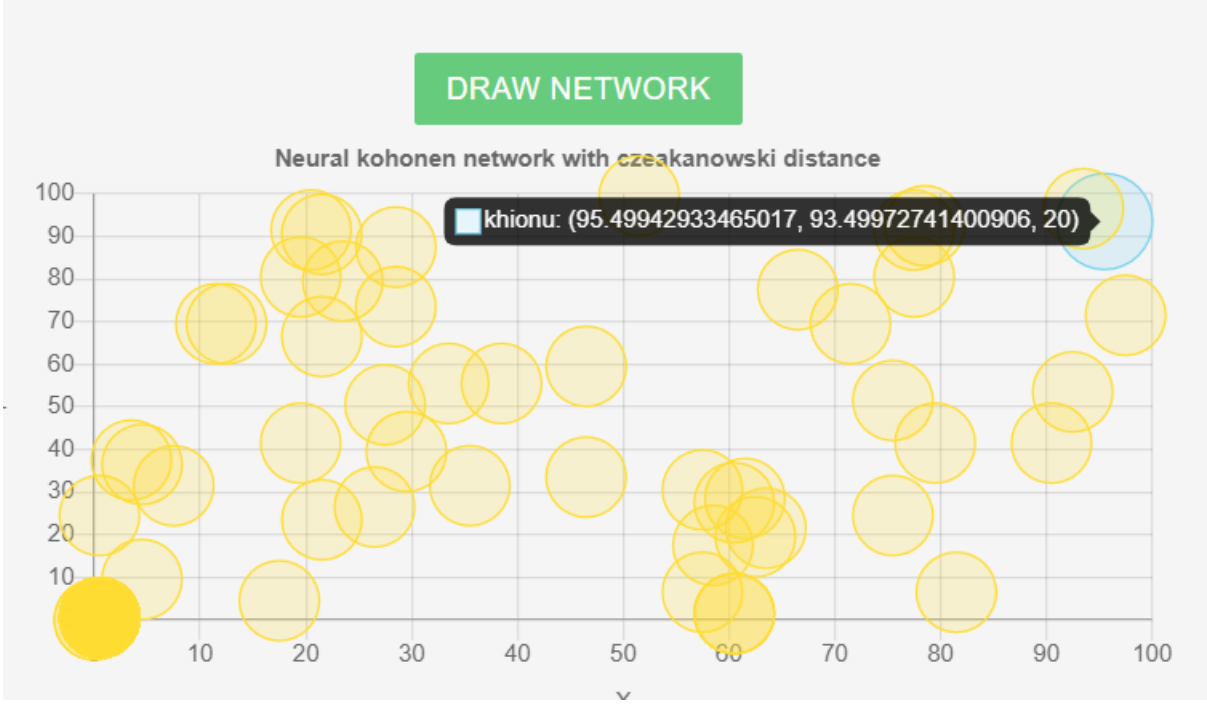


Fig. 24 *Reprezentarea predictiei pentru utilizatorul khionu*

In loc sa folosesc un nor de cuvinte, ce este mult prea intalnit si folosit, am optat sa folosesc un polar-area.

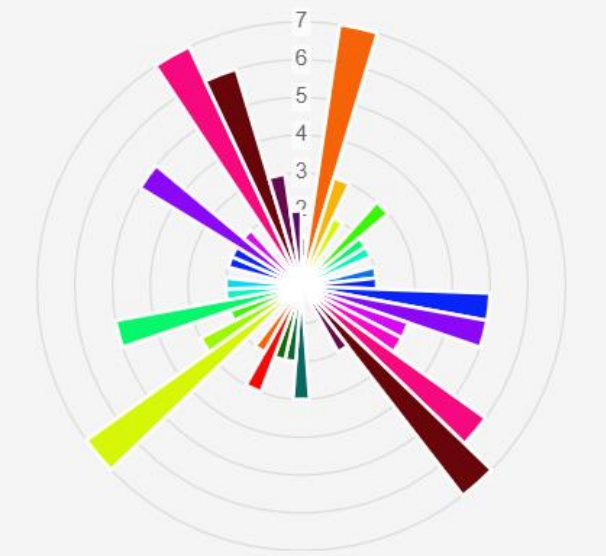


Fig. 25 *Graficul de tip polar-area pentru subiectele unui utilizator*

Ultimul element al paginii reprezinta partea de recomandari de subreddtis si utilizatori asemanatori asezate sub forma a doua liste.

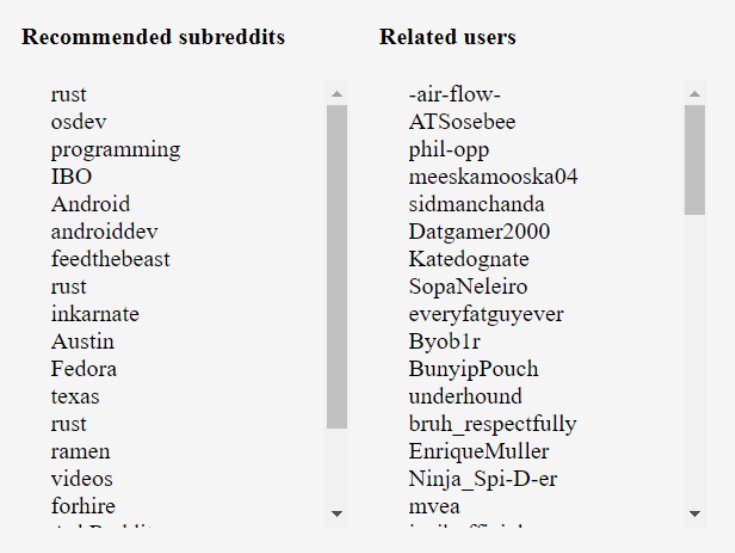


Fig. 26 *Tabelele ce contin recomandarile pentru utilizatori*

- Ultima pagina este cea care corespunde postarilor si este formata dintr-un container ce contine textul postariii si analiza postarii si un buton pentru a selecta postarea dorita.

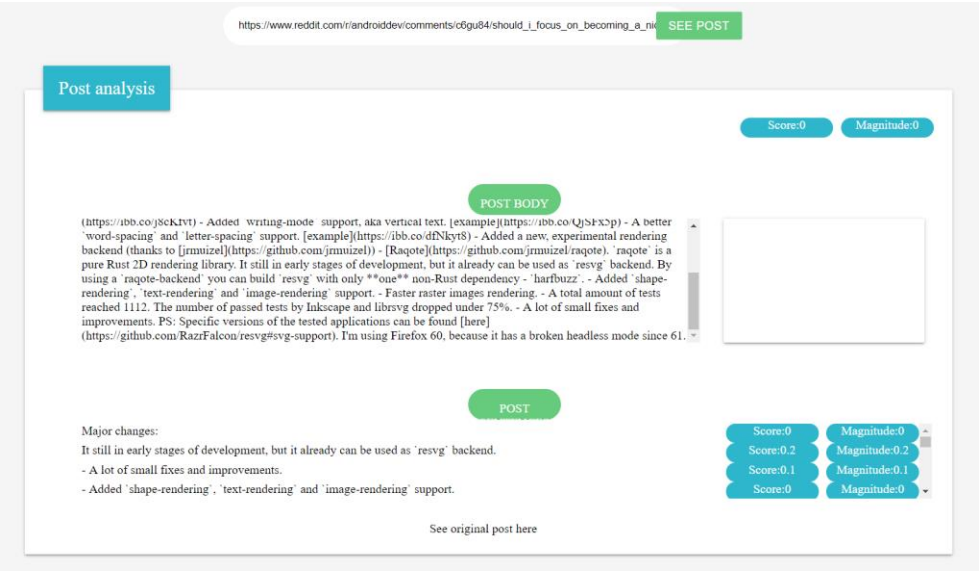


Fig.27 *Pagina pentru postari si planul general al acesteia*

In partea de sus a paginii se afla un buton, care apasat, initiaza o cerere catre server pentru a aduce datele asociate unui url, url care este unic pentru fiecare postare.

Pentru partea de vizualizare unui postari este present un container format din 3 mini-containere. Primul este textul actual al postarii, urmat, imediat la dreapta de containerul in care se afla categoriile din care face parte postarea si nivelul de incredere in aceasta analiza.

In ultimul container se afla propozitiile extrase din text si scorul sentimentului si magnitudinea acestuia. In partea de jos a containerului se afla un link catre postarea initiala de pe reddit.

De asemenea in componenta fiecarei pagini se poate gasi bara de menu a aplicatiei care arata astfel:

Fig. 28 *Bara de navigare*

In continuare voi prezenta un use-case pentru folosirea partii de client a aplicatiei:

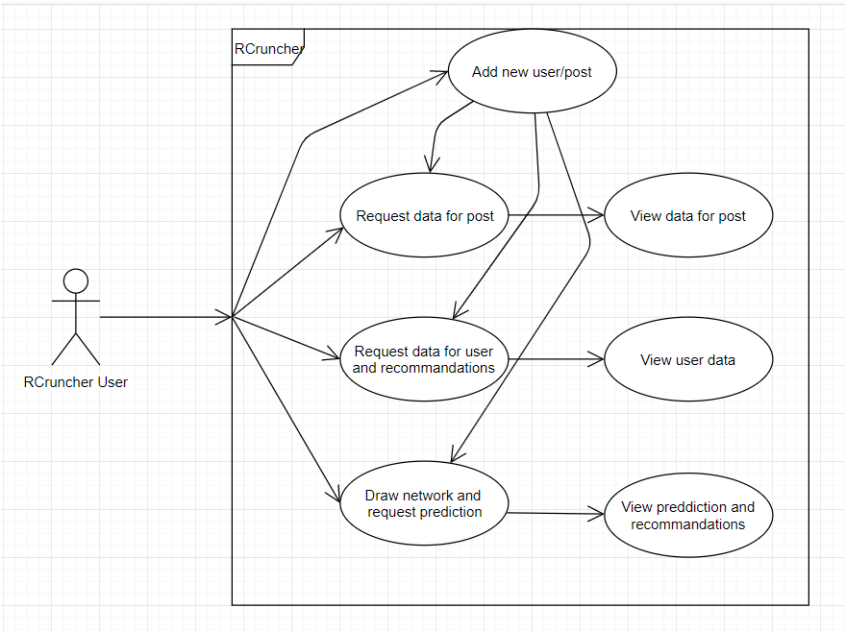


Fig. 30 *Use-case pentru folosirea aplicatiei*

Singurul servicu existent in momentul de fata este cel ce se ocupa cu crearea cererilor pentru partea de sever. Acesta este folosit de toate cele trei pagini si este de tip Injectable.Toate cererile catre server sunt de tip AJAX, folosindu-se formatul de date JSON.

Urmatoarea diagrama reprezinta arhitectura clientului:

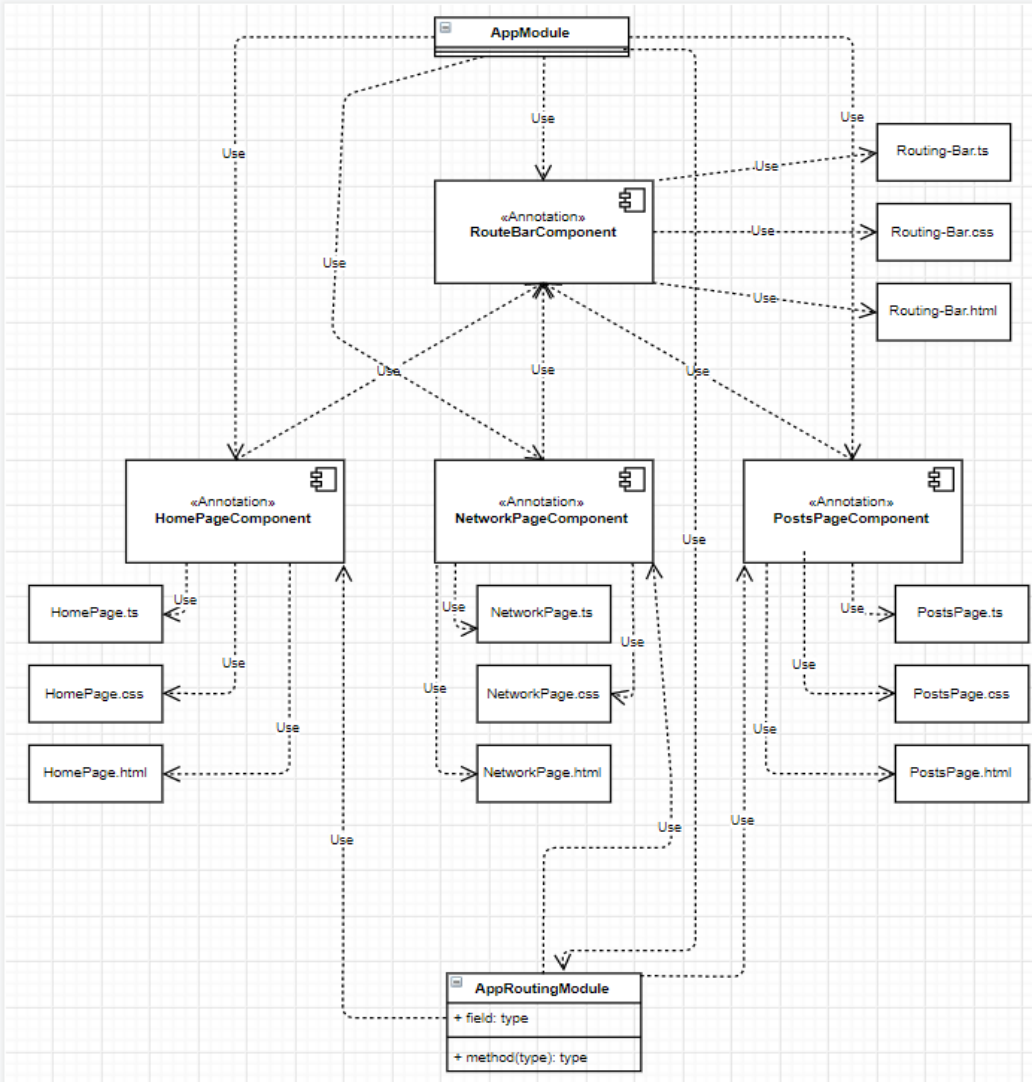


Fig. 31 *Arhitectura clientului*

Concluzii

In concluzie, aplicatia este cross-platform din cauza caracterului de pagina web a acesteia. Functionaliteatea poate fi imbunatatita prin colectarea si altor date cu caracter mai sensibil de pe reddit, cum ar fi aprecierile si voturile negative pentru a imbunatati acuratetea retelei. Pe langa acestea, in reteua neuronala poate fi introdusa si analiza sentimentelor, si deci, o retea mai precisa. Astfel pe partea de server se poate implementa un process de inregistrare si autentificare cu ajutorul JWT, dupa care, contul poate fi conectat cu ajutorul protocolului OAUTH cu userul de reddit respectiv.

Pe partea de postari, s-ar putea rula crawlerul incontinuu pentru a indexa , colecta si analiza cat mai multe postari si a imbunatati performanta aplicatiei.

O alta dezvoltare ulterioara a aplicatiei ar putea consta in adaugarea unui serviciu de geolocatie, fapt ce ar imbunatati cu mult acuratetea recomandarilor bazandu-se pe locatia utilizatorilor. Tot ca imbunatatire ulterioara se poate incerca conectarea cu una sau mai multe siteuri externe, sa spunem, de stiri pentru a intelege motivatia utilizatorilor. Tot pe partea de analiza a utilizatorilor, se poate identifica, cu ajutorul unui analizator semantic subiectele principale de care un utilizator este interesat: muzica, filme, politica etc.

Tot ca imbunatatire ulterioara se poate considera adaugarea unei legaturi catre conturi de social-media cum ar fi facebook sau twitter, pentru a crea o imagine de ansamblu asupra unei personae si nu numai asupra contului de reddit.

Punctual, la momentul actual imbunatatirea aplicatiei se poate realiza prin cresterea numarului de utilizatori ce fac parte din setul de antrenament al retelei neuronale. Tot la momentul actual, s-ar putea inlocui functionalitatea Google API-ului cu un modul propriu de limbaj natural pentru a creste autonomia aplicatiei. In plus, se pot creea cateva conturi speciale la nivelul aplicatiei, cu rolul de administrator, ce pot schimba functia de distanta a retelei Kohonen.

Bibliografie

Carti:

- Buraga Sabin (2005), Proiectarea siturilor Web, DESIGN SI FUNCTONALITATE, Editura Polirom
- Ciortuz Liviu, Munteanu Alina, Badarau Elena (2015), Exercitii de invatare automata, Editura Universitatii Alexandru Ioan Cuza
- Evans J. Eric (2003), Domain-Driven Design, Editura Addison Wesley

Linkuri:

- NestJs, https://docs.nestjs.com/
- Node.js, https://nodejs.org/en/docs/
- Retea neuronala, https://en.wikipedia.org/wiki/Self-organizing_map
- Machine learning, https://github.com/mljs
- TypeOrm, https://typeorm.io/#/
- Baza de date, https://www.mysql.com/
- AngularJS, https://docs.angularjs.org/api
- Documentatie distanta, http://www.naun.org/main/NAUN/ijmmas/mmmas-49.pdf
- ChartJs, https://www.chartjs.org/docs/latest/
- Toate diagramele au fost realizate in UML cu ajutorul draw.io

Anexa 1

LISTA DISCIPLINELOR OBLIGATORII

- **Algoritmica Grafurilor** (vezi pagina cursului aici)
- **Arhitectura Calculatoarelor şi Sisteme de Operare** (vezi pagina cursului aici)
- **Baze de Date** (vezi pagina cursului aici)
- **Calcul Numeric** vezi pagina cursului aici)
- **Fundamentele Algebrice ale Informaticii** (vezi pagina cursului aici)
- **Grafică pe Calculator** (vezi pagina cursului aici)
- **Ingineria Programării** (vezi pagina cursului aici)
- **Inteligenţa Artificială** (vezi pagina cursului aici)
- **Învăţare Automată** (vezi pagina cursului aici)
- **Limbaje Formale, Automate şi Compilatoare** (vezi pagina cursului aici)
- **Logică pentru Informatică** (vezi pagina cursului aici)
- **Probabilităţi şi Statistică** (vezi pagina cursului aici)
- **Programare Avansată** (vezi pagina cursului aici)
- **Programare Orientată-Obiect** (vezi pagina cursului aici)
- **Proiectarea Algoritmilor** (vezi pagina cursului aici)
- **Reţele de Calculatoare** (vezi pagina cursului aici)
- **Securitatea Informaţiei** (vezi fişa disciplinei aici)
- **Sisteme de Operare** (vezi pagina cursului aici)
- **Structuri de Date** (vezi pagina cursului aici)
- **Tehnologii Web** (vezi pagina cursului aici)

** Precizare: se va considera materia disciplinelor obligatorii studiată de-a lungul celor trei ani.*