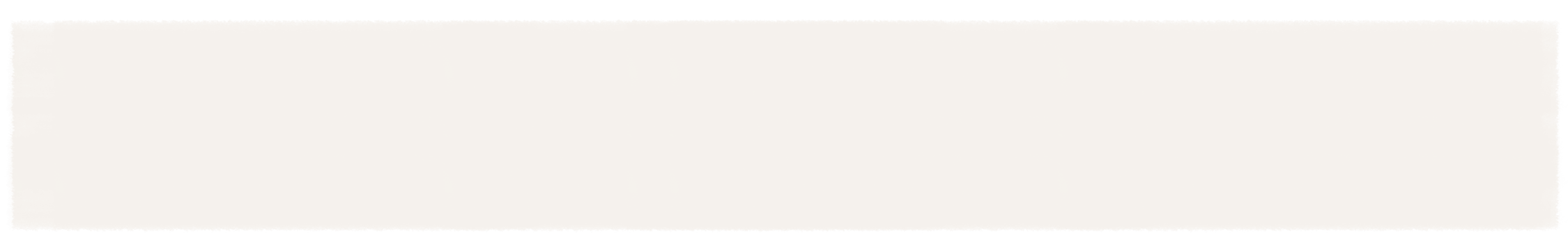
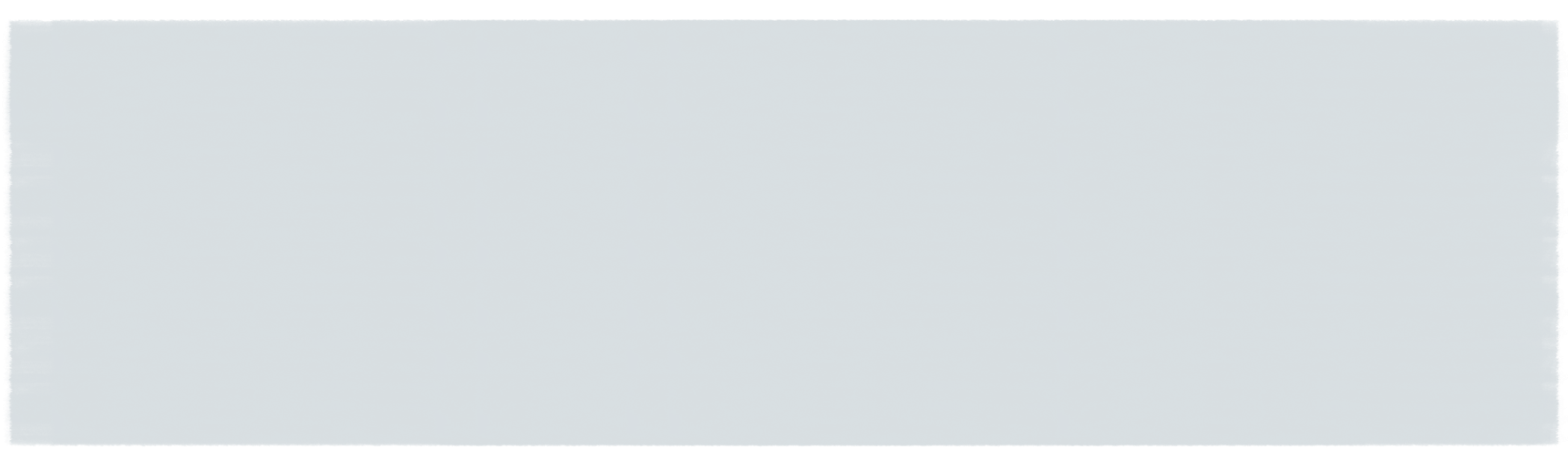
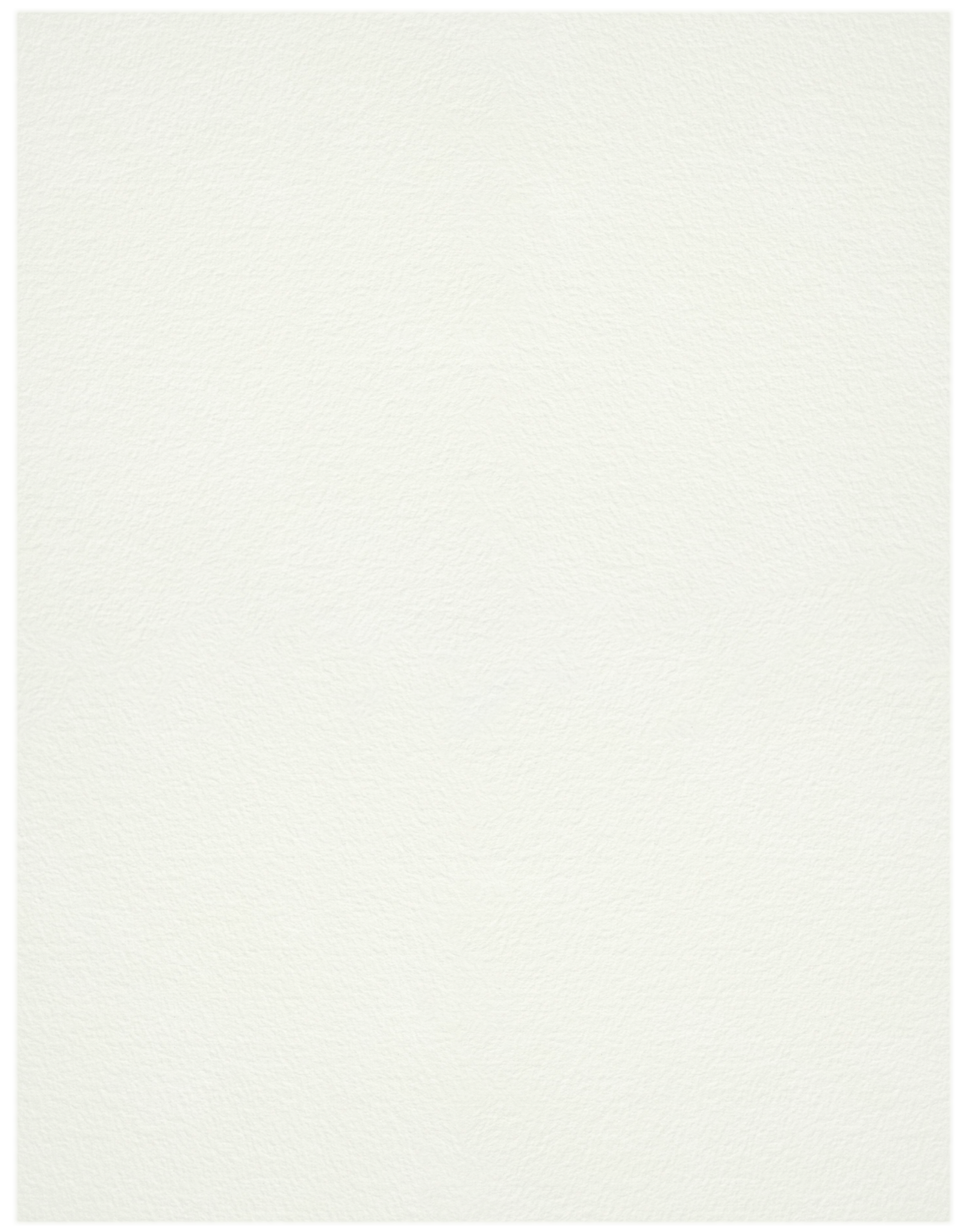
**Poročilo**



**Semantični spletni portal**

**Skupina "FERImdb"**

Vodja: Anja Hauptman

Člani:

Dominik Šbüll

Simona Siljanovska

Urška Nemet

Kazalo

[UVOD 3](#_Toc421651446)

[TF-IDF 4](#_Toc421651447)

[NAIVE BAYES CLASSIFIER 6](#_Toc421651448)

[ZAKLJUCEK 9](#_Toc421651449)

[Viri 10](#_Toc421651450)

# UVOD

Za delo pri projektu Semantični spletni portal smo si izbrali tematiko priporočanja filmov uporabnikom. V okviru predmetov Prevajanje programskih jezikov, Sistemska administracija, Spletno programiranje in Umetna inteligenca smo ustvarili spletni portal, na katerem ima uporabnik možnost, da vpiše naslov nekega filma, ki si ga je že ogledal, ali pa vpiše ključne besede, ki opišejo skupino filmov, ki ga zanimajo, in mu tako mi s pomočjo uporabe algoritmov iz umetne inteligence in podatkov, ki smo jih shranili v podatkovno bazo, priporočamo filme, ki bi uporabniku lahko bili všeč in bi si jih želel ogledati. Uporabnik ima tudi možnost izbire brskanja po filmih, ki so razdeljeni v kategorije, če ga takšen način izbire bolj zanima. V načrtu je bilo tudi to, da bi lahko prijavljenim uporabnikom glede na njihov profil in nastavitve, ki so si jih tam izbrali, priporočili filme, ki si še jih niso ogledali, na podlagi njihovih lastnih favoritov ali pa mu ponudili seznam filmov, ki bi ga utegnili zanimati glede na podobnosti okusa z drugimi uporabniki.

Za priporočanje filmov glede na druge uporabnike smo si izbrali algoritem Apriori, a se je članica, ki ji je bil algoritem dodeljen (Urška Nemet), odločila, da ga ne bo implementirala.

Podobno se je zgodilo z priporočanjem glede na gledane filme uporabnika samega, za kar smo si izbrali Pearsonov algoritem, a se je ponovno dodeljeni član (Dominik Šbüll) odločil proti njegovi implementaciji v spletni portal.

Za predlaganje filmov glede na posamezen že ogledan film, smo uporabili TF-IDF rankirno metriko. Z implementacijo le-tega se je ukvarjala Anja Hauptman, na podlagi podatkov, prenesenih iz spletne strani www.kolosej.si, kjer so polek vseh filmov napisani tudi njihovi krajši opisi. Iz teh podatkov so se vzele posamezne besede, za katere sta se izračunali tf in idf vrednosti, na podlagi katerih smo se lahko odločili, kako pomembne so posamezne besede za vsak film. Uporabnikom na podlagi teh podatkov shranjenih v podatkovni bazi nato priporočamo tiste filme, ki se po prej omenjenih besedah in njihovi pomembnosti najbolj ujemajo s filmom, ki ga je vpisal. Na podobnem principu sestoji tudi iskanje po ključnih besedah o filmu.

Za gručenje podatkov v bolj smiselne kategorije smo si izbrali Bayesov algoritem, ki ga je implementirala Simona Siljanovska. Uporabila je tf-idf podatke, ki jih je pred tem Anja že naložila v podatkovno bazo in na podlagi besed in kako so bile porazdeljene glede na filme, ustvarila okoli dvajset različnih kategorij, v katere je porazdelila filme. Še večja uporabnost algoritma pa je ta, da se bodo, ob dodajanju novih filmov v podatkovno bazo, novi filmi sami primerno razporedili v določene kategorije.

Natančneje sta algoritma opisana v nadaljevanju dokumenta.

Pri pisanju poročila sva sodelovali Anja in Simona.

# TF-IDF

Izvedba tf-idf rangirnega sistema je potekala po naslednjih točkah:

1. V okviru predmeta PPJ sem iz Kolosejeve spletne strani snela podatke o vseh filmih, ki jih imajo v bazi znanja. Ti podatki so vsebovali tudi opise posameznih filmov, ki sem jih uporabila za to nalogo.

Ker so bili podatki že naloženi v podatkovno bazo, sem iz baze s SQL poizvedbo v XML datoteko shranila ID-je vseh filmov in njihove opise. Tako sem si olajšala delo in sem lahko za vsak opis shranila posamezne besede in shranila tudi kateremu filmu pripadajo.

1. Opise sem najprej želela preoblikovati v bolj splošno obliko, zato sem z lematizacijo preoblikovala besede v njihove osnovne oblike (primer: besedo »obiskovalci« proces lematizacije preoblikuje v »obiskovalec«. Enako stori z besedo »obiskovalcev« ipd.). Za to sem uporabila orodje LemmaGen (<http://lemmatise.ijs.si/>), ki sem ga implementirala v svoj C# projekt.
2. Ko so bile vse besede v njihovih osnovnih oblikah, sem opise filmov razdelila na posamezne besede in za njih izračunala vrednosti tf (Term Frequency) – torej število kolikokrat se posamezna beseda pojavi v opisu filma.
3. Ko sem končala s preštevanjem vseh besed, sem izračunala še idf (Inverse Document Frequency) vrednosti za besede – pri tem vzamem število vseh dokumentov (v mojem primeru število opisov filmov - N) in jo delim z številom, v koliko dokumentih se beseda še pojavi (X). Nato to vrednost še logaritmiram (s poljubno osnovo, jaz sem uporabila ln).
4. Vrednost tf-idf se izračuna z množenjem vrednosti tf in idf. Višje kot je to število, bolj pomembna je beseda za opis filma.
5. Pridobljene in izračunane podatke sem dodala v podatkovno bazo spletnega portala, od kod jih uporabljamo pri rangiranju iskalnih rezultatov.

Primer:

Imejmo stavek »Na ta lep sončen dan sem videla lepo rožo.«.

S procesom lematizacije se je stavek preoblikoval v: »na to lepo sončno dan sem videti lepo roža«.

Če ta stavek razdelimo na posamezne besede dobimo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Beseda* | **Na** | **To** | **Lepo** | **Sončno** | **Dan** | **Sem** | **Videti** | **roža** |
| *Tf* | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| *Idf* |  |  |  |  |  |  |  |  |
| *idf* | 0,92 | 0,92 | 1,6 | 0,92 | 1,6 | 0,51 | 0,22 | 0,92 |
| *Tf-idf* | **0,92** | **0,92** | **3,2** | **0,92** | **1,6** | **0,51** | **0,22** | **0,92** |

Nato preštejemo kolikokat se posamezne besede pojavijo v stavku in to zapišemo kot vrednost tf.

Vrednost idf izračunamo glede na ostale stavke v sklopu obdelave. Recimo da se besede pojavijo še v drugih stavkih. Recimo da imamo še 5 drugih stavkov (N = 5) in X predstavlja v koliko stavkih se določena beseda še pojavi.

Po izračunu obeh vrednosti za vse besede, moramo vrednosti tf in idf zmnožiti.

Po izračunih opazimo, da sta za naš primer najpomembnejši besedi »lepo« in »dan«, saj imata najvišji tf-idf vrednosti, najmanj pomembna pa je beseda »videti«, saj je njena tf-idf vrednost najmanjša.

Po izračunu vseh podatkov je sledila njihova implementacija v spletni portal.

Ker sem podatke v podatkovno bazo že dodala, je večina implementacije potekala v obliki SQL poizvedb.

V portalu sem rangirno metriko TF-IDF uporabila na dveh mestih: ko uporabnik vpiše film, ki mu je všeč in mu z metriko predlagamo filme, ki so temu podobni in tudi pri iskanju filmov, ki so povezani s ključnimi besedami, ki jih uporabnik vpiše.

Najprej sem implementirala prvi problem. Na spletni strani imamo omogočeno iskanje po imenu filma, kamor uporabnik vpiše ime filma, ki si ga je že ogledal in mu je všeč. S tem imenom opravim poizvedbo v SQL bazo, kjer dobim ID filma in s to vrednostjo, lahko opravim klic procedure, ki sem jo napisala za pridobivanje podatkov.

V proceduri si najprej ustvarim začasno tabelo v katero shranim lematizirane besede iz opisa filma (ki jih dobim iz TF-IDF tabele). Nato s pomočjo te tabele izvedem poizvedbo po TF-IDF tabeli in iščem, kateri drugi filmi vsebujejo besede v tabeli. Te filme združim v skupine, da vidim, kateri filmi vsebujejo največ besed iz tabele in jih uredim po padajoči vrednosti (vrednost ujemanja dobim s seštevkom tf-idf vrednosti besed, ki se ujemajo).

Rezultat procedure je izvleček iz tabele, ki vsebuje podatke o filmih. Na spletni strani prikažem po 6 filmov, od tistega, ki se najbolj ujema, do tistega z najmanjšim ujemanjem.

Implementacija drugega dela algoritma je potekala po podobnem postopku kot prvi del. Tukaj od uporabnika zahtevamo ključne besede, za katere želi, da jih film opiše ali vsebuje, ločene s presledki. String z besedami pošljem kot vhod v drugo, a podobno, proceduro.

Ta najprej vhodni podatek razdeli glede na presledke, da iz njega dobi posamezne besede. Te besede nato lematiziram s poizvedbo v bazi, saj imamo tam shranjen slovar za lematizacijo, in njihove lematizirane oblike damo v enako razpredelnico kot v prejšnjem koraku. Nato ponovim korake od prej - torej poizvedbo po TF-IDF tabeli in urejanje po tf-idf vrednosti.

# NAIVE BAYES CLASSIFIER

Pri predmeta Umetna intelegenca smo potrebovali, vsakega clana iz skupine izbrat en algoritem, ga potem narediti in implementirati v nasega portala. Jaz za mojega algoritma sem izbrala Naivni Bayes klasifikator. Tisti algoritem sem uporabila pri klasificiranje nasih filmov v bazi, glede na nakljucnih besedah iz opisa vsakega filma.

Naivni Bayesov klasifikator je tehnika katera temelji na tako imenovano Bayesovo teoremo in je se posebej primerna, ko je dimenzionalnost vhodov visoka. Kljub svoji enostavnosti, lahko Naivni Bayes pogosto prikaze bolj sofisticirane metode za razvrscanje. Naivni Bayesov klasifikator lahko dela z poljubno stevilo neodvisnih spremenljivk, ki so neprekinjene ali kategoricne. Glede na nabor spremenljivk **X = {x1, x2, x3 ... xd}**, želimo zgraditi verjetnost posteriorno za dogodek Cj med naborom možnih izidov **C = {C1, C2, C3 ... Cd}**. V bolj znanem jeziku, X je napovednik in C je niz iz kategoricnih stopnjih, ki so prisotni v odvisne spremenljivke. S pomocjo Bayesovo pravilo:

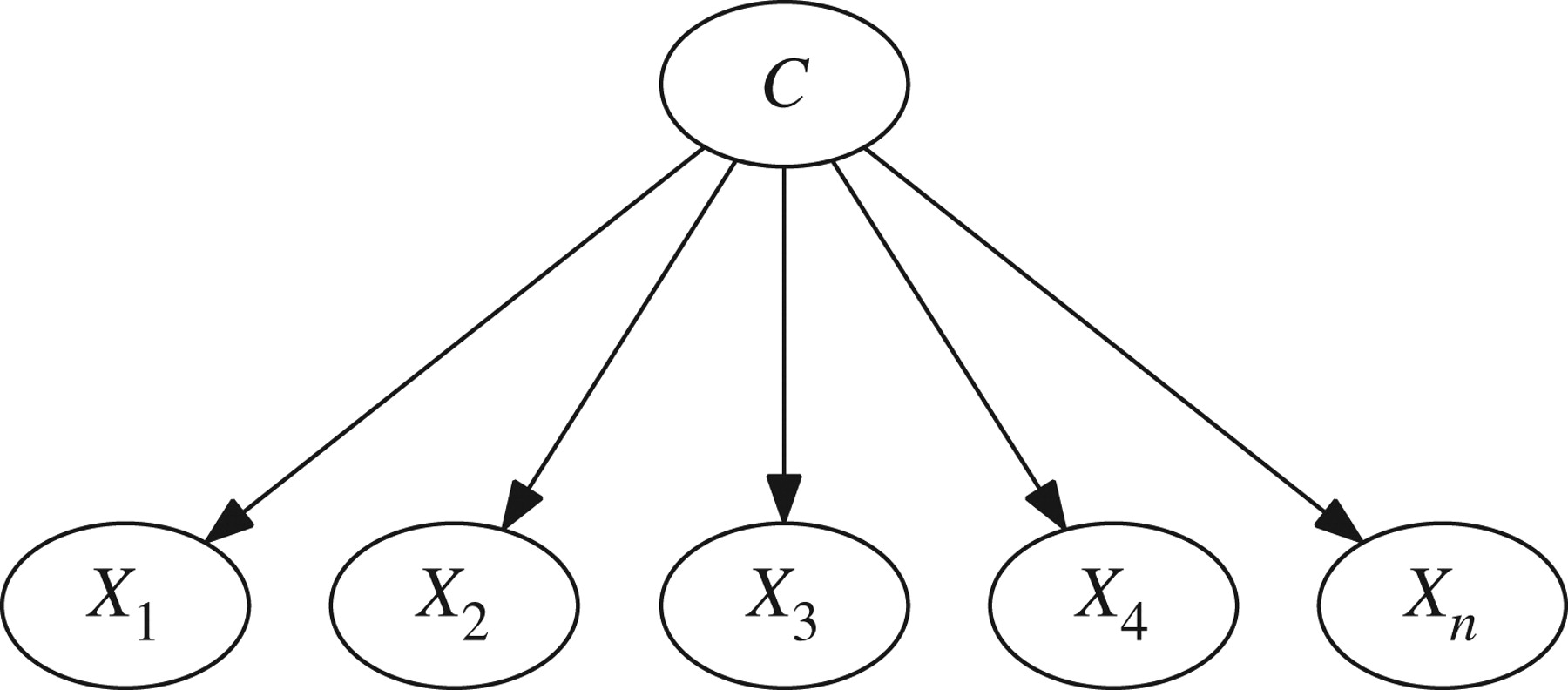
NaiveBayesIntro8.gif

kjer je **p (Cj | x1, x2, x3 ... xd)** verjetnost posterior clanstva razreda oz. to je verjetnost, da X pripada Cj. Ker naivni Bayes predpostavlja, da so pogojni verjetnosti neodvisnih spremenljivk statistično neodvisna, lahko razpade verjetnost oz.likelihood, na produkt pojmov:

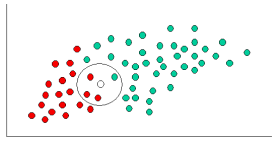
NaiveBayesIntro9.gif

NaiveBayesIntro10.gifin potem tisti produkt zapise posteriorno:

S pomocjo Bayesovo pravilo, ki je zgoraj napisano, lahko klasificirano novi objekt X z stopnjo razreda Cj, ki doseze najvisjo verjetnost posteriorno.



Ce zelimo prikazat kako deluje naivni Bayesov klasifikator, moremo si naprej dolocit neko ucno mnozico z podatkimi notri, in potem nasa naloga je, da razvrstimo novih primerov, ko prispejo, in sicer, odločimo v kateri razred bodo novih primerov pripadali na podlagi trenutnih podatkov ki jih imamo v ucni mnozici. Lahko vzamemo en primer, kjer bomo imeli rdece in zelene tocke, kot je prikazano na sliki in potem lahko klasificirano novo belo tocko, ki pride:



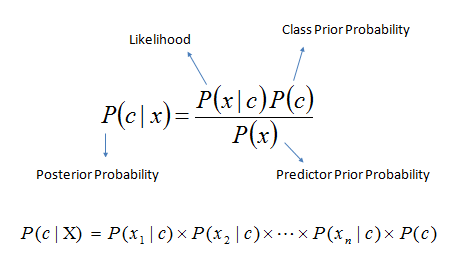
Koncno klasifikacijo naredimo, da najprej zracunamo prior verjetnost vsakega clana razreda (posebej za rdece in zelene tocke). Potem zracunamo likelihood verjetnost, tak da naredimo en krog okoli novega objekta ki si ga zelimo klasificirat, da lahko vidimo ali v tisto okolico so vec rdece ali zelene tocke in glede na to se bo nas objekt klasificiral. Na konci, glede na te dve verjenosti, zracunamo koncno oz. posterior verjetnost, ki predstavlja produkt te dve verjetnosti in dobimo rezultat kje se bo nasi novi objekt klasificiral.

**Delovanje algoritma in implementacija v projekta**

Naivni Bayesov klasifikator sem ga v nasega projekta uporabila, tako da klasificiram nove filme ki pridejo oz. ki se bodo dodajali v bazi po zanrih, glede na nakljucnih besedah iz opisa vsakega filma in potem to prikazem v nasega projekta pri iskanjem filmov po zanrih, da lahko vidimo v kateri zanr se bo novi film dodal. V prvem delu sem opisala kako deluje nasplosno naivni Bayes, ampak tukaj bom opisala katere podatke sem uporabila za mojo klasifikacijo in kako sem to naredila in implementirala v nasega projekta.

Najprej so bili zracunani stevilo in pomembnosti vsako besedo iz opisa vsakega filma s pomocjo algoritma TFIDF. Potem sem jaz za mojo klasifikacijo vzela iz tabelo TFIDF 30 besed iz vsakega opisa, ki sem jih dobila z padajocem urejanju glede na proizvod tf\*idf. Potem sem ustvarila tabelo ki jo bom potrebovala kot ucno mnozico za delovanje mojega algoritma. Notri v tisto tabelo sem dala podatke za vsakega filma, besede iz opisov filma ki sem jih dobila iz tabelo TFIDF in zanrih.

Kdaj sem koncala z pridobivanjem podatkov sem naredila algoritem in sem uporabila Bayesovo pravilo, kot je prikazano na sliki:



Naprej zracunam prior verjetnost za vsakega zanra v mojega algoritma, tako da prej zracunam stevilo vseh filmov in stevilo filmov v vsakega zanra. Potem zracunam se stevilo filmov vsakega zanra ki se nahajajo v okolici novega filma ki ga bomo klasificirali, da lahko vidimo iz katerega zanra so najvec filmov v okolici novega filma. Potem zracunam se likelihood verjetnost. Kdaj vse to koncam, na konci v mojega algoritma se zracunam tisto koncno verjetnost oz. posterior verjetnost ki nam pove v kateri zanr se bo novega filma klasificiral. Tisto verjenost zracunam kot produkt prior in likelihood verjetnosti. Kdaj koncam podam novega filma in kot rezultat na zaslonu dobim zanr v kateri se dodal film ki sem ga testirala.

Ce vzamem en film kot testni podatek in ga klasificiram z mojega algoritma, da podatki podam kot je dol napisano:

**$film = array('beseda'=>'dalmatinec', ...(se neke dodatne podatke za filma));**

**$klasifikator = 'zanr';**

**$tabela = 'Naive\_Bayes\_Classifier';**

**Bayes($film, $klasifikator, $tabela);**

Screenshot 2015-06-09 19.43.46.pngna zaslon dobim kot rezultat zanr, v kateri se bo novega filma klasificiral:

Tisti novi film ga shranim v naso bazo in potem to prikazem pri iskanjem po zanrih ki ga imamo narejeno v nasi spletni strani. To lahko vidimo kdaj si izberemo zanr v kateri se bo film klasificiral in potem vidimo da se tisti film prikaze v seznam filmov za tistega zanra.

# ZAKLJUCEK

Iz vaje pri predmetu Umetna inteligenca smo zadovoljni. Vaje so bile zelo zanimive in tekom drugega semestra smo se veliko naucili. Nase mnenje je, da ce bi imeli vec casa, bomo boljse naredili nase vaje. Zelo nas veseli, da smo uspeli naredit in implementirat tista dva algoritma, TFIDF in Naive Bayes Classifier, ki smo jih jaz in Anja izbrali pri Umetni intelegenci. Tista dva druga algoritma, Apriori in Pearson, niso uspeli nasi kolegi Urska in Dominik jih implementirat. Ni bilo tezave pri razumevanju nasih algoritmov, kjer lahko na internetu najdes zelo literature za oba algoritma. Tudi nas zelo veseli da smo uspeli nase algoritme naredit in implementirat z nasimi podatkami, ki smo jih uporabljali v nasega projekta. Pri izdelavi nasih algoritmov nismo imeli prevec tezave, kjer so zelo zanimivi in enostavni algoritmi za naredit, tak da ne potrebujes prevec casa za implementacijo. Tudi je nam bilo vsec da pri tistih algoritmov dobis natacne rezultate, ki jih lahko potem prikazes na neki lep nacin. Tisti rezultati smo jih uporabili in prikazali pri iskanju filmov po kljucnih besedah in pri iskanju filmov po zanrih. Ce pride v postev da prvic delamo vaje pri predmetu taksne narave, menimo da smo uspesno opravili svoje naloge.

# Viri

[http://en.wikipedia.org/wiki/Tf–idf](http://en.wikipedia.org/wiki/Tf%E2%80%93idf)

<http://www.tfidf.com>

<http://stevenloria.com/finding-important-words-in-a-document-using-tf-idf/>

<http://lemmatise.ijs.si/>

http://www.statsoft.com/Textbook/Naive-Bayes-Classifier#Technical%20Notes

http://en.wikipedia.org/wiki/Naive\_Bayes\_classifier

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20

withInsect\_examples.pdf