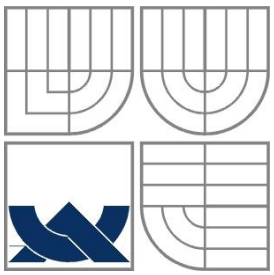
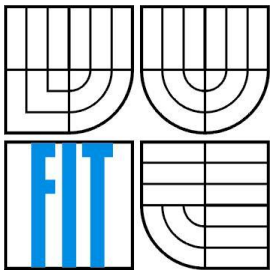


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VYUŽITÍ METOD DOLOVÁNÍ DAT PRO ANALÝZU SOCIÁLNÍCH SÍTÍ

USING OF DATA MINING METHOD FOR ANALYSIS OF SOCIAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE
AUTHOR

BC. ANDREJ NOVOSAD

VEDOUCÍ PRÁCE
SUPERVISOR

ING. VLADIMÍR BARTÍK, PH.D.

BRNO 2013

Abstrakt

Práce se zabývá problematikou dolování dat v prostředí sociálních sítí. Podává přehled o dolování z dat a možných metodách dolování. Práce také zkoumá sociální média a sítě, co mohou poskytnout a jaké problémy se sebou přinášejí. Jsou prozkoumané API třech sociálních sítí a jejich možnosti z hlediska získání dat vhodných pro dolování. Zkoumají se techniky dolování znalostí z textových dat. Je popsán způsob implementace webové aplikace, která doluje data ze sociální sítě Twitter pomoci algoritmu SVM. Implementovaná aplikace klasifikuje zprávy na základě jejich textu do tříd reprezentujících kontinenty původu. Je provedeno několik experimentů v softwaru RapidMiner a v implementované webové aplikaci a jejich výsledky jsou prozkoumány.

Abstract

Thesis discusses data mining the social media. It gives an introduction about the topic of data mining and possible mining methods. Thesis also explores social media and social networks, what are they able to offer and what problems do they bring. Three different APIs of three social networking sites are examined with their opportunities they provide for data mining. Techniques of text mining and document classification are explored. An implementation of a web application that mines data from social site Twitter using the algorithm SVM is being described. Implemented application is classifying tweets based on their text where classes represent tweets' continents of origin. Several experiments executed both in RapidMiner software and in implemented web application are then proposed and their results examined.

Klíčová slova

Dolovanie z dát, sociálne médiá, sociálne siete, API, OAuth, Facebook, Twitter, Last.fm, sociálny graf, JSON, PHP, Phirehose, JavaScript, Nette, MySQL, dolovanie z textových dát, klasifikácia dokumentov, TF-IDF, Naivný Bayes, k-najbližší susedia, SVM, support vector machine, RapidMiner

Keywords

Data mining, social media, social networks, API, OAuth, Facebook, Twitter, Last.fm, social graph, JSON, PHP, Phirehose, JavaScript, Nette, MySQL, text mining, document classification, TF-IDF, Naive Bayes, k-nearest neighbors, SVM, support vector machine, RapidMiner

Citace

Andrej Novosad: Využitie metód dolovania dát pre analýzu sociálnych sietí, diplomová práce, Brno, FIT VUT v Brně, 2013

Využitie metód dolovania dát pre analýzu sociálnych sietí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Vladimíra Bartíka, Ph. D.
Další informace mi poskytli Michal Turoň, Marián Hacaj.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Andrej Novosad
19.05.2013

Poďakovanie

Chcel by som sa poďakovať svojmu vedúcemu Ing. Vladimírovi Bartíkovi za čas, ktorý mi bol ochotný venovať a za jeho príjemný a uvoľnený prístup a tiež rady, ktoré mi poskytol pri analýze projektu a pri vytváraní tejto dokumentácie.

Obsah

1 Úvod

Hlavným zámerom tohto projektu je problematika dolovania z dát a získavanie znalostí z dát v oblasti sociálnych sietí. Sociálne siete predstavujú čoraz väčšiu súčasť života pre mnoho užívateľov internetu, ktorí trávia na nich stále viac a viac svojho voľného času. So svojimi miliónmi aktívnych užívateľov z celého sveta tak sociálne siete vo svojich dátach ukrývajú potenciálne veľké množstvo informácií, ktoré by mohli byť využiteľné na viaceré účely, či už v oblasti marketingu alebo politike. Tento trend narastá už niekoľko rokov a je veľmi pravdepodobné, že v tom bude pokračovať aj v blízkej budúcnosti. Sociálne siete sa tak stávajú často terčom pre tzv. dataminerov, ktorí môžu takto objavovať nové trendy a zaujímavé informácie v rôznych oblastiach.

Kapitola 2 poskytuje prehľad problematiky dolovania z dát ako takej a zároveň sa snaží tento problém posadiť do kontextu sociálnych sietí.

V tretej kapitole tejto práce sú preskúmané tri rôzne sociálne siete z pohľadu toho, aké možnosti ponúkajú pre dolovanie z ich dát. Jedná sa o webovú sociálnu sieť Facebook, mikroblogovaciu službu Twitter a hudobný portál Last.fm.

Dolovanie znalostí z textov poníma kapitola 4, kde sú vysvetlené princípy, postupy a algoritmy, ktoré sa v tejto oblasti informatiky používajú. Jedným z hlavných zameraní je algoritmus SVM, ktorý okrem iného slúži na klasifikáciu vzoriek do tried.

Kapitola 5 sa zaoberá spôsobom, akým možno získavať tweety zo sociálnej siete Twitter pre ich neskoršie využite v oblasti dolovania znalostí z textov. Ďalšia kapitola priblíži prípravu pred samotnou implementáciou webovej aplikácie. Ide o nastavenie a využitie prostredia pre dolovanie, program RapidMiner, v ktorom boli vykonané experimenty pre porovnanie dolovacích metód Naivný Bayes a k-najbližší susedia. Následne je v kapitolách 7 a 8 predložený návrh a popísaná konkrétna implementácia webovej aplikácie, ktorá doluje informácie z textových dát. Kapitoly 9 a 10 zahrňujú experimentálnu časť, získané výsledky a ich celkové zhodnotenie.

Na záver je predložené celkové zhrnutie práce, jej prínosu a prípadných vylepšení a možných smerov, ktorými by sa ďalší vývoj implementovanej aplikácie mohol uberať.

Táto diplomová práca nadväzuje na semestrálny projekt, z ktorého boli prevzaté kapitoly 2 a 3. Z týchto kapitol vzišlo následné rozhodnutie, akým spôsobom bude vývoj tejto práce pokračovať a čím konkrétne sa bude práca v ďalších častiach zaoberať. Šlo teda o preskúmanie problematiky dolovania znalostí z textových dát a o implementovanie webovej aplikácie, ktorá bude klasifikovať krátke textové správy pomocou algoritmu SVM.

2 Teoretický prehľad k dolovaniu z dát

HÍbková analýza dát (ang. data mining [dejta majnyn] – dolovanie z dát) [1] alebo neodborne dolovanie dát je proces analýzy dát z rôznych perspektív a ich sumarizácia na užitočné informácie. Dolovanie z dát tiež znamená získavanie znalostí z databáz (ang. Knowledge Discovery from Data - KDD) [2], ktoré opisuje typický proces extrakcie zaujímavých (netriviálnych, skrytých, predtým nepoznaných a potenciálne užitočných) modelov dát a vzorov z veľkých objemov dát. Tieto modely reprezentujú znalosti získané z dát. [3]

Získavanie znalostí zhruba pozostáva z nasledujúcich úloh: predspracovanie dát, dolovanie z dát a následne spracovanie dát. Tieto kroky nutne nemusia byť oddelené úlohy, ale môžu byť skombinované dohromady. Dolovanie z dát je integrálna súčasť viacerých súvisiacich oblastí ako štatistika, matematika (matematické modelovanie tj. klasifikačné pravidlá alebo stromy, regresia, zhluková analýza), umelá inteligencia (neurónové siete, rozpoznávanie vzorov, samoučiace sa algoritmy), databázové systémy, nástroje OLAP (on-line analytické spracovanie), strojové učenie, vizualizácia, bioinformatika a iné. [1, 4]

Algoritmy dolovania z dát sa hrubo delia na algoritmy s kontrolovaným učením (učenie sa s učiteľom), polo-kontrolovaným (zmiešané učenie sa) a nekontrolovaným učením (učenie sa bez učiteľa). Klasifikácia je bežný príklad prístupu s kontrolovaným učením. Pre algoritmy s kontrolovaným učením je množina dát typicky rozdelená do dvoch častí: trénovacie a testovacie dáta so známymi triedami. [4]

Algoritmy s kontrolovaným učením budujú klasifikačné modely z trénovacej množiny dát a používajú naučené modely pre predikciu. Pre ohodnotenie činnosti klasifikačného modelu je tento model aplikovaný na testovaciu množinu dát, čím sa získa presnosť klasifikácie. Medzi typické metódy s kontrolovaným učením patrí klasifikácia pomocou rozhodovacích stromov, naivná bayesovská klasifikácia, klasifikácia pomocou neurónových sietí a algoritmus k-najbližších susedov.

Algoritmy s nekontrolovaným učením sú navrhnuté pre dáta bez tried. Zhlukovanie je typickým príkladom nekontrolovaného učenia. Algoritmy s nekontrolovaným učením pre danú úlohu budujú model založený na podobnosti alebo rozdielnosti medzi objektmi z množiny dát. Podobnosť alebo rozdielnosť medzi dátovými objektami môže byť meraná pomocou Euklidovskej príp. Manhattanskej vzdialenostnej funkcie.

Algoritmy s polo-kontrolovaným učením sú najlepšie využiteľné tam, kde existuje malé množstvo označených dát a veľké množstvo neoznačených dát. Dve typické metódy zmiešaného učenia sa sú polo-kontrolovaná klasifikácia a polo-kontrolované zhlukovanie. Prvý spôsob používa označené dáta pre klasifikáciu a neoznačené dáta pre vylepšenie a zjemnenie hraníc klasifikácie. Druhý spôsob používa označené dáta pre správne vedenie zhlukovania.

Pre úplnosť boli spomenuté rôzne spôsoby, ktorými môže byť získavanie znalostí z dát prevedené. Táto práca však bude bližšie pojednávať konkrétne o technikách automatickej klasifikácie textových dokumentov. Tieto techniky, pomocou ktorých môžu byť textové dokumenty automaticky klasifikované, budú hlbšie preskúmané v kapitole 4.

2.1 Sociálne siete

Sociálne siete sú definované ako skupina internetovo založených aplikácií, ktoré sú postavené na ideológii a technologických základoch Webu 2.0, a ktoré umožňujú tvorbu a výmenu užívateľom vytvoreného obsahu. Sociálne médiá sú konglomerátom rôznych typov sociálnych médií a stránok vrátane tradičných médií, ako sú noviny, rádio, televízia a netradičné médiá, ako Facebook, Twitter, Last.fm, Reddit, Google+ atď. [5]

Sociálne médiá ponúkajú užívateľom jednoduchý spôsob pre vysoko efektívnu vzájomnú komunikáciu takých rozmerov, aké nemožno vidieť pri tradičných médiách. Popularita sociálnych médií pokračuje v exponenciálnom raste, z čoho vyplýva evolúcia sociálnych sietí, blogov, mikro-blogov, wiki stránok, sociálnych noviniek, zdieľanie médií (text, fotky, audio a video) atď.

2.1.1 Dolovanie z dát sociálnych sietí

Užívatelia vytvoria na sociálnych sieťach každý deň obrovské množstvo obsahu. Je veľmi pravdepodobné, že tento trend bude v budúcnosti pokračovať s exponenciálne väčším objemom dát. Preto je pre výrobcov, spotrebiteľov a poskytovateľov služieb dôležité prísť na správny spôsob, ako tieto údaje vhodne manažovať a využiť. Sociálne siete sa vyvíjajú smerom dopredu prostredníctvom snahy zodpovedať nasledujúce otázky:

- Ako môže užívateľ dať o sebe vedieť? / Ako môže byť vypočutý?
- Aký zdroj informácií by mal užívateľ využiť?
- Ako zlepšiť užívateľovu skúsenosť?

Odpovede na tieto otázky sú skryté v dátach sociálnych sietí. Tieto výzvy predstavujú pre „data minerov“ hojné príležitosti k vytvoreniu nových algoritmov a metód pre získavanie znalostí zo sociálnych sietí.

Dáta generované na sociálnych sieťach majú iný charakter, než konvenčné dáta v tvare „atribút–hodnota“ z klasického dolovania z dát. Dáta sociálnych médií do značnej miery tvoria samotní užívatelia sociálnych sietí. Tieto dáta sú rozsiahle, zašumené, rozptýlené, neštruktúrované a dynamické, čo v oblasti získavania znalostí z dát predstavuje nemalé prekážky pri objavovaní nových a efektívnych techník a algoritmov.

Napríklad Facebook má 137.6 miliónov unikátnzch návštevníkov za mesiac a to iba v USA. Twitter má 140 miliónov aktívnych užívateľov, ktorí denne pošlú 340 miliónov tzv. tweetov (krátkych textových správ). [6]

Na základe konkrétnej platformy sociálnej siete môžu byť dáta často veľmi zašumené. Odstránenie šumu z dát je esenciálnou úlohou pred samotným dolovaním. Odborníci totiž zaznamenávajú, že spameri vygenerujú viac dát ako legitímni užívatelia. [7,8]

Stránky sociálnych sietí sú dynamické a priebežne sa vyvíjajúce. Napríklad Facebook počas svojho pôsobenia priniesol niekoľko nových konceptov ako timeline, tvorenie skupín pre užívateľa a početné množstvo zmien týkajúcich sa súkromia užívateľov.

Existuje veľa ďalších zaujímavých otázok týkajúcich sa ľudského správania, ktoré možno študovať a skúmať využívajúc pri tom dáta zo sociálnych sietí. Sociálne siete môžu tiež pomáhať inzerentom nachádzať vplyvných ľudí pre maximalizovanie dosahu ich produktov. Sociálne siete môžu tiež pomôcť sociológom odhaľovať ľudské správanie v rámci rôznych záujmových skupín. Okrem toho majú sociálne siete už takmer neoddeliteľnú úlohu pri uľahčovaní a napomáhaní masovým hnutiam, protestom a revolúciám ako boli Arab Spring, Occupy Wall Street, revolúcia v Egypte v roku 2011 a iné. [9, 10, 11]

2.1.2 Súkromie, bezpečnosť a dôvera

Používanie sociálnych médií so sebou nesie obavy týkajúce sa užívateľovej bezpečnosti a jeho súkromia. Vznikajú nové problematické otázky, pretože na seba narážajú dve protichodné požiadavky: na jednej strane chce mať užívateľ čo najviac priateľov a zdieľať čo možno najviac informácií, no na strane druhej chce mať aj čo najviac zachovaného súkromia. Byť spoločenský vyžaduje otvorenosť a transparentnosť a naopak mať súkromie so sebou prináša obmedzenie množstva informácií, ktoré užívateľ zdieľa. Navyše stránky sociálnych sietí k tomu, aby naďalej fungovali a rástli, povzbudzujú užívateľov, aby sa ľahko medzi sebou našli a rozširovali svoje siete priateľov čo najširšie, teda aby boli otvorení. S množstvom rozmanitých informácií odhalených v užívateľských profiloch (odkiaľ môžu byť nepriamo dostupné aj informácie o iných užívateľoch), sa môžu samotní užívatelia spolu s ich priateľmi a členmi ich skupín vystaviť riziku rôznych útokov. Sociálne médiá boli už terčom mnohých pasívnych ale aj aktívnych útokov, ako sú sledovanie, šikanovanie, krádež informácií (tzv. phishing), spam, podvod a iných. [4]

Napríklad iba menšina užívateľov si mení predvolené nastavenia súkromia. V niektorých prípadoch sú užívateľské profily úplne verejné, takže sú všetky informácie pre kohokoľvek dostupné. Keď je profil verejne prístupný, stáva sa terčom zlomyseľných užívateľov, ktorí využívajú tieto dáta pre osobný prospech. [12]

Tendencia tvorcov sociálnych sietí je stále viac pamätať na užívateľovo súkromie. Je to poznať z toho, že v priebehu rokov s narastajúcimi skúsenosťami menia politiku pojednávajúcu

o súkromí užívateľov. Prejavuje sa to tak, že napr. stupeň súkromia je pre nových užívateľov predvolene nastavený na privátnejší. Okrem toho sa sprostredkovatelia sociálnych sieti snažia šíriť osvetu ohľadom bezpečnostných rizík a nastavení rôznymi upozorneniami a pripomienkami.

3 API sociálnych sietí

V tejto kapitole sú preskúmané API (skratka pre Application Programming Interface) troch rôznych sociálnych sietí a ich možnosti z pohľadu získavania dát vhodných pre dolovanie. Ide o sociálne siete Facebook, Twitter a Last.fm.

Application programming interface alebo skratkou **API** (rozhranie pre programovanie aplikácií). Tento termín je používaný v programovaní. Ide o zbierku funkcií a tried (ale aj iných programov), ktoré určujú, akým spôsobom sa majú funkcie knižníc volať zo zdrojového kódu programu. API funkcie sú programové celky, ktoré programátor volá namiesto vlastného naprogramovania. [13]

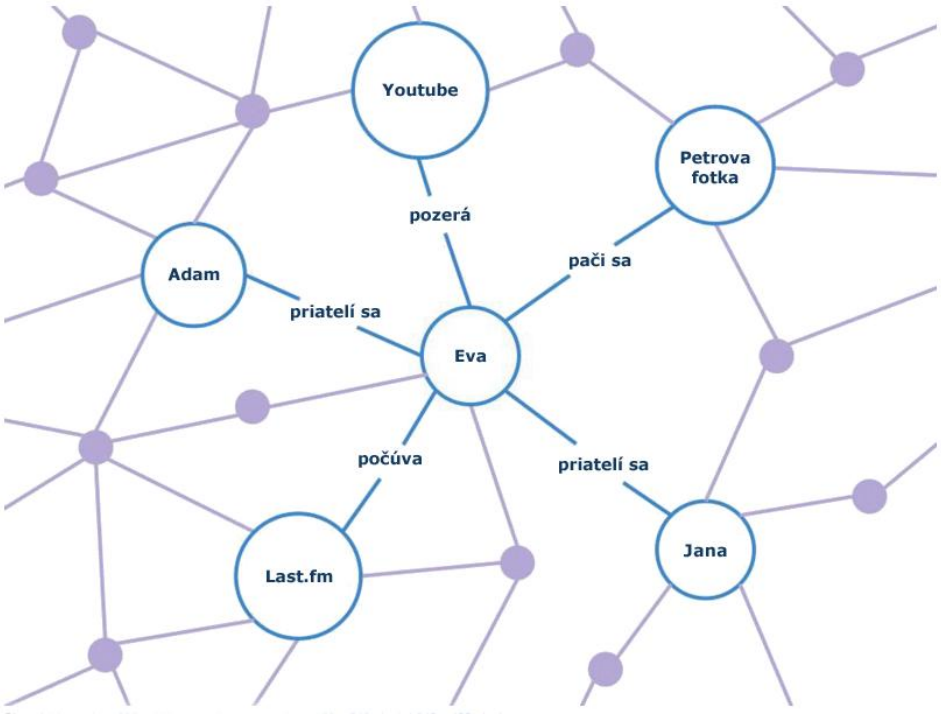
3.1 Facebook

Facebook je sociálna sieť spustená vo februári roku 2004. V septembri roku 2012 má Facebook viac než miliardu aktívnych užívateľov, z ktorých viac ako polovica používa Facebook na mobilných zariadeniach. [14]

3.1.1 Sociálny graf

Jadro Facebooku tvorí tzv. sociálny graf zložený z ľudí a ich vzťahov k iným ľuďom a objektom. [15]
Sociálny graf v kontexte internetu je tzv. sociogram. Sociogram je grafická reprezentácia sociálnych prepojení, ktoré daná osoba má. Zobrazuje teda štruktúru osobných vzťahov internetových užívateľov. [16]

Obrázok 1 zachytáva rôzne typy vzťahov medzi sociálnymi objektami. Užívateľ Eva je priateľ Adama a Jany, kým Adam a Jana priatelia nie sú. Eva je ich spoločný priateľ. Eva počúva Last.fm rádio a pozerá video z Youtube.



Obrázok 3.1 - príklad sociálneho grafu [16]

3.1.2 Graph API

Graph API predstavuje jednoduchý, konzistentný pohľad na sociálny graf Facebooku, rovnomerne reprezentujúc objekty v grafe (napr. ľudí, fotky, udalosti, stránky) a vzťahy medzi nimi (napr. priateľstvá, zdieľaný obsah, označené fotky – tzv. foto tags).

Graph API je primárny spôsob ako dostať dáta do a zo sociálneho grafu Facebooku. Ide o nízkoúrovňové API založené na HTTP, ktoré je možné použiť na dotazovanie sa na dáta alebo na iné úlohy, ktoré by aplikácia mohla potrebovať. Open Graph umožňuje definovať nové objekty a akcie v užívateľovom sociálnom grafe a spôsob ako vytvárať nové inštancie týchto akcií a objektov pomocou Graph API.

Každý objekt v sociálnom grafe má svoje unikátne ID. Vlastnosti tohto objektu sú prístupné na adrese https://graph.facebook.com/ID. Napr. oficiálna stránka pre Facebook platform má ID 19292868552, takže objekt je možné dostať na adrese https://graph.facebook.com/19292868552, kde je objekt reprezentovaný v nasledujúcej JSON notácii:

```
{
   "about": "Build and distribute amazing social apps on Facebook...",
   "company_overview": "Facebook Platform enables anyone to build...",
   "is_published": true,
   "talking_about_count": 25405,
   "username": "FacebookDevelopers",
   "website": "http://developers.facebook.com",
   "were_here_count": 0,
```

```
   "category": "Product/service",
   "id": "19292868552",
   "name": "Facebook Developers",
   "link": "http://www.facebook.com/FacebookDevelopers",
   "likes": 362557,
   "cover": {
      "cover_id": "10151121467948553",
      "source": "http://sphotos-f.ak.fbcdn.net/hphotos-ak-123.png",
      "offset_y": 0
   }
}
```

3.1.2.1 JSON

JavaScript Object Notation (JSON) je textovo založený otvorený štandard, ktorý je navrhnutý pre človekom čitateľnú výmenu dát. Tento štandard je odvodený zo skriptovacieho jazyku JavaScript pre reprezentáciu jednoduchých dátových štruktúr a asociatívnych polí, ktoré sa nazývajú objekty. Napriek vzťahu s JavaScriptom je JSON jazykovo nezávislý a existujú dostupné parsre pre mnoho rôznych jazykov. Formát JSON je často používaný pre prenášanie štruktúrovaných dát cez internetovú sieť. Primárne je využívaný pre prenos dát medzi serverom a webovou aplikáciou, takže sa jedná o alternatívu k XML. [17]

3.1.3 Autentifikácia a autorizácia

Pomocou Graph API je možné ľahko získať prístup ku všetkým verejným informáciám o objekte, avšak k získaniu dodatočných informácií je nutné dostať od užívateľa povolenie. Toto sa robí pomocou nástroja Facebook Login autentifikáciou užívateľa a následného požiadania užívateľa, aby aplikáciu autorizoval. Graph API používa pre autorizáciu otvorený štandard OAuth 2.0.

3.1.3.1 OAuth

OAuth je otvorený štandard pre autorizáciu. OAuth sprostredkováva spôsob, pomocou ktorého môžu klienti pristupovať k prostriedkom serveru v mene vlastníka prostriedkov (čo môže byť iný klient alebo koncový užívateľ). Pre koncových užívateľov taktiež poskytuje proces pre autorizáciu a sprostredkovanie prístupu tretím stranám k prostriedkom na serveri bez zdieľania ich osobných údajov (typicky prihlasovacie meno a heslo). [18]

3.1.3.2 Facebook Login

Facebook ponúka viacero možných procesov prihlásenia sa pre rôzne zariadenia a projekty. Tieto procesy generujú tzv. access token, ktorý môže programátor využiť pre vytváranie API volaní v mene daného užívateľa. O perzistenciu tokenu a podpisovanie API volaní sa automaticky stará klientské JavaScriptové SDK, ktoré je vhodné pre webovú aplikáciu. Okrem toho existujú aj iOS SDK alebo Android SDK.

Klientské JavaScript SDK

JavaScript SDK ponúka rýchly spôsob ako integrovať prihlasovací proces do aplikácie pre akékoľvek zariadenie, na ktorom je možné pustiť internetový prehliadač podporujúci JavaScript. Keďže SDK sa stará o každý access token, ktorý bol vygenerovaný počas prihlasovania a volania Graph API sú automaticky podpisované, jediné, čo musí programátor spracovávať a ukladať, sú dáta poskytnuté API volaniami.

Nasledujúci diagram načrtáva proces klientského prihlasovania:

Obrázok 3.2 – proces prihlasovania pomocou JavaScript SDK [15]

3.1.3.3 Povolenia

Aby aplikácia mohla používať užívateľské dáta, je nutné od každého užívateľa, aby dal explicitné povolenie, že súhlasí s tým, aby jeho osobné informácie mohla využívať tretia strana. Aplikácia môže od užívateľov požadovať istú množinu osobných údajov. Rôzni užívatelia však majú rôzne nastavenie a každý jej môže poskytnúť len iné obmedzené množstvo informácií. Aplikácia teda musí byť napísaná tak, aby bola schopná pracovať s variabilnou množinou informácií obdržaných od užívateľov. Okrem toho niektorí užívatelia nemusia povoliť užívať aplikácii žiadne svoje osobné dáta. Získavať teda veľké objemy dát pre dolovanie by vyžadovalo relatívne veľké množstvo času, nakoľko by bola od každého potenciálneho užívateľa nutná jeho spolupráca a povolenie.

3.2 Twitter

Twitter je online sociálna mikroblogovacia sieť, ktorá umožňuje svojim užívateľom posielať a čítať textové správy dlhé maximálne 140 znakov. Tieto správy sa hovorovo nazývajú „tweets“. Twitter bol spustený v júli 2006, odkedy rapídne nabral celosvetovú popularitu s viac ako 500 miliónmi registrovaných užívateľov v roku 2012, ktorí spolu denne vygenerujú cez 340 miliónov tweetov. Twitter sa od svojho spustenia dostal medzi desať najnavštevovanejších stránok. Neregistrovaní užívatelia môžu čítať tweety, kým registrovaní užívatelia ich môžu odosielať buď cez webové rozhranie, cez SMS alebo škálou aplikácií pre mobilné zariadenia. [19]

Dolovať dáta z Twitter statusov nie je triviálny problém. V porovnaní napr. s novinami alebo bežnými blogmi, kde sú vety väčšinou dobre konštruované a mená napísané vo svojich plných tvaroch, sú tweety plné slangu a novovznikajúceho žargónu. Tweet obmedzený dĺžkou 140 znakov taktiež ľudí núti k značnej dávke kreativity, aby sa im podarilo do správy vtesnať želanú informáciu.

Okrem toho sa môže zdať, že 140 znakov dlhé statusy nemusia obsahovať dostatok relevantného obsahu pre dolovanie. Je treba poznamenať, že tweety obsahujú oveľa viac dát, než len akokoľvek krátku myšlienku, ktorú chcel užívateľ zdieľať so svojimi priateľmi cez túto sociálnu sieť. Každý tweet obsahuje okrem samotnej správy aj množstvo metadát, teda informácií, napr. o tom, kedy a kým, prípadne z akej aplikácie, bol tweet odoslaný.

Príklad toho, ako môže tweet spolu s jeho metadátami vyzerať v JSON notácii:


```
{
"id" => 12296272736,
"text" => "An early look at Annotations",
"created_at" => "Fri Apr 16 17:55:46 +0000 2010",
"in_reply_to_user_id" => nil,
"in_reply_to_screen_name" => nil,
"in_reply_to_status_id" => nil
"favorited" => false,
"truncated" => false,
"user" =>
      {"id" => 6253282,
      "screen_name" => "twitterapi",
      "name" => "Twitter API",
      "description" => "The Real Twitter API",
      "url" => "http://apiwiki.twitter.com",
      "location" => "San Francisco, CA",
      "profile_background_color" => "c1dfee",
      "profile_background_image_url" =>
      "http://a3.twimg.com/profile_background_images/59931/background.png",
      "profile_background_tile" => false,
      "profile_image_url" =>
      "http://a3.twimg.com/profile_images/689684365/api_normal.png",
      "profile_link_color" => "0000ff",
      "profile_sidebar_border_color" => "87bc44",
      "profile_sidebar_fill_color" => "e0ff92",
      "profile_text_color" => "000000",
      "created_at" => "Wed May 23 06:01:13 +0000 2007",
      "contributors_enabled" => true,
      "favourites_count" => 1,
      "statuses_count" => 1628,
```

```
      "friends_count" => 13,
      "time_zone" => "Pacific Time (US & Canada)",
      "utc_offset" => -28800,
      "lang" => "en",
      "protected" => false,
      "followers_count" => 100581,
      "geo_enabled" => true,
      "notifications" => false,
      "following" => true,
      "verified" => true},
"contributors" => [3191321],
"geo" => nil,
"coordinates" => nil,
"place" =>
      {"id" => "2b6ff8c22edd9576",
      "url" => "http://api.twitter.com/1/geo/id/2b6ff8c22edd9576.json",
      "name" => "SoMa",
      "full_name" => "SoMa, San Francisco",
      "place_type" => "neighborhood",
      "country_code" => "US",
      "country" => "The United States of America",
      "bounding_box" =>
            {"coordinates" =>
            [[[-122.42284884, 37.76893497],
            [-122.3964, 37.76893497],
            [-122.3964, 37.78752897],
            [-122.42284884, 37.78752897]]],
            "type" => "Polygon"}},
"source" => "web"}
```

Potencionálne najrelevantnejšie metadáta sú zvýraznené tučne a ide v poradí o tieto informácie:

- Text samotnej správy
- Dátum vytvorenia správy
- Užívateľ
    - Autorovo prihlasovacie meno
    - Autorovo meno v tvare, v akom sa zobrazuje na stránke pre ostatných užívateľov
    - Autorova biografia
    - Autorova stránka
    - Autorova lokácia (ide o textové pole a nedá sa zaručiť, či by lokácia bola dohľadateľná geografickými koordinátami)
    - Dátum vytvorenia účtu
    - Počet obľúbených tweetov, ktoré sa tomuto užívateľovi páčili
    - Počet tweetov, ktoré autor vytvoril
    - Počet užívateľov, ktorých autor tohto tweetu sleduje
    - Časová zóna
    - Užívateľov vybraný jazyk
    - Počet ľudí, ktorí sledujú autora tweetu
    - Značka, či sa jedná o overeného užívateľa

- GeoJSON Tag
- Miesto pridružené tweetu (miesto odoslania tweetu)
    - ID miesta
    - Názov miesta
    - Typ miesta (mesto, dedina, oblasť...)
    - Krajina, v ktorej sa miesto nachádza
    - Kód krajiny
    - Koordináty miesta
- Zdroj, z ktorého bol tweet odoslaný

Z jedného tweetu teda možno vydedukovať mieru, akou sa užívateľ aktívne zapája, jeho popularitu alebo lokáciu (ktorá môže byť celkom presná, ak sa jedná o geograficky označené miesto). [20, 21]

3.2.1 Twitter REST API

REST API ponúka jednoduché rozhranie pre väčšinu funkcionality Twitteru. Pomocou dotazov je možné získavať všetky verejné dostupné informácie.

Representational State Transfer (REST) je štýl softwarovej architektúry pre distribuované systémy, ako je World Wide Web. Rozhranie REST je použiteľné pre jednotný a jednoduchý prístup k zdrojom. Zdrojom môžu byť dáta, prípadne stavy aplikácie (ak sú tieto stavy reprezentovateľné konkrétnymi dátami). REST je teda orientovaný dátovo, nie procedurálne. Všetky zdroje majú vlastný identifikátor URI. REST architektúry pozostávajú z klientov a serverov. Klienti posielajú servrom požiadavky; servre spracúvajú požiadavky a vracajú patričné odpovede. [23]

3.2.2 Twitter Streaming API

Streaming API poskytuje možnosť streamovať statusy v takmer reálnom čase ponad perzistentné spojenie. Je ich možné filtrovať pomocou kľúčového slova alebo užívateľovho identifikátoru.

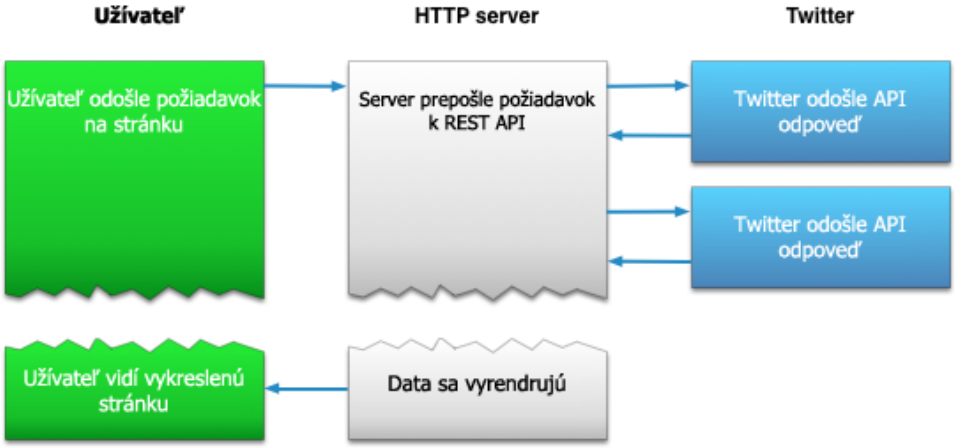
Sada streamovacích API poskytnutých Twitterom dáva vývojárom prístup s nízkou odozvou do globálneho streamu tweetov. Správna implementácia streamovacieho klienta by zahrňovala posielanie správ, ktoré by upozorňovali, že nastal nový tweet, alebo že nastala nejaká iná udalosť bez nadbytočnej réžie, ktorá je spojená s dotazovaním sa na REST koncový bod. Streaming API umožňuje špecifikovať a sledovať veľké množstvo kľúčových slov, na základe ktorých navracia tweety s geograficky označenou lokáciou z konkrétneho regiónu. Okrem toho si možno nechať vrátiť verejné statusy konkrétnej množiny užívateľov.

Twitter ponúka niekoľko streamovacích koncových bodov, z ktorých každý je prispôsobený na iný konkrétny účel: verejný stream (public stream), užívateľský stream (user stream), stream stránok (site stream). [23]

Verejné streamy	Streamy verejných dát tečúcich cez Twitter. Sú vhodné pre sledovanie špecifických užívateľov alebo konkrétne témy a na dolovanie dát.
Užívateľské streamy	Užívateľov stream obsahuje zhruba všetky dáta korešpondujúce s jedným konkrétnym užívateľom a jeho pohľadom na Twitter.
Streamy stránok	Multi-užívateľská verzia užívateľského streamu. Streamy stránok sú určené pre servre, ktoré sa pripájajú na Twitter v mene viacerých užívateľov.

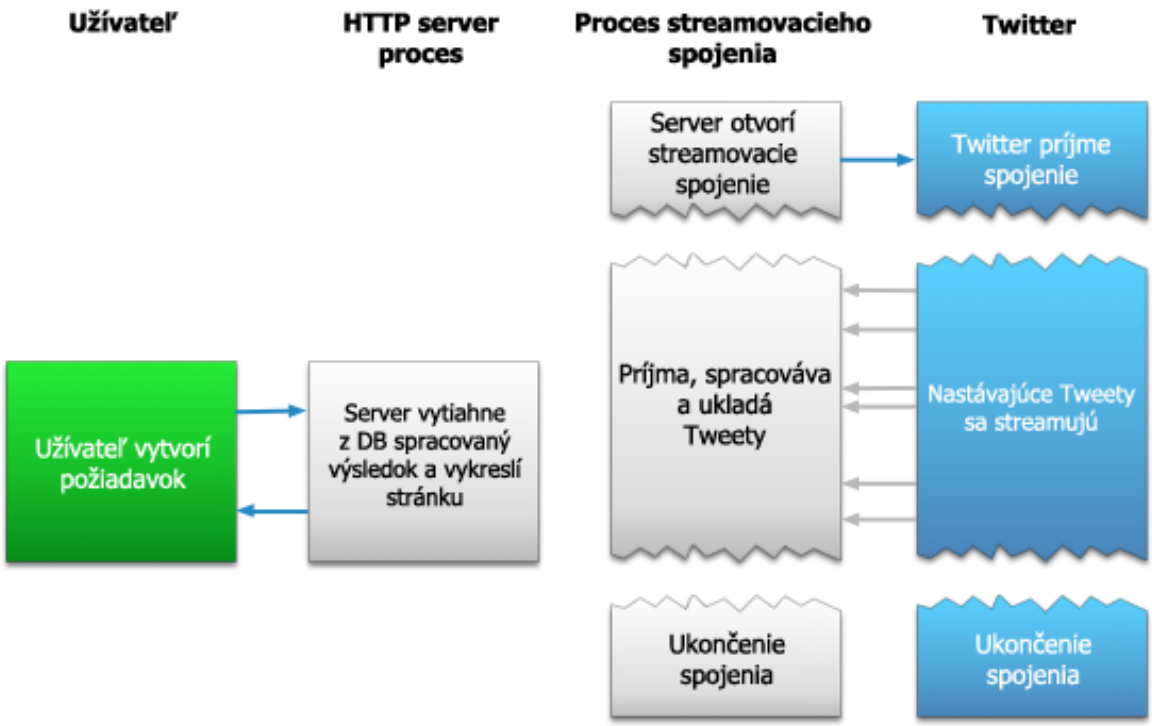
Rozdiely medzi Streaming API a REST API

Pripojenie sa k streaming API vyžaduje udržiavať otvorené prezistentné HTTP spojenie. V mnoho prípadoch to zahŕňa premýšľať o danej aplikácii iným spôsobom, než za použitia REST API. Uvažujme napr. webovú aplikáciu, ktorá prijíma užívateľove požiadavky, pošle jeden alebo viac požiadavkov Twitter API, následne naformátuje a vypíše výsledok užívateľovi ako odpoveď pre jeho pôvodnú požiadavku. Tento scenár je znázornený na obrázku 3.



Obrázok 3.3 – scenár s použitím REST API [19]

Aplikácia, ktorá sa pripája pomocou Streaming API nebude schopná zriadiť spojenie ako odpoveď na užívateľovu požiadavku spôsobom, aký je znázornený na obrázku 2. Namiesto toho je kód pre udržiavanie Streaming spojenia typicky pustený procese, ktorý je oddelený od procesu, ktorý narába s HTTP požiadavkami. Tento scenár znázorňuje obrázok 4.



Obrázok 3.4 – scenár s použitím Streaming API [19]

Proces starajúci sa o streamovanie získava ako vstup tweety a vykonáva potrebné parsovanie, filtrovanie a / alebo agregáciu pred samotným uložením do dátového skladu alebo databáze. Proces, ktorý sa stará o HTTP spojenie sa dotazuje dátoveho skladu pre výsledky na základe užívateľových požiadaviek. Síce je tento model komplexnejší ako predchádzajúci, benefity vyplývajúce zo stremovania tweetov v reálnom čase z neho robia užitočný model pre veľa rôznych typov aplikácií.

3.2.2.1 Verejné toky

Tieto toky ponúkajú vzorky z verejných dát „tečúcimi“ cez Twitter. Akonáhle aplikácia vytvorí spojenie so streamovacím koncovým bodom, je jej dodaný tok tweetov. Výhodou oproti REST API je, že netreba uvažovať a robiť si starosti s obmedzujúcimi limitami a rýchlosťami pri dotazovaní.

Koncové body:

- POST statuses/filter
- GET statuses/sample
- GET statuses/firehose

Spojenia

Každý účet môže vytvoriť iba jedno spojenie k verejným koncovým bodom. Pripojenie sa na verejný tok viac než jedenkrát s tými istými prihlasovacími údajmi spôsobí, že sa najstaršie spojenie odpojí. Klienti, ktorí sa snažia vytvárať nadmerný a neprimeraný počet spojení (či už úspešných alebo neúspešných), riskujú, že ich IP adresa bude automaticky zablokovaná.

3.2.2.2 Užívateľské toky

Užívateľské toky ponúkajú toky dát a udalostí špecifických k danému autentifikovanému užívateľovi. Treba poznamenať, že užívateľské toky nie sú zamýšľané pre spojenia typu server-server. Na vytvorenie spojení v mene viacerých užívateľov z rovnakého stroja je vhodnejšie použiť streamy stránok.

Koncový bod

- GET user

Spojenia

Je vhodné minimalizovať počet pripojení, ktoré aplikácia vytvorí s užívateľskými tokmi. Každý Twitter účet je totiž limitovaný na len niekoľko málo simultánnych spojení pre jednu OAuth aplikáciu, neberúc do úvahy IP adresu. Po vyčerpaní počtu spojení danou aplikáciou, sa najstaršie spojenie ukončí. Účet pripájajúci sa z príliš veľa inštancií tej istej OAuth aplikácie cyklicky strieda spojenia.

Aplikácia musí byť schopná spracovať HTTP chybu 420, ktorá indikuje, že účet sa snaží prihlasovať príliš často. Táto chyba značí, že účet presiahol limitovaný počet spojení pre aplikáciu. Následne sa dá užívateľovi najavo, že ich prístup k užívateľským tokom bol automaticky a dočasne zakázaný. Odporúča sa vypnúť nadbytočné kópie aplikácie, prípadne inštancie bežiace na inom prístroji, aby sa znovu sprístupnilo streamovanie.

Každá aplikácia ma svoju vlastnú alokáciu, takže prihlasovanie sa z Aplikácie 1 neovplyvní Aplikáciu 2 a naopak. Napriek tomu pustených príliš mnoho kópií buď Aplikácie 1 alebo Aplikácie 2 spôsobí problémy. Taktiež treba poznamenať, že počet simultánnych spojení je limitovaný na danú IP adresu bez ohľadu na konkrétnu aplikáciu.

Aplikácie, ktoré podporujú simultánne zobrazenie viacero Twitter účtov môžu otvoriť jedno spojenie pre jeden účet.

3.2.2.3 Dobré návyky

Aplikácia sa uplatní najlepšie, ak bude využívať aj užívateľské streamy spoločne s REST API. Aplikácia, ktorá sa pripája na užívateľské toky môže spätne dopĺňať dáta pomocou REST API, aby tak pokryla časové periódy, kedy z nejakého dôvodu vypadlo spojenie.

Pri prechode z používania REST API na používanie užívateľských tokov je vhodné spraviť posledný REST dotaz po tom, čo prebehlo úspešné pripojenie, čím sa zaistí, že nedošlo k žiadnej strate dát.

Pri odpojení sa z užívateľského toku je dobré sa pokúsiť znovu pripojiť aspoň minútu alebo dve pred tým, než sa bude pokračovať s dotazovaním sa na REST API. Toto oneskorenie pomôže predísť zahlteniu REST API dotazov, keby malo dôjsť k menšiemu zdržaniu alebo iným ťažkostiam u API užívateľských tokov.

Je vhodné sa snažiť nevyužívať celý REST API limit počas streamovania. Odporúča sa používať REST API, akoby to bol vzácny zdroj.

3.2.2.4 Streamy stránok

Streamy stránok ponúkajú služby, ako sú webové stránky alebo mobilné služby, ktoré dostávajú aktualizácie a nové správy pre veľkú skupinu užívateľov. Udalosti môžu byť streamované pre akéhokoľvek užívateľa, ktorý má povolený OAuth prístup do aplikácie. Desktopové aplikácie alebo aplikácie s málo užívateľmi by mali používať užívateľské streamy.

Streamy stránok sú v dobe písania tohto textu v limitovanej beta verzii. Prístup je obmedzený pre skupinu účtov na whiteliste. Ďalšie rozoberanie streamov stránok teda nie je podstatné, nakoľko by neboli využiteľné.

Scoiálna sieť Twitter tak so svojím obrovským počtom denne odoslaných tweetov ponúka možnosti ako získať veľké množstvo materiálu pre získavanie znalostí z dát. Pomocou jeho streamovacieho API sa dá v relatívne krátkom čase získať dostatočne veľká vzorka dát o rôznych témach a názorov ľudí z rôznych lokácií z celého sveta.

3.3 Last.fm

Last.fm je služba pre odporúčanie hudby užívateľom. Užívatelia sa zaregistrujú a stiahnu program The Scrobbler, ktorý im pomáha objavovať novú hudbu na základe toho, aké pesničky užívatelia počúvakú. [24]

Last.fm API ponúka možnosť implementovať programy používajúce Last.fm dáta. Je možné vytvoriť webové alebo desktopové aplikácie alebo aj aplikácie pre mobilné zariadenia. Je treba si vytvoriť API účet, ktorý následne poskytne API kľúč, aby bolo možné využívať webové služby

Last.fm. Odpovede na volania metód tohto API sú vo formáte XML, ktoré je v štýle REST. Koreňová adresa API je http://ws.audioscrobbler.com/2.0/

Koreňovej URL sa posiela parameter s danou metódou, čo je vyjadrené ako 'package.method' spolu s konkrétnymi argumentami. Nasledujúce parametre sú vyžadované pre všetky volania:

api_key : kľúč Last.fm API

method : metóda API vyjadrená ako *package.method*, korešpondujúca pomenovaniu metódy z dokumenácie Last.fm API

napr:

```
http://ws.audioscrobbler.com/2.0/?method=artist.getSimilar&api_key=xxx...
```

API podporuje niekoľko transportných formátov, ale odpovedá implicitne v špeciálnom Last.fm XML formáte. Je taktiež možné dostávať odpovede aj vo formáte JSON.

V ponuke je veľké množstvo metód, pomocou ktorých možno získavať dáta o jednotlivých umelcoch (ich fanúšikoch, udalosti, kde vystupujú, podobných umelcoch atď.), ich albumoch, pesničkách, udalostiach, informácie o užívateľoch, tagoch a iné. Väčšina metód nevyžaduje autentifikáciu. Prevažne sa jedná o metódy, ktoré majú čítací charakter a len vracajú nejaké dáta. Metódy, ktoré majú charakter zápisu naopak vyžadujú autentifikáciu užívateľa. V tomto prípade treba užívateľa poslať na adresu last.fm/api/auth spolu s vyššie spomínaným vygenerovaným API kľúčom ako parameter. Používa sa HTTP GET dotaz, ktorý môže vyzerať napr. takto:

```
http://www.last.fm/api/auth/?api_key=xxx
```

Ak užívateľ nie je prihlásený do Last.fm, bude presmerovaný na prihlasovaciu stránku a následne po prihlásení bude požiadaný o povolenie, aby aplikácia mohla používať jeho účet. Pre potreby dolovania dát by však tento úkon nemal byť dôležitý, čo poskytuje aplikácii dostačujúce možnosti získavania potrebných dát bez zbytočnej réžie.

Pri vytváraní aplikácie, ktorá používa Last.fm API, je vhodné pamätať na niekoľko zásad. Napríklad pri vytváraní webovej aplikácie nie je rozumné dotazovať sa API pri načítaní každej stránky. Mohlo by to mať za následok pozastavenie účtu, keby aplikácia pokračovala v posielaní niekoľkých dotazov za sekundu. Okrem toho treba používať kódovanie UTF-8 pri posielaní argumentov API metódam.

Pre potreby vytvorenia webovej aplikácie využívajúcej Last.fm API existuje viacero, aj keď iba neoficiálnych, možností. Pre jazyk PHP to sú:

- PHP last.fm API
- Felix Brun's PHP libs

Oba prostriedky ponúkajú možnosť cacheovania odpovedí priamo na súborový systém alebo do databázy. Prvý nástroj vracia dáta ako pole, druhý vracia dáta v objektoch.

4 Teória k dolovaniu textových dát

Dolovanie textových dát (knowledge discovery in texts – KDT, alebo často nazývaný aj „text mining", resp. dolovanie z textov) označuje proces získavania znalostí z textu. Tieto informácie sa typicky odvodzujú zo vzorov a trendov v textoch a štatistiky. Dolovanie textových dát obvykle zahrňuje proces spracovania vstupného textu, odvodzovanie vzorov v rámci štruktúrovaných dát a nakoniec vyhodnotenie a interpretácia výstupu. Úlohy dolovania textových dát obvykle zahrňujú kategorizáciu textov, zhlukovanie textov, extrahovanie konceptov alebo entít, sumarizáciu dokumentov a analyzovanie sentimentu. [25]

Proces objavovania znalostí v množine textových dokumentov sa v dôsledku inherentnej neštruktúrovanosti a neurčitosti jazyka javí ako omnoho zložitejší než proces objavovania znalostí v databázach. [26]

Proces dolovania znalostí z textových dokumentov je možné vykonávať spôsobom ako bežný proces objavovania znalostí. Jednotlivé kroky, ktoré tento proces zahrňuje, sú:

Získanie relevantnej množiny dokumentov

Prvým zásadným krokom je získať dostatočne veľkú a relevantnú vzorku dát. Pokiaľ ide o skúmanie konkrétnej aplikačnej domény, je žiaduce, aby táto množina dát pokrývala túto oblasť celú, alebo aspoň v dostatočnej miere. Vzorky sa môžu získať z existujúcich zdrojov buď automatizovaným spôsobom, manuálne alebo kombinovane.

Predspracovanie dát

Ďalším krokom je získané dáta spracovať takým spôsobom, aby sa s nimi dalo pracovať v rámci použitých algoritmov. Z textov sa odstránia nepodstatné informácie ako neplnovýznamové slová, upravia sa veľkosti písmen, prevedú sa rôzne tvary rovnakého slova na jeho spoločný morfologický základ apod. Takto upravené dáta sa prevedú na numerickú formu reprezentácie, s ktorou môže následne vybraný algoritmus pracovať.

Dolovanie v textoch

Tretí krok zahrňuje samotné dolovanie a závisí od vybranej techniky dolovania v textových dátach. Môže ísť o klasifikáciu textov, zhlukovanie, prípadne extrakciu informácií na základe k tomuto účelu vybraného algoritmu.

Vizualizácia a interpretácia výsledkov

Posledným krokom je vizualizovanie a predostretie výsledkov, ktoré boli algoritmom vypočítané. Slúžia na to rôzne nástroje, ktoré môžu dosiahnuté výsledky reprezentovať. Interpretáciu týchto výsledkov musí urobiť už človek sám.

4.1 Tokenizácia

Tokenizácia je proces rozloženia toku textu do slov, fráz, symbolov alebo iných zmysluplných elementov nazývaných tokeny. Tokeny možno definovať ako systémom rozpoznané a akceptované skupiny znakov s kolektívnym významom. Zoznam tokenov sa používa ako vstup pre ďalšie spracovanie v dolovaní textových dát.

Vzhľadom na využitie tokenizácie na predspracovanie a následné dolovanie znalostí z textov pomocou klasifikačných a zhlukovacích algoritmov je vhodné zohľadniť isté javy a špeciálne tvary lexikálnych jednotiek a riešiť ich pomocou adekvátneho návrhu systému tokenizačných pravidiel. Sú to najmä [27]:

- Čísla – čísla a zmiešané alfanumerické reťazce väčšinou nie sú dobrými termami vo vektorovej reprezentácii dokumentov, a to kvôli ich neurčitosti.
- Zloženie slova – je otázkou, či slová so spojovníkom pri tokenizácii spojiť, alebo rozdeliť do viacerých tokenov. Možným riešením je preskúmať, či sa textové jednotky pred a za spojovníkom nachádzajú v slovníku prípustných tvarov.
- Veľkosť písma – znaky sa často v procese tokenizácie konvertujú buď na veľké, alebo na malé písmo, čo však môže niekedy znamenať stratu sémantickej informácie.

4.2 Izolácia koreňa slova

Proces, ktorý z tvaru slova v texte určí jeho základný tvar sa nazýva **lematizácia.** Cieľom je zoskupiť rôznym spôsobom vyskloňované tvary slova, aby mohli byť analyzované ako jedna položka [28].

Špeciálnou formou lematizácie je **izolácia koreňa slova** (angl. „**stemming**“). Ide o jednoduchší prístup získania koreňa, ktorý je možné ľahšie implementovať a má menšie časové nároky. Takto získaný koreň sa nazýva „**stem**“. Tento stem nemusí byť identický s morfologickým základom slova; väčšinou je dostačujúce, že súvisiace slová sa namapujú na rovnaký stem aj napriek tomu, že daný stem nie je validný morfologický koreň slova. [26, 29]

V angličtine nie je izolácia koreňa tak náročná a najpoužívanejší je **Porterov algoritmus**. [26] Je založený na odstraňovaní prípon, pričom využíva pevný zoznam prípon a niektoré ďalšie pravidlá morfológie anglického jazyka. V implementácii som využil práve tento algoritmus, ktorý bol pre jazyk PHP verejne dostupný a ľahko zakomponovateľný do webovej aplikácie. [30]

Pre ilustráciu uvediem niekoľko príkladov, ktoré budú názornejšie vo vysvetlení, ako je proces stematizácie uskutočňovaný. V práci sa zameriavam len na anglický jazyk a tak sú nasledujúce príklady ukázané tiež na anglických slovíčkach. Algoritmus pre anglický jazyk by mal teda identifikovať reťazec „cats“ (a prípadne reťazec „catlike“, „catty“ atď.) ako odvodený od základu „cat“ a reťazce „stemmer“, „stemming“, „stemmed“ ako odvedené od základu „stem“. Algoritmus stematizácie redukuje slová „fishing“, „fished“, „fish“ a „fisher“ na základ slova „fish“, no na druhej strane slová „argue“, „argued“, „argues“, „arguing“ a „argus“ sa všetky redukujú na stem „argu“ (čo ilustruje práve prípad, v ktorom daný stem nie je ani samotné slovo ani morfologický základ), ale „argument“ a „arguments“ sa redukujú na stem „argument“.

Proces stemming je dôležitý krok predspracovania, nakoľko sú po jeho aplikácii rôzne gramatické tvary slova identifikované a indexované (započítané) ako jedno a to isté slovo.

4.3 Eliminácia neplnovýznamových slov

Všetky identifikované a lematizované tokeny sú kandidátmi pre termy vektorového modelu textového dokumentu. Termy, čiže kľúčové slová, by mali čo najpresnejšie vyjadrovať obsah daného dokumentu. Zároveň je pre efektívnosť ďalšieho spracovania pomocou klasifikačných a zhlukovacích algoritmov výhodné minimalizovať počet termov popisujúcich jednotlivé dokumenty. Tieto dve skutočnosti sú dôvodmi pre to, aby sa z ďalšieho spracovania vylúčili tokeny s malým príspevkom k celkovému obsahu textu. Vo všeobecnosti sa predpokladá, že hlavnými nositeľmi obsahu textu sú plnovýznamové slová, predovšetkým substantíva a adjektíva. Lematizované tokeny, ktoré vznikli z týchto slov, sú vhodné na reprezentáciu obsahu dokumentov a majú sa transformovať na termy. Naopak, minimálny a zanedbateľný prínos k obsahu textu sa predpokladá pri neplnovýznamových slovách – spojkách, predložkách, zámenách, časticiach, rôznych netextových elementoch a sčasti aj pri čísliciach a číslovkách. Tieto takzvané stop-slová (angl. stop-words, niekedy tiež noise-words) sa obyčajne v texte vyskytujú s vyššou frekvenciou, pričom však k celkovému obsahu textu prispievajú iba malým informačným ziskom. Tokeny vzniknuté z týchto slov je vhodné vylúčiť zo zoznamu termov. [26] Typické stop-slová pre angličtinu sú napríklad:

a, about, above, after, again, an, and, any, are, be, before, behind, been, both, brief, can, come, did, didn't, down, during, each, else, et, etc, except, far, for, from, get, go, got, had, half, has, have, he, hello, his, how, in, into, it, just, last, let, low, me, most, much, my, near, no, none, not, now, of, off, on, our, out, own, past, per, rather, recent, say, see, self, she, so, soon, such, take, than, that, the, their, then, there, they, this, to, too, try, under, up, upon, us, use, via, want, was, we, were, what, when, who, why, would, yes, yet, ...

Štandardizovaný a všeobecne akceptovaný zoznam stop-slov neexistuje, pretože zaradenie vhodného slova medzi stop-slová je častokrát závislé od aplikácie alebo domény, ktorej sa analyzované texty týkajú.

Jedným zo spôsobov odstraňovania stop-slov je taký, že sa popredu ručne pripraví zoznam týchto neplnovýznamových slov, ktorý sa nazýva tzv. negatívny slovník (angl. stop-word list). Z textov sa odstránia slovíčka, ktoré sa nachádzajú v tomto slovníku. Nevýhoda vyplývajúca z tohto prístupu je tá, že je závislý na konkrétnom jazyku, ktorý sa v textoch používa. Okrem toho samozrejme dochádza k istej strate informácií zo získaných vzoriek textu, avšak túto nevýhodu dostatočne vyvažuje skutočnosť, že vďaka tejto redukcii sa znižuje príznakový priestor, pomocou ktorého sú dokumenty reprezentované. Vďaka tomu sa zefektívni práca dolovacích algoritmov, či už ide o klasifikáciu, prípadne zhlukovanie apod. Tento prístup vopred definovaného slovníka som zvolil a použil v tomto projekte s množinou obsahujúcou anglické neplnovýznamové slovíčka.

4.4 Klasifikácia textových dokumentov

Klasifikácia textových dokumentov je úloha, v ktorej ide o to zaradiť dokument do jednej alebo viacerých tried alebo kategórií. Pre tento projekt bolo treba preskúmať automatické tj. algoritmické spôsoby klasifikácie dokumentov.

4.4.1 Automatická klasifikácia dokumentov

Ako bolo už načrtnuté v kapitole 2, úlohy automatickej klasifikácie dokumentov môžu byť rozdelené na tri rôzne typy:

- **klasifikácia dokumentov s kontrolovaným učením**, kde je potrebné, aby existoval nejaký externý mechanizmus (ako napr. spätná väzba od človeka), ktorý sprostredkováva pre dokumenty správne zaradenie do tried.
- **klasifikácia dokumentov s nekontrolovaným učením** (tiež známa ako zhlukovanie dokumentov), kde musia byť dokumenty klasifikované bez akejkoľvek referencie alebo odkazu na externé informácie.
- **klasifikácia dokumentov s polo-kontrolovaným (zmiešaným) učením**, kde sú časti dokumentu označené vonkajším mechanizmom

Pre účely klasifikácie dokumentov existuje množstvo v praxi používaných techník. Niektoré sú vypísané v nasledujúcom zozname a zahrňujú:

- Naivný Bayesovský klasifikátor (angl. Naive Bayes)
- Metóda zhlukovania dát (angl. Expectation Maximization, EM)

- TF-IDF
- Latentné sémantické indexovanie (angl. Latent semantic indexing, LSI)
- SVM (angl. skratka pre „Support Vector Machine“)
- Neurónové siete
- Algoritmus K-najbližších susedov
- Rozhodovacie stromy

Ďalšie časti práce sa zameriavajú na klasifikáciu s kontrolovaným učením. Pri tomto spôsobe algoritmizácie je množina dát rozdelená do dvoch častí: trénovacia a testovacia, čo bude vysvetlené neskôr.

4.5 Modely reprezentácie textových dokumentov

Modelom reprezentácie textových dokumentov sa nazýva taká formalizácia dokumentu, resp. jeho textu v prirodzenom jazyku, ktorá nejakým spôsobom vyjadruje obsah tohto textu a zároveň umožňuje ich efektívne spracovanie metódami dolovania v textoch, predovšetkým algoritmami klasifikácie a zhlukovania. Významnou vlastnosťou a predpokladom aplikovateľnosti modelov reprezentácie textov je možnosť matematicky porovnávať texty podľa ich vzájomnej podobnosti, príbuznosti ich obsahov [31].

Východiskom konštrukcie rôznych modelov reprezentácie textových dokumentov pre úlohy získavania znalostí z textov je matica dokument-term. Texty z dokumentov v analyzovanom korpuse sa predspracujú postupmi popísanými v predchádzajúcich častiach čiže konverziou na čistý text, segmentáciou, tokenizáciou, lematizáciou, morfologickou a prípadne aj komplexnou jazykovou analýzou. Pre každý z dokumentov sa získa zoznam (vektor) lematizovaných termov, ktoré vyjadrujú obsah daného dokumentu. Problémy pri tejto reprezentácii môže spôsobovať potenciálne veľký počet termov, ktorý negatívne ovplyvňuje efektivitu zhlukovacích a klasifikačných algoritmov, a tiež nerovnomerné zastúpenie termov vzhľadom na ich príspevok k celkovému obsahu dokumentu resp. kolekcie. Prvý problém, teda zníženie počtu termov v reprezentácii dokumentu, sa rieši odstránením neplnovýznamových slov.

Matica dokument-term je vo všeobecnosti riedka matica, obsahujúca veľa núl na pozíciách termov, ktoré sa nenachádzajú v textoch niektorých dokumentov z korpusu. Preto sa pre úlohu vyhľadávania informácií používa upravená matica dokument-term, ktorá pre každý z termov uchováva iba zoznam tých dokumentov, v ktorých sa daný term vyskytuje. Takáto reprezentácia textu sa nazýva invertovaný index (alebo skrátene index), proces transformácie dokumentu (textu) do tejto reprezentácie sa označuje ako indexácia..

Existuje niekoľko modelov reprezentácie textových dokumentov, medzi ktoré patrí **Boolovský model**, **pravdepodobnostný model** a **vektorový model**. Boolovský model používa binárne ohodnotenie prítomnosti resp. absencie slova v texte dokumentu. Pravdepodobnostný model vychádza z Boolovského modelu a tiež používa binárne ohodnotenie, líši sa však vo výpočte vzájomnej podobnosti dvoch textov. Najpoužívanejší model pri získavaní znalostí z textov je model vektorový [26], ktorý som využil v implementácii aj v (pred samotnou implementáciou) uskutočnených testoch v programe RapidMiner a tento model bude rozobratý podrobnejšie.

4.5.1 Vektorový model

Jeden z nedostatkov Boolovského modelu je, že nie je spôsob ako zhodnotiť, ktoré slová sú v ktorom dokumente relevantnejšie. Ak by však bolo možné ohodnotiť termy v dokumente na základe toho, v akej miere sú reprezentatívne v dokumente ako celku, mohli by sme presnejšie zoradiť dokumenty podľa toho, ako veľmi sa k sebe podobajú. Táto myšlienka formuje základ pre **vektorový model.**

Vektorový model (angl. „Vector Space Model“, VSM) je algebraický model reprezentácie textových dokumentov ako vektorov identifikátorov, ako napr. indexovaných termov. Dokumenty sú reprezentované ako vektor:

$$d_j = (w_{1,j}, w_{2,j}, \dots, w_{t,j})$$

Každá dimenzia vyjadruje odlišný term. Ak sa term nachádza v dokumente, jeho hodnota vo vektore je nenulová. Bolo vyvinutých niekoľko rôznych spôsobov ako počítať tieto hodnoty, zvané tiež ako váhy termov. Jedna z najznámejších schém je **TF-IDF** váhovanie.

4.5.2 TF-IDF

Pre ohodnotenie slov dokumentov váhami som použil metódu TF-IDF. Skratka **TF–IDF** stojí pre anglický termín „**term frequency–inverse document frequency**“, čo by sa dalo preložiť ako „frekvencia výrazu-inverzná dokumentová frekvencia“. Ide o numerickú štatistickú metódu, ktorá reflektuje ako veľmi je dané slovo dôležité v rámci daného dokumentu z kolekcie dokumentov. Každé slovo nachádzajúce sa v dokumente je teda ohodnotené číselnou hodnotou TF-IDF. Táto hodnota sa priamo úmerne zvyšuje s počtom koľkokrát je dané slovo obsiahnuté v dokumente, ale je vyrovnaná častosťou tohto slova v celej kolekcií dokumentov. To napomáha kontrolovať fakt, že niektoré slová sú obecne bežnejšie, než ostatné. [32]

Hodnota TF-IDF je súčin dvoch štatistických údajov, frekvencie slova a inverznej dokumentovej frekvencie. Existujú rôzne spôsoby pre určovanie oboch štatistických hodnôt. V prípade frekvencie termu $tf(t,d)$, najjednoduchším výberom je použitie hrubej frekvencie termu v dokumente, tj. počet koľkokrát sa term t nachádza v dokumente d. Ak označíme hrubú frekvenciu termu t ako $f(t,d)$, potom jednoduchá TF schéma bude $tf(t,d) = f(t,d)$.
Existujú ešte ďalšie možnosti:

- Boolovské frekvencie: $tf(t,d) = 1$ ak sa term *t* nach8dza v dokumente *d*, v opačnom prípade $tf(t,d) = 0$
- Logaritmická frekvencia: $tf(t,d) = 1 + \log \mathrm{f}(t,d)$ a 0 keď $f(t,d) = 0$
- Normalizovaná frekvencia, ktorá slúži na to, aby sa predišlo skresleniu pri dlhších dokumentoch, napr. hrubá frekvencia termu vydelená hrubou frekvenciou termu s maximálnou frekvenciou v danom dokumente: $tf(t,d) = \frac{f(t,d)}{max\{f(w,d):\, w \in d\}}$

Inverzná dokumentová frekvencia je miera toho, či je daný term naprieč všetkými dokumentmi bežný alebo naopak vzácny. Je získaná vydelením celkového počtu dokumentov počtom dokumentov obsahujúcich daný term a následným zlogaritmovaním výsledného podielu:

$idf(t,D) = log\, \frac{|D|}{|\{d \in D:\, t \in d\}|}$ kde:

- $|D|$ značí mohutnosť *D*, alebo celkový počet skúmaných dokumentov
- $|\{d \in D : t \in d\}|$: počet dokumentov obsahujúcich term *t* (tj. $tf(t,d) \neq 0$). Ak sa term v korpuse dokumentov nenachádza, viedlo by to k deleniu nulou. Je teda bežnou praxou upraviť rovnicu na $1 + |\{d \in D : t \in d\}|$.

Potom sa TF-IDF vypočíta ako:

$$idf(t,d,D) = tf(t,d) \times idf(t,D)$$

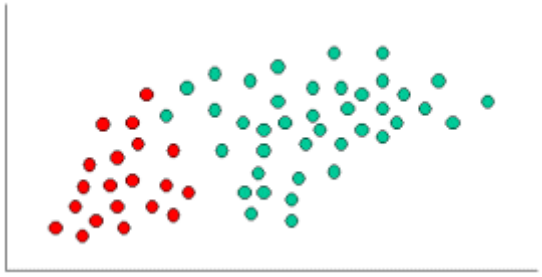
Logaritmus slúži na isté zahladenie výsledkov. Ak je napr. term A zastúpený v dokumente X práve raz a term B je v tom istom dokumente zastúpený dvakrát, potom je term A špecifickejší a mal by poskytnúť lepšie výsledky, ale nie nevyhnutne dvakrát lepšie výsledky. Logaritmus tieto rozdiely zahladí. [33]

Vysoká váha v TF-IDF je dosiahnutá vysokou frekvenciou termu v danom dokumente a nízkou dokumentovou frekvenciou termu v celkovej kolekcii dokumentov. Váhy teda majú tendenciu odfiltrovávať bežné termy. Nakoľko pomer vnútri IDF logaritmu je vždy väčší než alebo rovný 1, hodnota IDF (a TF-IDF) je väčšia než alebo rovná 0. Ako sa term objavuje vo viacerých dokumentoch, pomer vnútri logaritmu sa približuje 1, čím sa hodnoty IDF a TF-IDF blížia k 0.

4.5.3 Naivný bayesovský klasifikátor

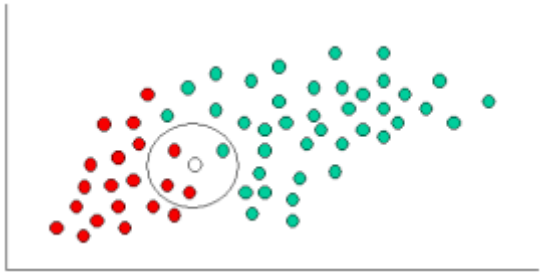
Prvý algoritmus, ktorý som použil na experimentovanie a následné porovnanie s ostatnými algoritmami, sa nazýva Naivný bayesovský klasifikátor. Je založený na tzv. Bayesovom teoréme a je obzvlášť vhodné ho použiť, keď je vysoká dimenzionalita vstupných dát. Ide o jednoduchú metódu klasifikácie, no napriek svojej jednoduchosti vie tento klasifikátor často poskytnúť presnejšie výsledky než iné sofistikované klasifikačné metódy. [34]

Demonštráciu princípu Naivného baysovského klasifikátoru ukážem na nasledujúcom príklade.



Obr. 4.1 – príklad klasifikovaných vzoriek pre naivný bayesovský klasifikátor

Ako vidno z obr. 4.1, klasifikované objekty môžu byť zelené alebo červené. Nakoľko je v príklade dvakrát viac zelených objektov než červených, je dvakrát viac pravdepodobné, že nový objekt bude tiež zelený. V bayesovskej klasifikácii táto skutočnosť figuruje ako tzv. *apriórna pravdepodobnosť*. Ide o pravdepodobnosť, s akou bude nový neznámy objekt klasifikovaný do jednej z tried. Aprióna pravdepodobnosť je teda založená na predchádzajúcej skúsenosti.



Obr. 4.2 – neznáma vzorka pre naivný bayesovský klasifikátor

Pre klasifikáciu neznámeho objektu (biely krúžok na obr. 4.2) je rozumné uvažovať, že čím viac zelených (alebo červených) objektov je v okolí nového objektu, tým väčšia pravdepodobnosť, že nový objekt bude práve tejto farby. Pre nový objekt teda existujú dve rôzne pravdepodobnostné informácie a výsledná klasifikácia sa robí pomocou kombinácie oboch týchto pravdepodobností do tzv. *posteriórnej pravdepodobnosti* za použitia tzv. Bayesovho teorému.

4.5.3.1 Bayesov teorém

Nech X a Y sú dva obecne rôzne javy. Symbolom $P(X|Y)$ sa značí podmienenú pravdepodobnosť javu X za podmienky výskytu javu Y. Táto hodnota udáva, aká je pravdepodobnosť, že nastane jav X, ak bezpečne vieme, že nastal jav Y. Podmienená $P(X|Y)$ sa vypočíta pomocou vzorcu:

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)}$$

kde $P(X \cap Y)$ je pravdepodobnosť, že nastanú javy X a Y súčasne; $P(Y)$ je pravdepodobnosť javu Y. [3] Hodnota $P(X|Y)$ nie je obecne zhodná s hodnotou $P(Y|X)$. Po úpravách sa dostane vzťah, ktorý sa nazýva Bayesovský vzorec:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

4.5.4 Klasifikátor založený na k-najbližšom susedstve

Ďalší algoritmus, ktorý som otestoval na účely klasifikácie, je klasifikátor, ktorý funguje na princípe k-najbližšieho susedstva (angl. „k-nearest neighbors“ alebo skratkou „k-NN“). Ide o jeden z najjednoduchších algoritmov spomedzi algoritmov s učením s učiteľom. V tomto prístupe sa počíta, že každá vzorka dát (každý textový dokument) je reprezentovaný *n*-ticou atribútov číselnej hodnoty. Vzorky sú teda obsiahnuté v n-dimenzionálnom priestore.

Hodnota *k* v názve algoritmu udáva počet najpodobnejších vzoriek, ktoré sa vyberú vzhľadom k práve klasifikovanej vzorky. Vyberie sa teda *k* najbližších vzoriek a trieda práve klasifikovanej vzorky sa určí podľa triedy najpočetnejšie obsiahnutej v tomto susedstve najbližších vybraných prvkov.

Na to, ako zistiť mieru podobnosti dvoch vzoriek, existuje niekoľko (nie len) numerických metód. Na experimenty som využil dve numerické metódy: **euklidovskú vzdialenosť** a **kosínovú podobnosť**.

4.5.4.1 Euklidovská vzdialenosť

Medzi dvoma vzorkami $X = (x_1, x_2, \dots, x_n)$ a $Y = (y_1, y_2, \dots, y_n)$ je definovaná tzv. **Euklidovská vzdialenosť**, ktorá sa dá vypočítať podľa nasledujúceho vzorca [3]:

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

4.5.4.2 Kosínová podobnosť

Ďalšou možnosťou porovnania dvoch textových dokumentov, ktorú som zvolil, bola kosínová podobnosť, ktorá lepšie zodpovedá významovej podobnosti dokumentov [26]. Je to miera podobnosti dvoch vektorov, ktorá sa získa vypočítaním kosínusu uhla týchto vektorov. Ide o skalárny súčin vektorov, vydelený súčinom ich veľkostí.

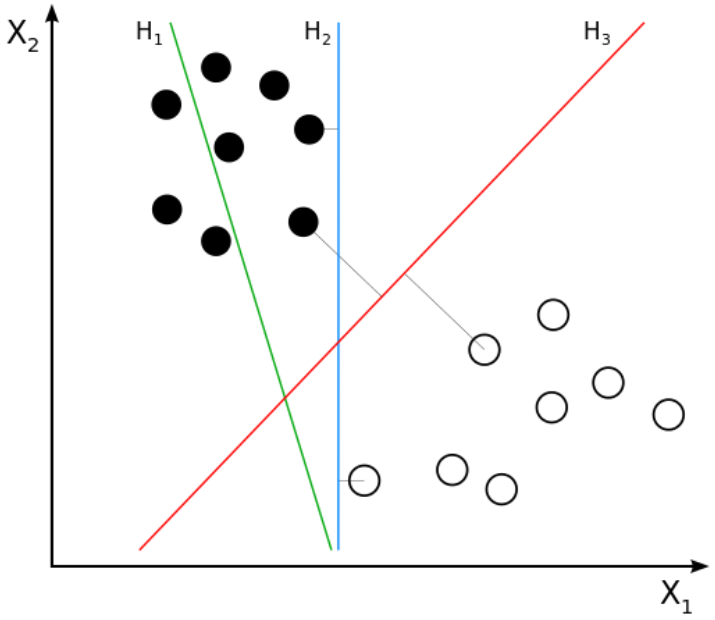
Posledným zo skúmaných algoritmov pre klasifikáciu je najzložitejší a najpodstatnejší, nakoľko je to algoritmus, ktorý bol využitý v implementácii, a preto mu je venovaná zvlášť podkapitola. Algoritmus je označovaný skratkou SVM (angl. skratka pre „Support Vector Machine“).

4.6 SVM

SVM (angl. „Support Vector Machine“) je algoritmus, ktorý je založený na štatistickej teórii učenia sa – lineárnej a nelineárnej regresii. Analyzuje dáta a rozpoznáva vzory, používa sa na klasifikáciu a regresiu. Pre prípad tejto práce sa využije jeho schopnosť klasifikácie do tried. Na vstup berie množinu vstupných dát a pre každú vstupnú vzorku predikuje, do ktorej z dvoch možných tried vzorka patrí. Jedná sa teda o ne-pravdepodobnostný binárny lineárny klasifikátor. Majúc množinu trénovacích vzoriek, kde každá je označená ako patriaca do jednej z dvoch kategórií, trénovací algoritmus SVM vybuduje model, ktorý priraďuje nové vzorky do jednej alebo druhej kategórie. SVM model je reprezentáciou vzoriek ako bodov v priestore namapovaných takým spôsobom, že vzorky jednotlivých tried sú oddelené priestorom, ktorý je čo najširší. Nové vzorky sú teda namapované do tohto istého priestoru a predikcia ich triedy je určená na základe toho, na ktorú stranu od spomínaného oddeľovacieho priestoru spadajú. [35]

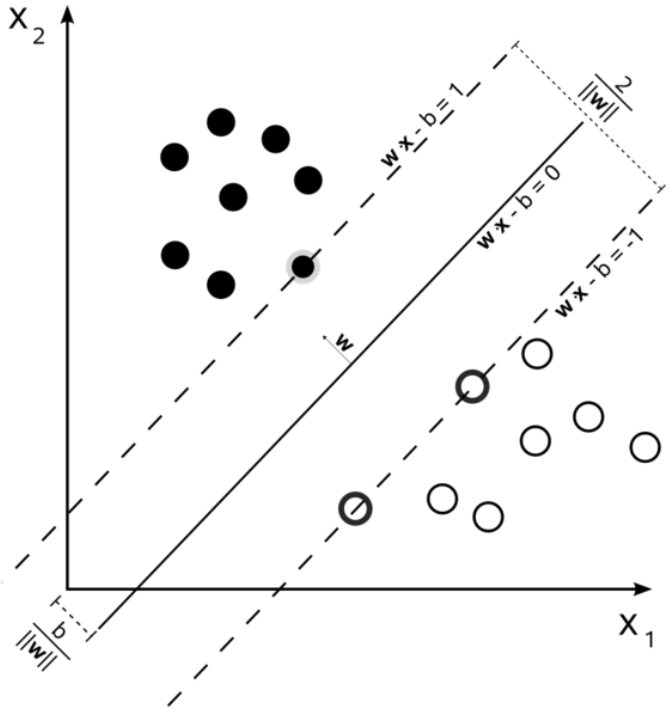
Okrem lineárnej klasifikácie sú SVM schopné efektívne vykonávať aj nelineárnu klasifikáciu pomocou techniky, ktorá sa nazýva anglicky ako tzv. „kernel trick”, kde sa implicitne mapujú vstupné vzorky do viacdimenzionálneho priestoru.

V praxi teda SVM skonštruuje hyperplochu, alebo množinu hyperplôch v multidimenzionálnom priestore. Intuitívne, dobrá separácia jednotlivých kategórií je dosiahnutá tak, že oddeľujúca hyperplocha je skonštruovaná tak, že má čo možno najväčšiu vzdialenosť k najbližšej vzorke trénovacej množiny jednotlivých tried. Tento okraj sa nazýva angl. „functional margin“ Obecne teda platí, že čím širší okraj, tým menšia chybovosť v klasifikácii. Tento princíp bližšie znázorňuje obr. 4.3, kde okraj H1 neoddeľuje triedy, okraj H2 ich oddeľuje s malým okrajom a okraj H3 ich oddeľuje s najväčším možným okrajom.



Obrázok 4.3 - možnosti oddelenia vzoriek tried [45]

Na obrázku 4.4 je znázornená oddeľujúca hyperplocha a okraje pre SVM, ktoré bolo natrénované na základe vzoriek z dvoch rôznych tried. Vzorky nachádzajúce sa priamo na okrajoch tejto hyperplochy sa nazývajú podporné vektory (angl. „support vectors“).



Obrázok 4.4 – znázornenie oddeľujúcej hyperplochy a podporných vektorov [45]

4.6.1 Klasifikácia do viac tried pomocou SVM

Bolo spomínané, že SVM je binárny klasifikátor, a teda klasifikuje vzorky do jednej z dvoch tried. Klasifikácia SVM do viac tried (angl. multi-class SVM classification) prideľuje vzorkám jednu z konečného počtu tried. Hlavný prístup, ako toto docieliť je ten, že sa problém klasifikácie do viacero tried rozloží do niekoľkých problémov binárnej klasifikácie. Toto je bežný prístup a funguje tak, že sa vytvorí niekoľko binárnych klasifikátorov, ktoré rozlišujú medzi označením jednej triedy a zvyškom tried (angl. **one-versus-all**) alebo medzi každou dvojicou tried (angl. **one-versus-one**). V prvom prístupe sa klasifikácia robí spôsobom „víťaz berie všetko“, kde klasifikátor s najvyšším výsledkom udelí vzorke konkrétnu triedu. Tu je dôležité, aby výstupné funkcie klasifikátorov boli nakalibrované tak, aby produkovali medzi sebou porovnateľné výsledky. V druhom prípade sa klasifikácia robí tak, že každý klasifikátor udelí vzorku jednu z dvoch tried a následne sa vzorka zaradí do triedy, ktorá mala najväčší počet hlasov. [36] Existujú ešte iné spôsoby SVM klasifikácie do viacerých tried, no tie sú nad rámec tejto práce, nakoľko ich použitá knižnica nepodporuje.

4.6.2 Nelineárna klasifikácia

Nie vždy je možné rozdeliť vzorky dvoch tried v priestore priamkou, v trojrozmernom priestore plochou, prípadne v N-dimenzionálnom priestore hyperplochou. V niektorých prípadoch by bolo nutné takéto rozdelenie urobiť krivkou (nelineárnym regiónom). Namiesto toho, aby sa k dátam algoritmus snažil vypočítať a dosadiť nelineárne plochy, SVM rieši túto situáciou pomocou tzv. *kernel funkcie*, ktorá mapuje dáta z jedného priestoru do iného priestoru, kde je možné použiť k separácii jednotlivých tried hyperplochu. Aby bolo možné vykonať túto separáciu, kernel funkcia môže previesť dáta do viacdimenzionálneho priestoru, kde môže byť separácia jednoduchšia.

Je možné využiť mnoho kernel funkcií, ale pri zopár konkrétnych sa ukázalo, že pracujú dobre v širokom množstve prípadov. Medzi najpoužívanejšie kernelové funkcie patrí:

- lineárny kernel
- polynomiálny kernel
- tzv. „radial basis function“
- tzv. sigmoid kernel

5 Získavanie dát

Pre výber jazyka PHP na samotné získavanie dát zo sociálnej siete Twitter som sa rozhodol na základe toho, že je v ňom napísaných niekoľko rôznych obľúbených, na tieto účely bežne používaných nástrojov. Aby som sa nemusel Twitter Streaming API dotazovať ručne, využil som PHP rozhranie Phirehose a framework nad týmto rozhraním nazvaný 140dev Twitter Framewok.

5.1 Phirehose

Phirehose je PHP rozhranie k Twitter Streaming API. Ide o knižnicu, pomocou ktorej je možné sa z PHP aplikácii ľahko pripojiť k Streaming API a ukladať z tohto toku v reálnom čase sa objavujúce novo vytvorené tweety. Táto knižnica umožňuje prichádzajúce statusy zaraďovať do fronty, smeruje k tomu, aby bola implementovaná presne na základe odporúčaní z Twitter Streaming API dokumentácie, pričom využíva minimum času procesoru. [37]. Je treba poznamenať, že táto knižnica je určená pre používanie iba v prostredí príkazového riadku, tzn. nie pre vsadenie do webového skriptu, ani pustenie v internetovom prehliadači.

Okrem pripojenia a autentifikácie do Streaming API a zaraďovania tweetov do fronty podľa zvoliteľného prístupu, Phirehose za programátora tiež rieši opätovné pripájanie sa pri možných výpadkoch a chybách spomínaného API. Udržuje teda stále spojenie a navyše monitoruje vzniknuté chyby a výkonnostné metriky.

Knižnica Phirehose však žiadnym spôsobom nedekóduje a nespracováva samotné tweety a ani neposkytuje funkcionalitu, ktorá sa netýka priamo Streaming API (napr. informácie z užívateľských účtov, vyhľadávanie atp.)

5.2 140dev Twitter Framework

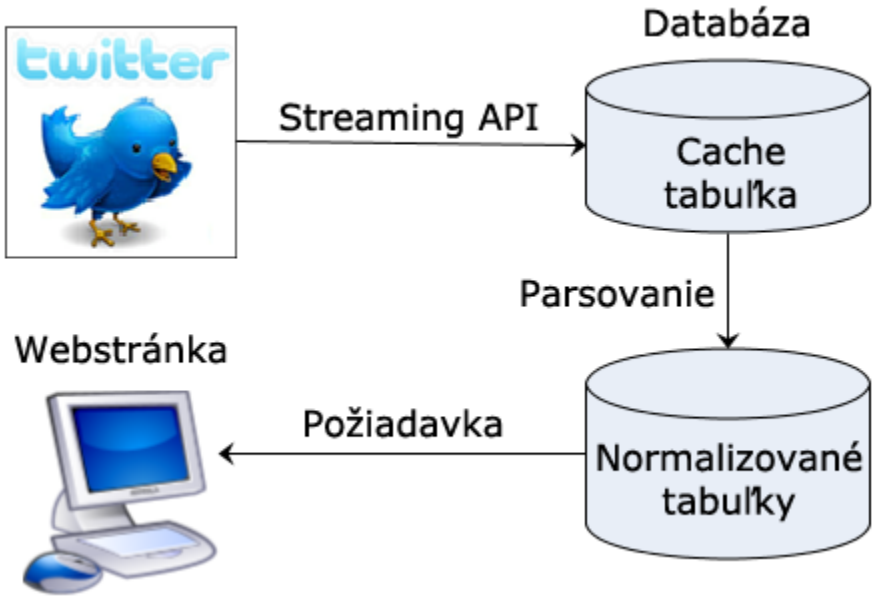
Na dekódovanie tweetov a ich následné spracovávanie som využil voľne dostupnú knižnicu, ktorej cieľom je ešte výraznejšie zjednodušenie prístupu k Twitter API. Je napísaná v PHP a JavaScripte, pre ukladanie dát využíva databázu MySQL a pre autentifikáciu OAuth. [39] Jadro tohto frameworku tvorí Twitter Database Server, ktorý využíva Streaming API pre agregovanie tweetov obsahujúcich vybrané kľúčové slová. Entity objavujúce sa v tweetoch sú ukladané do separátnych tabuliek pre optimalizovanie výkonu dolovania dát. Tweet môže obsahovať nasledujúce entity:

- hashtag - slovo alebo fráza, ktorá začína znakom # (dvojitý krížik), napr. #WeHaveAPope. Pomocou hashtagov je možné dohľadať rôzne trendy (nový film, album, dôležitú udalosť), prípadne označiť význam príspevkov (sarkazmus, hnev atď.)

- URL tj. webových adries
- @mention – použitie akejkoľvek prezývky v tele tweetu v tvare „@prezývka”

5.2.1 Architektúra frameworku

Framework je postavený spôsobom, ktorý oddeľuje proces získavania nových tweetov od prezentačnej vrstvy, ktorá zobrazuje vhodné dáta na webovej stránke, prípadne vrstvy, ktorá môže dáta iným spôsobom ďalej spracovávať.



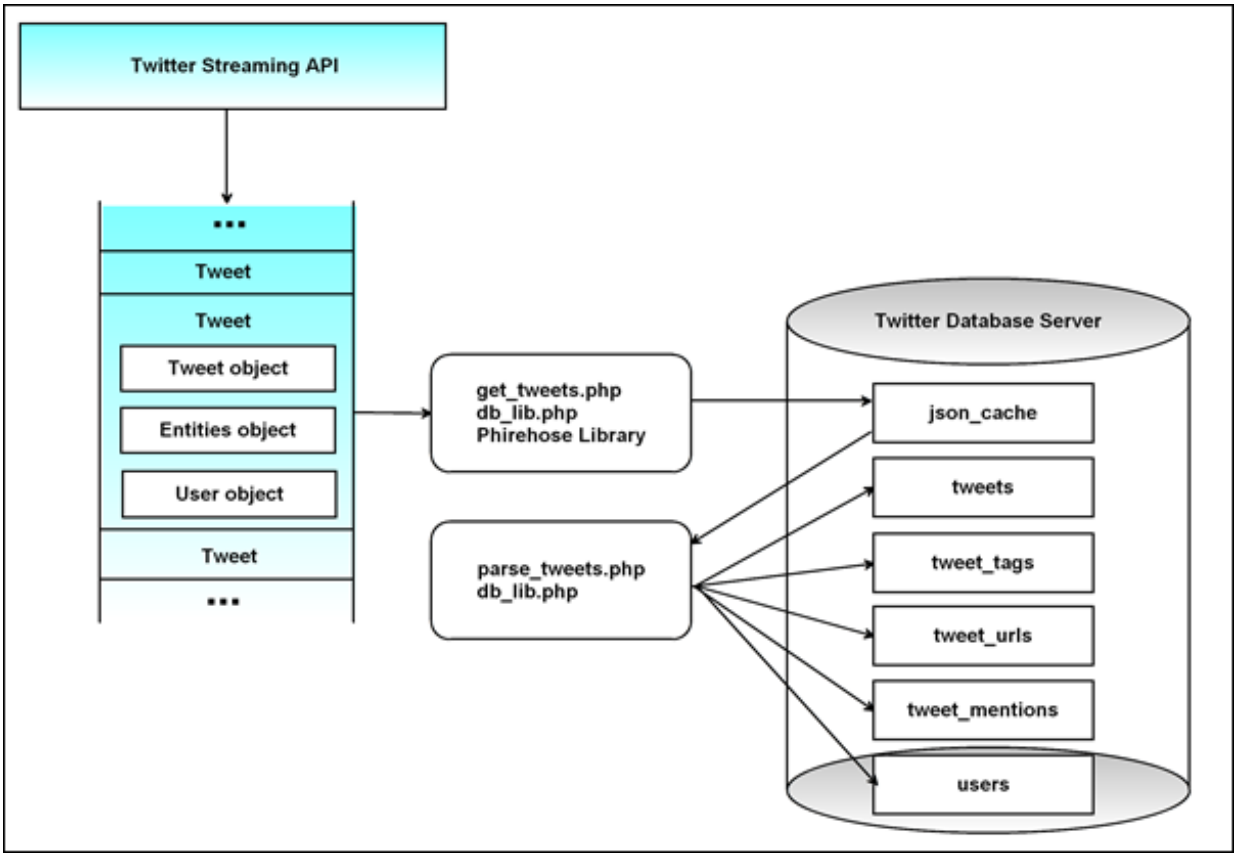
Obrázok 5.1 – architektúra frameworku 140dev [39]

Takáto architektúra vytvorí vrstvu medzi Twitter API a užívateľským rozhraním. Na jednej strane je API a kód, ktorý zberá dáta. Na druhej strane je kód, ktorý dodáva výsledok užívateľom ako webová aplikácia. V strede je databázový dizajn a programovacie techniky, potrebné pre uloženie dát tweetu do normalizovanej databázovej schémy, ktorá je optimalizovaná pre dotazy, ktoré daná aplikácia potrebuje.

Väčšinu práce ohľadom získavania tweetov vykonáva knižnica Phirehose, ktorá je nainštalovaná spolu s 140dev frameworkom. Aby bolo možné využívať Phirehose, je nutné vytvoriť PHP skript, ktorý rozšíri triedu Phirehose a pridá kód, ktorý ukladá tweety hneď pri ich prijatí. Túto úlohu rieši skript **get_tweets.php**, ktorý je púšťaný ako nepretržitý proces na pozadí.

Skript **parse_tweets.php** spracováva každý nový tweet, ktorý je pridaný do tabuľky **json_cache** skriptom get_tweets.php. Dáta z tweetov sú extrahované zo stĺpca **raw_tweet** v tabuľke json_cache a následne rozložené do tabuliek: **tweets**, **tweet_tags**, **tweet_urls**, **tweet_mentions** a

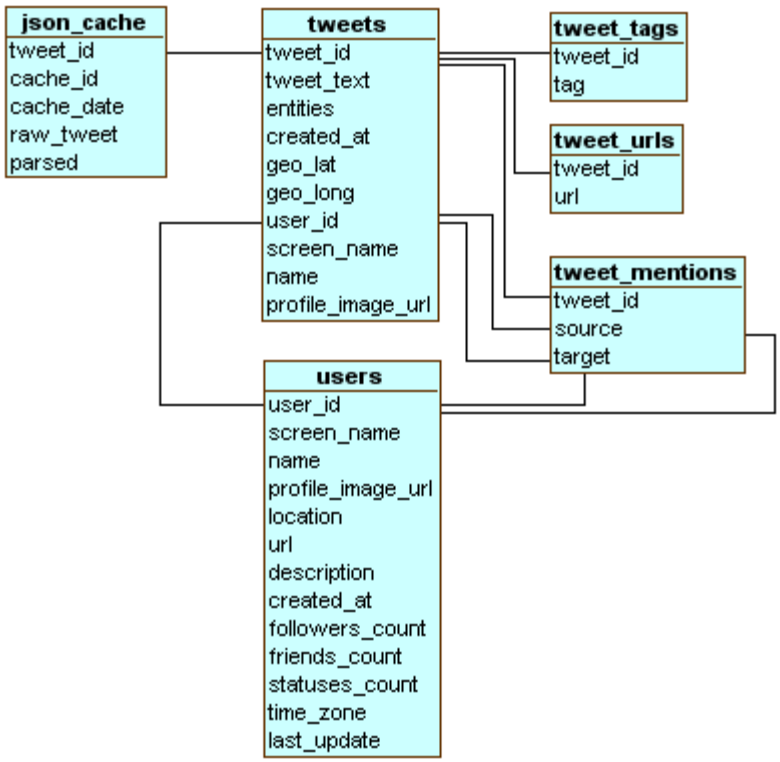
users. Ak je užívateľ už obsiahnutý v tabuľke users, jeho dáta sú aktualizované dátami z najnovšieho tweetu.



Obrázok 5.2 – architektúra kódu frameworku 140dev [38]

5.2.2 Databázová schéma

Pri štruktúre databázy je vhodné si všimnúť, že každý riadok z tabuľky json_cache obsahuje jednoznačný identifikátor tweetu **tweet_id**. Vďaka tomuto je možné sa v budúcnosti vrátiť a znovu rozparsovať výsledky API dotazovania sa. Toto je výhoda, pretože Twitter pravidelne mení štruktúru výsledkov získaných pomocou Streaming API. Sú pridávané nové hodnoty, prípadne existujúce políčka zmenia názov alebo dátový typ. Väčšinou sú tieto zmeny oznámené vopred, no nie je to pravidlo a možnosť znova rozparsovať časť alebo všetky v minulosti uložené dáta môže pôsobiť ako záchrana. [38]



Obrázok 5.3 – databázova schéma frameworku 140dev [38]

6 RapidMiner

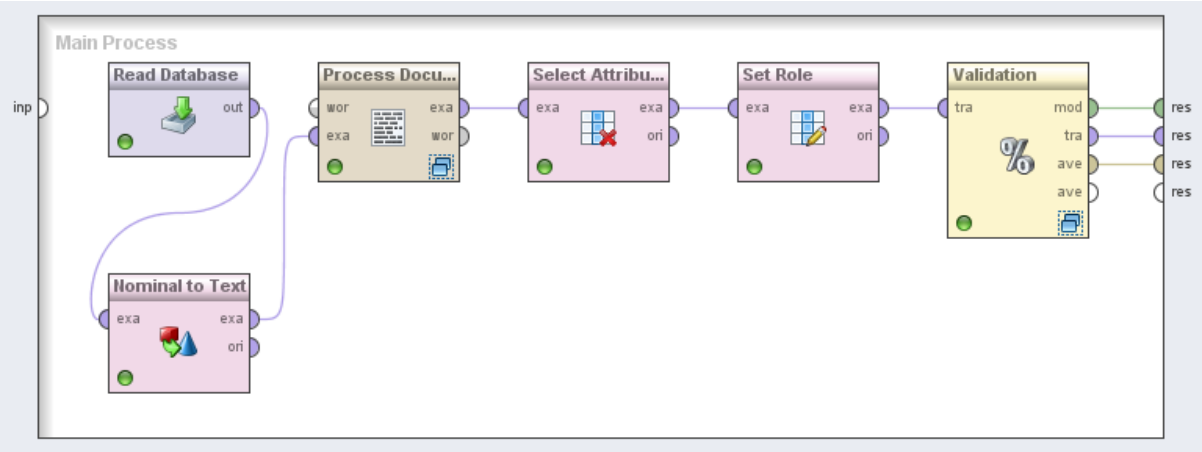
Pred samotnou implementáciou webovej aplikácie na dolovanie textových dát zo sociálnej siete Twitter som najprv chcel overiť, či implementácia takejto aplikácie môže priniesť nejaké relevantné výsledky a teda či má implementácia vôbec zmysel. Na overenie toho, na koľko presne je možné z krátkych textov tweetov odhadovať iné informácie, som sa rozhodol využiť software RapidMiner.

Ide o prostredie, ktoré poskytuje prostriedky pre strojové učenie, dolovanie z dát, získavanie znalostí z textových dát (po nahratí špeciálneho prídavného modulu), prediktívnu analýzu a biznis analýzu. Toto prostredie sa bežne používa pre výskum, edukačné účely, rapídne prototypovanie, vývoj aplikácií a priemyselné aplikácie. [39] Jedná sa o veľmi obľúbené a solídne prostredie, kde som mohol namodelovať niekoľko skúmaných scenárov dolovania textových znalostí z tweetov a získať tak pomerne jednoducho niekoľko rôznych výsledkov, ktoré som mohol následne porovnať a zhodnotiť. Veľkou výhodou tohto prístupu bolo, že som mohol model čo najsprávnejšie „nakalibrovať“ vyskúšaním viacerých techník a ich kombinácií a získať tak najlepšie možné percento úspešnosti klasifikácie tweetov do kategórií podľa kontinentu ich pôvodu.

Na základe charakteru skúmaných dát sa dá určiť najvhodnejší klasifikačný algoritmus, ktorý poskytne najlepšie riešenie a výsledky pre daný problém. Algoritmy sa môžu od seba líšiť vzhľadom na celkovú presnosť, prípadne času na vykonanie úlohy. V praxi je často vhodné navrhnúť a vytvoriť niekoľko modelov pre každý algoritmus, vytvoriť najlepší model pre každý algoritmus a následne vybrať najlepšie riešenie pre samotné nasadenie riešenia do praxe a jeho implementáciu.

6.1 Schéma dolovania

Pred implementáciou webovej aplikácie som najprv vytvoril model v programe RapidMiner 5, ktorý ponúka všetky potrebné nástroje a techniky na daný problém. Nakoľko bolo potrebné dolovanie z textových dát, bolo treba doinštalovať modul pre túto úlohu, ktorý dodal funkcionalitu pre spracovanie textových dát ako tokenizácia, stemming apod. V tejto časti opíšem jednotlivé použité funkčné bloky a ich nastavenia. Kompletná schéma, na ktorej boli prevedené prvotné experimenty pred samotnou implementáciou, je na obrázku 6.1.



Obrázok 6.1 – schéma dolovania v programe RapidMiner

Ako názov napovedá, blok **Read Database** sa stará o to získať vzorku dát z databázy pomocou jednoduchého MySQL SELECT príkazu. Vyberá texty všetkých tweetov, ktoré majú vyplnený aj atribút pre kontinent. Ten sa použije pri krížovej validácii (viď. Kapitola 6.1.2).

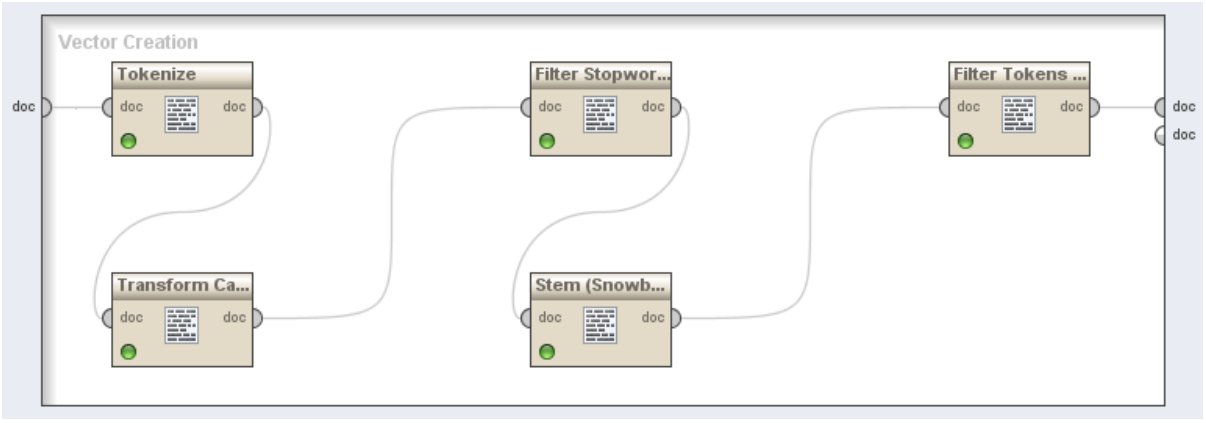
Aby bol RapidMiner schopný pracovať s textami tweetov, bolo nutné použiť operátor **Nominal to Text**, ktorý text tweetu prekonvertuje na reťazec. Takto skonvertované tweety slúžia následne ako vstupná množina pre ďalší operátor.

Process Documents from Data je operátor, ktorý vykonáva spracovanie textov na vstupe podľa presne nakonfigurovaného nastavenia, ktoré sa skladá z ďalších samotných funkčných blokov vo vnútri tohto operátora. Schéma spracovania textov je znázornená na obrázku 6.2 a bližšie popísaná v časti 6.1.1 Schéma pre spracovanie textov.

Pomocou operátorov **Select Attribute** a **Set Role** sa programu RapidMiner napovie, aký konkrétny atribút (v tomto prípade vybraný stĺpec z databázy) bude zastupovať akú rolu. Nastavenie je teda také, že atribút *kontinent* bude zastupovať triedy, do ktorých bude možné tweety klasifikovať. Tieto informácie sa odovzdajú ďalej do posledného a najpodstatnejšieho funkčného bloku **Validation (X-Validation)** teda krížovej validácie, ktorej princíp a nastavenie je popísané v časti 6.1.2. Výsledky krížovej validácie sú posledným výstupom a po spustení a dokončení celého procesu sa zobrazia užívateľovi.

6.1.1 Schéma pre spracovanie textov

Vnútro funkčného bloku pre spracovanie jednotlivých textov tweetov sa skladá z piatich operátorov, ktoré sú zapojené sériovo, kde vstup jedného operátoru je výstupom operátoru, ktorý ho predchádza.



Obrázok 6.2 – schéma funkčného bloku pre spracovanie textov v programe RapidMiner

6.1.1.1 Tokenize

Tokenize je prvý blok v spracovaní textov. Význam tokenizácie je priblížený v časti 4.1. Tokenizáciu je možno nastaviť do jedného z viacerých módov, na základe ktorého sa body na rozdelenie textu vyberajú podľa iného princípu. Mód bloku je nastavený tak, že text tweetu sa rozdelí na tokeny na nepísmenových (angl. non-letters) znakoch.

6.1.1.2 Transform Cases

Tento operátor je nastavený jednoducho tak, že transformuje všetky veľké písmená na malé, aby sa slová, ktoré sú rozdielne len vo veľkosti písmen, spracovávali jednotne.

6.1.1.3 Filter Stopwords

Ide o operátor, ktorý z tweetov odstráni anglické neplnovýznamové slová (viď časť 4.3). Používa na to slovník zabudovaný v programe RapidMiner.

6.1.1.4 Stem (Snowball)

Toto je operátor pre izoláciu koreňa slov (viď časť 4.2). *Snowball* je framework „stemming“ algoritmov. Rôzne jazyky potrebujú rôzne stemming algoritmy a je teda možné zvoliť jazyk. Na tieto účely som zvolil *Porter stemming* algoritmus, ktorý je veľmi často používaný a stal sa z neho de facto štandard pre stemming anglických slov.

6.1.1.5 Filter Tokens (by Length)

Posledný operátor v sérii operátorov v rámci spracovania textov tweetov slúži na to, že odstráni z textu slová, ktoré nezapadajú svojou dĺžkou do vymedzeného rozsahu. Pomocou tohto operátoru som nechal odstrániť slová, ktoré sú kratšie ako dva znaky, nakoľko by neniesli žiadnu reálnu informačnú hodnotu a model by musel s nimi zbytočne počítať.

Po týchto spomínaných krokoch sa z výsledných spracovaných tweetov vypočíta vektor slov, ktorý bude numericky reprezentovať daný tweet. Je možné vybrať schému pre vytvorenie tohto vektora. Na tieto účely som použil schému TF-IDF (viď. časť 4.5.2).

6.1.2 Krížová validácia

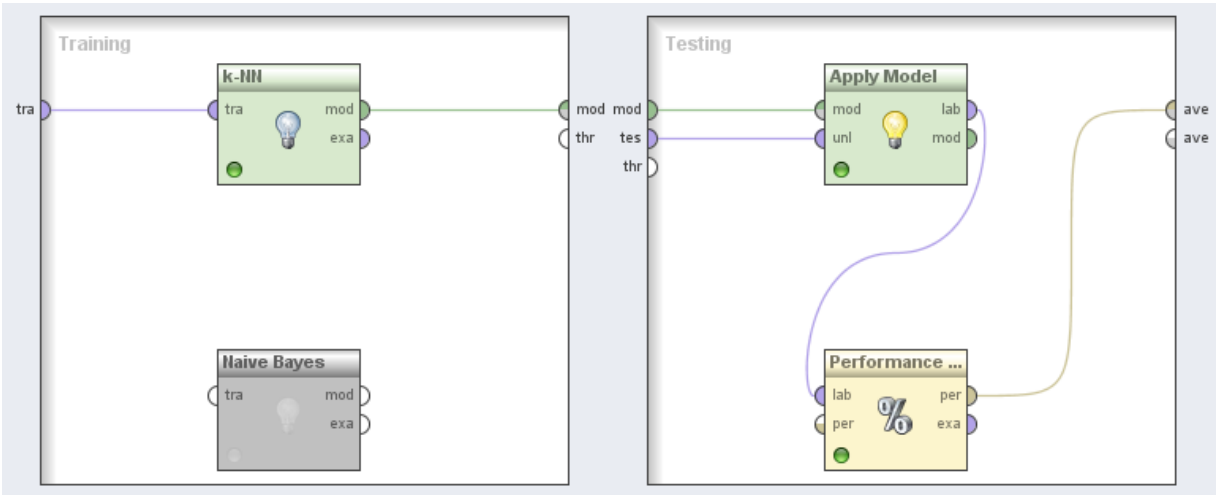
Posledným krokom v schéme modelu je tzv. **krížová validácia** (angl. cross-validation). Krížová validácia je technika pre validáciu modelu štatistickej analýzy. Je to metóda, ktorou možno zisťovať, ako bude tento model ovplyvňovať nezávislé vzorky dát. Tento postup je významný pre predikciu neznámych vzoriek po predchádzajúcej klasifikácii známych vzoriek. Ide v nej teda o to odhadnúť, na koľko presne bude predikčný model fungovať v praxi.

Princíp krížovej validácie spočíva v tom, že vstupná vzorka dát je rozdelená na komplementárne podmnožiny. Jedna podmnožina slúži ako trénovacia, na ktorej je vykonaná analýza. Druhá podmnožina sa nazýva testovacia, na ktorej je overená analýza z trénovacej množiny. Aby sa aspoň do istej miery predišlo kolísavosti nameraných výsledkov, vykonáva sa viacero kôl krížovej validácie, pričom sa vždy trénovacia a testovacia množina vyberie z inej časti vstupnej vzorky dát. Výsledky validácie sa zo všetkých vykonaných kôl spriemerujú. [40]

Existuje viacero typov krížovej validácie. Typ krížovej validácie, ktorý používa RapidMiner a ktorá je implementovaná v knižnici LibSVM, ktorej použitie bude opísané neskôr, sa nazýva **k-fold krížová validácia.** V tomto type validácie je vstupná vzorka dát rozdelená na *k* rovnako mohutných podmnožín. Pre každú podmnožinu je potom prevedená nasledujúca procedúra: vytvorí sa model za využitia zvyšných *(k - 1)* podmnožín ako trénovacia množina. Tento model je overený na aktuálnej množine. To znamená, že každá podmnožina je použitá na testovanie práve raz. Po *k* kolách sa výsledky spriemerujú.

V experimentoch v programe RapidMiner som skúšal rôzne hodnoty parametru *k*. Ďalej v práci budú predostreté výsledky experimentov krížovej validácie prevádzaných s rôznymi hodnotami parametru *k* aj pomocou knižnice LibSVM použitej priamo v implementácii.

Schéma krížovej validácie je zobrazená na obrázku 6.3. Z obrázku je vidno, ako je proces validácie rozdelený na trénovaciu a testovaciu časť. V trénovacej časti je vybraný algoritmus pre vyhodnotenie vektorového modelu. Experimentoval som s algoritmami *k-najbližší susedia* a *Naivný Bayesov algoritmus* (na obrázku zašednutý tj. deaktivovaný). V testovacej časti sa algoritmus aplikuje a jeho výkon zvaliduje. Pomocou bloku **Performance** sa získajú výsledky ako celková presnosť.



Obrázok 6.3 – Schéma krížovej validácie v programe RapidMiner

6.1.3 Confusion Matrix

Výsledky toho, nakoľko úspešne alebo neúspešne klasifikačný algoritmus predikoval zaradenie jednotlivých textových dokumentov (v prípade tejto práce ide o klasifikovanie jednotlivých tweetov) do konkrétnych tried, sa zobrazuje pomocou tzv. „**confusion matrix**“ tabuľky. Táto tabuľka zobrazuje počet správnych a nesprávnych predikcií natrénovaného modelu a porovnáva ho s aktuálnymi správnymi hodnotami v rámci testovacej množiny dát. Tabuľka obsahuje N × N buniek s výsledkami, kde N vyjadruje počet tried. Výhodou confusion matrix je to, že sa z nej dá vyčítať povaha nesprávne predikovaných klasifikácií ako aj ich konkrétny počet.

V tejto tabuľke figurujú nasledujúce 4 základné čísla:

- **TP** (angl. skratka pre true positive): počet prípadov, kedy bol úsudok algoritmu v rámci triedy pozitívny (tj. jedná sa o dokument z danej kategórie) a správny
- **TN** (angl. true negative): počet prípadov, kedy bol úsudok algoritmu taký, že sa nejedná o práve danú triedu a tento úsudok bol správny
- **FP** (angl. false positive): počet prípadov, kedy bol úsudok algoritmu pozitívny, ale nesprávny
- **FN** (angl. false negative): počet prípadov, kedy bol úsudok algoritmu negatívny a nesprávny

Confusion matrix tiež obsahuje ďalšie dve štatistické hodnoty pre každú triedu. Ide o **presnosť** (angl. precision) a **návratnosť** (angl. recall). Precision, teda presnosť, vyjadruje v percentách pomer počtu správnych úsudkov k počtu nesprávnych úsudkov. Vypočíta sa pomocou vzorca:

$$precision = TP \,/\, (TP + FP)$$

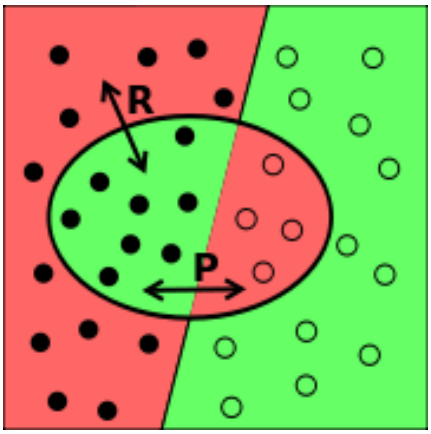
Recall percentuálne vyjadruje počet pozitívnych prípadov, ktoré boli správne ohodnotené. Vzorec na výpočet hodnoty recall:

$$recall = TP \;/\; (TP + FN)$$

Posledným číslom je **správnosť** (angl. accuracy), ktorá je percentuálnym vyjadrením predikcií, ktoré boli správne. Vypočíta sa vzorcom:

$$accuracy = (TP \; + \; TN) \;/\; (TP \; + \; TN + FP \; + \; FN)$$

Význam hodnôt presnosť a návratnosť je znázornený na obrázku 6.4, kde relevantné položky (položky danej triedy) sú naľavo od priamky a získané položky sú vnútri oválu. Červené regióny reprezentujú chyby: naľavo od priamky sú to nezískané relevantné položky (FN), zatiaľ čo napravo od priamky sú to získané irelevantné položky (FP). Vysoká hodnota návratnosti teda znamená, že algoritmus vrátil väčšinu zo všetkých možných relevantných výsledkov, kým vysoká presnosť znamená, že algoritmus vrátil podstatne viac relevantných výsledkov než irelevantných.



Obrázok 6.4 – znázornenie významu hodnôt presnosti a návratnosti

7 Návrh webovej aplikácie

Na základe výsledkov z preskúmania troch rôznych API sociálnych sietí a charakteru skúmaných a získaných dát, som mohol navrhnúť štruktúru webovej aplikácie. Najadekvátnejšia sociálna sieť na túto úlohu mi prišla sieť Twitter s jej miliónmi tweetov objavujúcich sa denne a kvôli možnosti ich všetky použiť v dolovaní bez toho, aby aplikácia musela obdržať povolenie od každého užívateľa, ako je tomu napr. pri sociálnej sieti Facebook.

Po preskúmaní štruktúry tweetov a dát, aké so sebou nesú, bolo treba rozhodnúť o tom, o aké informácie sa bude aplikácia zaujímať, tj. čo sa bude presne z týchto dát dolovať a aká sa na to vyberie vhodná technika. Tweety majú možnosť byť označkované geografickou lokáciou miesta, kde boli vytvorené pomocou zemepisnej šírky a dĺžky. Táto informácia by sa dala využiť v korelácii s iným údajom v rámci tweetu, prípadne užívateľa, ktorý daný tweet vytvoril. Jedna z možností bola skúmať korelácie medzi textom tweetu a práve spomínanou geografickou lokáciou a overenie, či nie je možné odhadnúť na základe tak krátkeho textu ako je 140 znakový tweet lokáciu jeho vzniku. Náplň ďalších častí práce je overenie toho, nakoľko je možne s využitím vhodných prostriedkov dolovania textových dát odvodiť z textu tweetu kontinent, na ktorom tweet vznikol.

Z takto zadaného problému a charakteru dát sa ponúka implementáciu rozdeliť na tri hlavné časti podľa funkcionality, akú vykonávajú. Prvý modul bude slúžiť na získavanie geografických informácií o každom tweete na základe zemepisnej šírky a dĺžky. Druhý modul sa bude starať o manipuláciu s databázou a s tweetami v nej. Tretí modul bude mať za úlohu vykonávať samotné dolovanie z textových dát, počítať na to potrebné hodnoty a klasifikovať tweety do jednotlivých kategórií.

Samotná webová aplikácia bude mať jednoduché užívateľské rozhranie, kde budú nastaviteľné parametre pre dolovanie. Úložisko dát na účely dolovania bude databáza MySQL, ktorá bude pre tento projekt dostačujúca. Po preskúmaní možností pre získavanie tweetov a porovnaní vhodných algoritmov na dolovanie bolo na mieste rozhodnutie, že aplikácia bude napísaná v jazyku PHP za použitia rozhrania Phirehose, ktoré je na účely ukladania tweetov často používané. Ide o PHP rozhranie k Twitter Streaming API. Aby sa však dali efektívne odchytávať a ukladať tweety, bude treba naprogramovať a spustiť na serveri na pozadí proces, alebo systémového deamona, ktorý po spustení vytvorí perzistentné HTTP spojenie, ktoré sa bude následne udržiavať podľa potreby pre rôzne časové obdobia, kým sa nenazbiera dostatočná vzorka tweetov. Tento proces ukladania a parsovania tweetov bol bližšie popísaný v kapitole 5. Algoritmus pre samotnú klasifikáciu bude SVM, ktorý je bližšie preskúmaný v kapitole 4. Pre tieto potreby existuje knižnica, ktorá tento algoritmus implementuje a je jednoducho použiteľná v PHP.

8 Implementácia

Webovú aplikáciu som sa rozhodol napísať v jazyku PHP. Nakoľko je modul pre získavanie tweetov napísaný tiež v PHP za pomoci na to vhodných knižníc, naskytá sa tu vďaka tomu možnosť tento modul s aplikáciou v budúcnosti prepojiť a eventuálne nastaviť získavanie a ukladanie tweetov podľa aktuálnej potreby. Navyše pre tento prístup existuje možnosť využiť priamo v kóde PHP knižnicu **LIBSVM** [41] pomocou na to predpripraveného rozšírenia **PHP SVM**, ktoré túto knižnicu obaľuje pomocou PHP rozhrania pre jej ľahké použitie v skriptoch. [42]

Pre základy, na ktoré som aplikáciu položil, som zvolil obľúbený nástroj **Nette Framework.** Je vhodný ako na menšie, tak celkom komplexné webové riešenia. [43] Aplikácia by tak mohla byť v budúcnosti ľahko rozšíriteľná o prípadnú novú funkcionalitu.

Na manipuláciu s MySQL databázou som si vybral ľahko použiteľnú databázovú vrstvu **dibi.** [44]

8.1 Geografická lokácia

Pred tým, než by bolo možné uložené tweety použiť ako trénovacie dáta pre získanie modelu na klasifikáciu, je nutné týmto tweetom odvodiť kontinent na základe ich geografických súradníc. O túto funkcionalitu sa v implementácii stará trieda GeoService.

Získať kontinent na základe zemepisnej šírky a dĺžky je možné viacerými spôsobmi. Existuje niekoľko zadarmo poskytovaných webových služieb, ktoré pracujú s geografickými dátami tohto typu a je možné od nich získavať rôzne geografické informácie ako krajina, mesto, dokonca ulica nachádzajúca sa na daných koordinátoch. Možností sa naskytuje niekoľko:

- Google Maps reverse geocoding API
- Yahoo Geoplanet API
- SimpleGeo
- Geonames.org
- Bing reverse geocoding API

Všetky vymenované služby sú jednoduché HTTP koncové body pre získavanie adresy na základe kombinácie údajov zemepisnej šírky a dĺžky. V implementácii som využil konkrétne webovú službu od Geonames.org. V kóde sa jednoducho zavolá konkrétna adresa s koordinátmi a naspať do aplikácie je navrátený dvojmiestny kód krajiny, ktorá sa na daných koordinátoch nachádza. Kód krajiny sa mapuje na prislúchajúci kontinent už priamo v zdrojovom kóde aplikácie. Po takto získaných kontinentoch pre každý tweet je možné tweety nechať spracovať aplikáciou na vytvorenie modelu pre klasifikáciu.

Nevýhoda služby geonames.org je tá, že má obmedzujúce limity na to, koľko takýchto volaní je dovolené pre danú konkrétnu IP adresu spraviť za deň. Pre účely tohto projektu táto skutočnosť nebola nijak obzvlášť obmedzujúca, nakoľko limit je stanovený na niekoľko tisíc denne a tak tento limit nebol v priebehu experimentovania takmer nikdy prekročený. Aj napriek tomuto obmedzeniu som sa rozhodol využiť túto službu, pretože v prípade, že by bola aplikácia používaná s mohutnejšími vzorkami dát, služba geonames.org ponúka možnosť, ako tomuto obmedzeniu predísť. Ak by to teda bolo potrebné, je možné si z tejto služby stiahnuť priamo databázu, ktorú táto služba používa a s priestorovým rozšírením MySQL vybudovať vlastnú službu, ktorá by vracala údaje bez akýchkoľvek obmedzení. Ak by nebolo nutné príliš veľké množstvo dotazov, druhá možnosť by bola využívať aspoň dve z predostretých služieb. Toto by mohlo byť realizovateľné tak, že po prekročení limitu jednej služby by toto aplikácia detekovala a automaticky by využila API niektorej ďalšej. Alternatívna služba by mohla byť napr. Google maps geocoding API, ktoré ma obmedzenie na 15000 dotazov z jednej IP adresy denne.

8.2 LIBSVM

LIBSVM je integrovaný software, ktorý ponúka okrem iného možnosti pre klasifikáciu numerických dát pomocou Support Vector Machines, čo je práve funkcionalita, ktorá je využitá v implementácii. Dôležitá schopnosť, ktorú knižnica musela mať, aby bola pre účely tohto projektu použiteľná, je podpora pre klasifikáciu dát do viacerých tried (angl. „multi-class classification“).

8.2.1 Použitie LIBSVM

Spôsob, akým sa knižnica používa, je taký, že sa najprv nadefinujú parametre, potom sa knižnici poskytnú trénovacie dáta, na základe ktorých sa vygeneruje model. Podľa vygenerovaného modelu sa môžu následne robiť predikcie na nových, „neznámych“ dátach. V nastavení LIBSVM ide o to vybrať:

- typ problému tj. či ide o klasifikáciu alebo regresiu
- typ tzv. kernel funckie
- tzv. „cost parameter“

Pre účely tejto práce sa z užívateľského rozhrania aplikácie dá vybrať jeden z dvoch spôsobov klasifikácie a to **C-SVC** alebo **nu-SVC**. Ide v podstate o tú istú techniku klasifikácie avšak s inými parametrami. C-SVC používa cost parameter ***C***, ktorý sa zadáva v rozmedzí od 0 do nekonečna a nu-SVC používa parameter ***nu***, ktorý sa zadáva vždy v rozmedzí od 0 do 1. Parameter *C* napovedá algoritmu SVM, akou veľkou mierou sa má snažiť vyhnúť sa nesprávnej klasifikácii každého jednotlivého vzorku. Vysoká hodnota parametru *C* znamená, že si klasifikátor vyberie radšej užšiu

oddeľujúcu hyperplochu, ak to pomôže klasifikovať viac vzoriek do správnej kategórie. Naopak veľmi nízka hodnota parametru *C* spôsobí, že klasifikátor bude hľadať širšie oddeľujúce hyperplochy aj napriek tomu, že takto zvolená hyperplocha spôsobí, že sa nesprávne klasifikuje viac vzoriek. Parameter *C* (a *nu*) teda ponúka flexibilitu v oddeľovaní jednotlivých kategórii. Kontroluje pomer medzi tým, aké veľké množstvo nesprávnych klasifikácií sa povolí, a mierou nakoľko sa bude snažiť vytvárať rigidne čo najširšie hyperplochy.

Ďalší povinný parameter, ktorý treba knižnici LIBSVM zadať, je typ **kernel funkcie**. Knižnica ponúka na výber 4 rôzne typy:

- lineárny
- polynomiálny
- radial basis function
- sigmoid

Po navolení parametrov sa formulár odošle a program vygeneruje model, ktorý sa môže v budúcnosti využívať na klasifikovanie nových tweetov, ktoré nenesú so sebou informáciu o kontinente (resp. geograficky označenom mieste) pôvodu.

8.2.2 Formát trénovacích a testovacích dát

Dáta môžu byť knižnici PHP SVM dodané buď ako súbor, dátový tok alebo pole. V prípade, že sú vzorky dodané ako súbor alebo dátový tok, musí obsahovať jeden riadok pre každú trénovaciu vzorku, ktorá musí byť naformátovaná v špecifickom tvare. Riadok pre danú vzorku musí obsahovať triedu reprezentovanú celým číslom, ktorá je nasledovaná postupnosťou párov *charakteristický_rys:hodnota*, ktorá je zoradená vzostupne podľa hodnôt charakteristických rysov. Charakteristické rysy sú celé čísla, hodnoty sú desatinné čísla, obvykle normalizované do rozmedzia od 0 do 1. Nasleduje príklad, ako by mohli vyzerať vstupné trénovacie dáta pre 3 vzorky. Každá vzorka reprezentuje jeden tweet z dvoch rôznych tried, kde trieda 1 značí napr. Severnú Ameriku a trieda 2 značí Európu. Každý charakteristický rys reprezentuje nejaké slovo a jemu prislúchajúca hodnota reprezentuje dôležitosť tohto slova v danom tweete. Rysy, ktoré majú hodnotu 0 (tj. slovo sa v tweete vôbec neobjavilo) jednoducho nie sú zahrnuté v týchto dátach. Rysy sú v prípade implementácie tohto projektu reprezentované

```
1 1:0.43 3:0.12 50:0.2
2 1:0.22 2:0.15 60:0.55 75:0.98
1 3:0.15 6:0.84 40:0.15 99:0.10
```

Na základe týmto spôsobom naformátovaných dát je možné vygenerovaný model uložiť a znova neskôr použiť na doposiaľ nevidené dáta. Model sa ukladá v textovom tvare ako súbor *model.svm* a obsahuje informácie o použitých parametroch a zoznam vygenerovaných support

vektorov. Hodnoty vstupných dát, ktoré som algoritmu predával na vstup, sa zhodovali na stotiny presne s hodnotami, ktoré vypočítaval model v programe RapidMiner. Algoritmus teda pracoval s vhodnými vstupnými dátami.

8.3 Štruktúra aplikácie

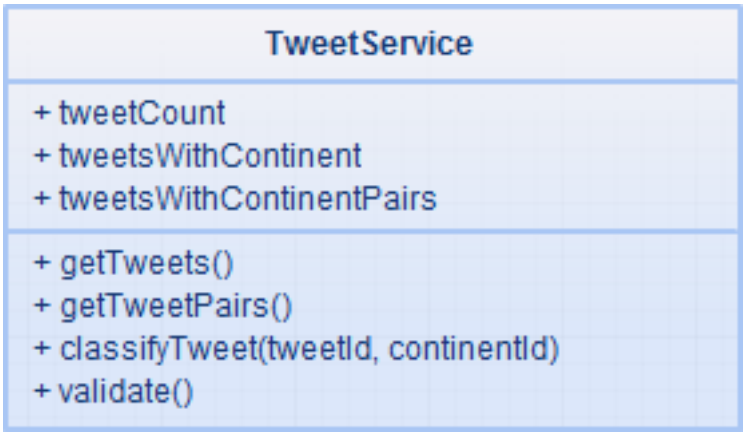
V tejto podkapitole budú priblížené tri hlavné servisné triedy, ktoré webová aplikácia využíva. Ide o Triedy GeoService, TweetService a ProcessDocumentService. Činnosti každej z nich budú ďalej bližšie popísané v jednotlivých podkapitolách.

8.3.1 Trieda GeoService

Obrázok 8.1 – diagram triedy GeoService

Táto trieda sa stará o získavanie potrebných geografických informácií k jednotlivým tweetom. Nakoľko sa pri získavaní tweetov procesom na pozadí ukladajú iba tweety, ktoré už nesú so sebou informácie o zemepisnej šírke a dĺžke, ďalším nevyhnutným krokom je získanie kontinentu pre každý uložený tweet. Na toto slúži metóda `updateTweetContinents`. Tá pre každý tweet, ktorý ešte nemá pridelený kontinent, volá metódu `getContinentFromCoordinates`, v ktorej sa najprv získa dvojmiestny kód krajiny podľa štandardu ISO 3166-1 alpha-2, k čomu sa využíva služba geonames.org, ktorá bola popísaná v časti 8.1. Po získaní kódu krajiny na základe zemepisných súradníc sa použije metóda `mapCountryCodeToContinent`, ktorá jednoduchým spôsobom mapuje daný kód krajiny na prislúchajúci kontinent, na ktorom sa krajina reprezentovaná poskytnutým kódom nachádza. Takto získaný kód krajiny a kontinent sa uloží do tabuľky tweets k danému záznamu tweetu do prislúchajúcich stĺpcov country_code a continent.

8.3.2 Trieda TweetService

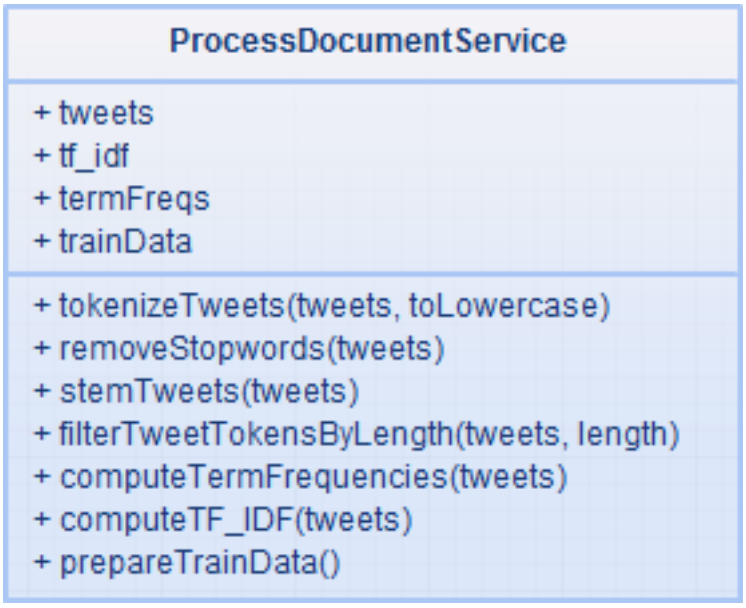


Obrázok 8.2 – diagram triedy TweetService

Servisná trieda TweetService slúži na to, aby vyberala z databázy relevantné množiny tweetov, s ktorými aplikácia má pracovať, a ktoré sa majú použiť na vypočítanie vektorového modelu textových dokumentov pomocou TF-IDF. Na získanie konkrétnych záznamov je vyhradená metóda `getTweetsWithContinent`, ktorá získava vzorku dát, ktorá sa využije na generovanie modelu algoritmom SVM. Tieto tweety sa zobrazujú v tabuľke v užívateľskom prostredí a je možné ich prehľadávať, prípadne zoraďovať podľa jedného zo zobrazených stĺpcov (id tweetu, užívateľova prezývka, text tweetu, dátum vytvorenia tweetu, skutočný kontinent pôvodu, predikovaný kontinent). Vybraná vzorka sa uloží do atribútu `tweetsWithContinent` a `tweetsWithContinentPairs`, ktoré vracajú k tomu určené metódy `getTweets` a `getTweetPairs`.

Trieda taktiež ukladá k záznamom predikované kontinenty metódou `classifyTweet`, na základe čoho môže následne vypočítať percentuálnu presnosť vykonanej predikcie metódou `validate`, v ktorej pri každom zázname porovnáva skutočný kontinent oproti predikovanému. Táto metóda vypočíta čísla pre tabuľku confusion matrix.

8.3.3 Trieda ProcessDocumentService



Obrázok 8.3 – diagram triedy ProcessDocumentService

Jedná sa o podstatnú časť implementácie, kde sú využité techniky dolovania z textov, ktoré sú opísané v kapitole 4. Triede sú predané tweety, ktoré boli vybraté triedou TweetService a postupne sa spracovávajú. Prvým krokom je tokenizácia, ktorú vykoná metóda `tokenizeTweets`. Pri tomto kroku sa znaky tokenov rovno transformujú na malé písmená. Takto tokenizované tweety sú ďalej predané metóde `removeStopwords`, ktorá odstráni tie tokeny, ktoré sa nachádzajú v zozname stop-slov. Ten je definovaný ako statické pole v samostatnej triede Stopwords, kde sa dajú prípadné stop-slová ľahko pridávať alebo mazať. Ďalší krok vykonáva metóda `stemTweets`, ktorá izoluje korene slovám spracovávaných tweetov pomocou Porterovho stemming algoritmu. Následne sa ešte vyhodia tokeny, ktoré sú kratšie ako 2 znaky pomocou metódy `filterTweetTokensByLength`. Takto upravené tweety sú pripravené na to, aby sa mohla vypočítať frekvencia jednotlivých slov a hodnôt TF-IDF pre každý token v každom tweete. Na toto slúžia metódy `computeTermFrequencies` a `computeTF_IDF`, ktoré hodnoty uložia do atribútov triedy `tf_idf` a `termFreqs`. Pomocou metódy `prepareTrainData` sa už len dáta pretransformujú do formátu, ktorý je vyžadovaný knižnicou PHP SVM (viď 8.2.2) metódou `prepareTrainData`. Týmto finálnym spôsobom pripravené dáta sa použijú na tvorbu modelu a klasifikáciu. Výsledky sú napokon zobrazené v tabuľke confusion matrix.

9 Experimentálna časť

V tejto kapitole sú predostreté výsledky experimentovania s nazbieranými dátami zo sociálnej siete Twitter. V podkapitole 9.1 sú ukázané výsledky experimentov, ktoré boli získané ešte pred samotnou implementáciou aplikácie. Boli získané prostredníctvom programu RapidMiner. V podkapitole 9.2 sú preskúmané výsledky nadobudnuté testovaním už implementovanej webovej aplikácie.

9.1 Experimenty v prostredí RapidMiner

Pred samotnou implementáciou som vykonal niekoľko testov v prostredí RapidMiner, aby som si overil, do akej percentuálnej miery sú existujúce algoritmy schopné klasifikovať aj také krátke texty, ako sú 140 znakové tweety.

9.1.1 Testovacie dáta

Prvá množina tweetov bola získaná v období, kedy bol zvolený pápež František, tj 13. marca 2013. V procese zbierajúcom dáta som navolil, aby tweety, ktoré sa mali ukladať, mali vo svojom texte obsiahnuté kľúčové slová *pope* alebo *vatican.* Ukázalo sa, že pri väčšej svetovej udalosti ľudia naozaj vo veľkej miere využívajú sociálnu sieť Twitter a tweety sa počas niekoľkých hodín ukladali tempom 40 až 60 tweetov za sekundu. Nazbieral som tak veľmi početnú množinu správ, z ktorých však len asi 1 až 2 percentá bolo označkovaných geografickými súradnicami. Vo výsledku to znamená, že som pracoval asi s 2289 tweetmi ohľadom témy voľby pápeža. Zastúpenie kontinentov v tejto množine dát je rozpísane v tabuľke 9.1.

Kontinent	Počet tweetov
North America	1401
Europe	362
Asia	428
South America	34
Africa	10
Oceania	54

Tabuľka 9.1 – rozloženie kontinentov v tweetoch pre experimenty ohľadom témy voľby pápeža

Ďalšie experimenty boli prevedené na odlišnej vzorke dát. Okrem voľby pápeža bola v tomto období veľmi rušným spôsobom aktívna Severná Kórea, ktorá prichádzala s jednou výstražnou správou za druhou. V tomto období som nazbieral teda ďalšiu vzorku dát, ktoré boli získavané pod

kritériom, že uložené tweety musia obsahovať kľúčové slová ako „north“, „korea“, „kim“. Opäť šlo o veľmi intenzívne obdobie, čo sa týka počtu tweetov generovaných užívateľmi po celom svete. Tentokrát som nazbieral vzorku 10550 tweetov, pre ktoré som mohol odvodiť kontinent ich pôvodu na základe ich geografických súradníc. Opäť platilo, že spomínaný počet tweetov tvoril asi o 1 až 2% zo všetkých uložených tweetov, ktorých boli stotisíce. Ide o dosť odlišnú tému a tým pádom sa dalo očakávať na jednej strane iné zastúpenie užívateľov s iným kontinentom pôvodu, ale na druhej strane aj obsahy samotných textových správ, ktoré títo užívatelia vygenerovali. Ukázalo sa však, že opäť tweetovali najmä obyvatelia Severnej Ameriky. Percentuálne zastúpenie jednotlivých kontinentov ukazuje tabuľka 9.2. Ďalšie experimenty teda boli prevedené na tejto vzorke dát.

Kontinent	Počet tweetov
North America	8124
Europe	1304
Asia	631
South America	351
Africa	75
Oceania	39

Tabuľka 9.2 – rozloženie kontinentov v tweetoch pre experimenty ohľadom témy Severná Kórea

9.1.2 Výsledky testov

Otestoval som najprv jeden z najbežnejších algoritmov a to Naivný bayesovský klasifikátor. Výsledky sú zobrazené v tabuľke 9.3.

Accuracy: 49,06%

	true Africa	**true Europe**	**true North America**	**true Asia**	**true South America**	**true Oceania**	**class precision**
pred. Africa	**1**	9	53	8	0	3	1.35%
pred. Europe	2	**132**	256	31	5	11	30.21%
pred. North America	4	133	**772**	103	16	27	73.18%
pred. Asia	2	38	158	**204**	5	3	49.76%
pred. South America	0	27	96	35	**7**	3	4.17%
pred. Oceania	1	23	66	47	1	**7**	4.83%
class recall	10.00%	36.46%	55.10%	47.66%	20.59%	12.96%	

Tabuľka 9.3 – experiment č. 1.1 algoritmus Naive Bayes

Ukázalo sa, že Bayesovský klasifikátor so svojou celkovou presnosťou 49% nie je veľmi vhodný na textové dáta tohto typu. Pre porovnanie som rovnakú vzorku dát použil na druhý experiment, kedy vzorky klasifikoval algoritmus k-najbližších susedov. Výsledky sú v tabuľke 9.4.

Accuracy: 66,32%

	true Africa	**true Europe**	**true North America**	**true Asia**	**true South America**	**true Oceania**	**class precision**
pred. Africa	**0**	1	0	0	0	0	0.00%
pred. Europe	1	**61**	78	10	2	4	39.10%
pred. North America	8	259	**1175**	136	27	42	71.34%
pred. Asia	1	39	142	**281**	5	7	59.16%
pred. South America	0	1	3	0	**0**	0	0.00%

pred. Oceania	0	1	3	1	0	**1**	16.67%
class recall	0.00%	16.85%	83.87%	65.65%	0.00%	1.85%	

Tabuľka 9.4 – experiment č. 1.2 pre tweety obsahujúce kľúčové slovo *pope,* algoritmus k-NN

Celková presnosť algoritmu k-najbližších susedov je vyššia, no stále nie sú výsledky dostatočne presvedčivé. Ďalšími experimentmi som chcel zistiť, ako veľmi celkovej presnosti pomôže mohutnejšia množina testovacích vzoriek. Nasledujúce dva experimenty sú teda prevedené na rovnakých dvoch algoritmoch, no s inou vzorkou dát, ktorá sa týkala Severnej Kórey.

Výsledky algoritmu Naivný Bayes sa zlepšili asi o 6% s celkovou presnosťou 55,04%. Sú zobrazené v tabuľke 9.5. Ako z tabuľky vidno, algoritmus relatívne presne klasifikoval iba vzorky zo Severnej a Južnej Ameriky.

Algoritmus k-najbližší susedia zaznamenal o dosť väčšie zlepšenie presnosti s celkovou presnosťou 79,52%. Sú zobrazené v tabuľke 9.6. Nakoľko má však tento algoritmus veľmi veľké pamäťové nároky, bolo možné otestovať ho najväčšmi na vzorke 6000 tweetov. Algoritmus by bol kvôli spomínanej pamäťovej náročnosti v reálnom prostredí prakticky nepoužiteľný, pretože by bolo treba vytvoriť model z čo najväčšej množiny dát. Aj toto bol jeden z hlavných dôvodov, prečo vo finálnej implementácii bol použitý algoritmus SVM, ktorý by mal dosahovať porovnateľné, prípadne lepšie výsledky.

Accuracy: 55,04%

	true North America	**true Europe**	**true Asia**	**true Africa**	**true South America**	**true Oceania**	**class precision**
pred. North America	**4399**	381	133	35	31	14	88.10%
pred. Europe	2017	**703**	78	15	11	13	24.78%
pred. Asia	635	78	**374**	5	3	2	34.09%
pred. Africa	444	53	16	**14**	6	2	2.62%
pred. South America	82	19	5	2	**294**	0	73.13%
pred. Oceania	547	70	25	4	6	**8**	1.21%
class recall	54.15%	53.91%	59.27%	18.67%	83.76%	20.51%	

Tabuľka 9.5 – experiment č. 1.3 pre tweety obsahujúce kľúčové slovo *korea,* algoritmus Naive Bayes

Accuracy: 79,52%

	true North America	**true Europe**	**true Asia**	**true Africa**	**true South America**	**true Oceania**	**class precision**
pred. North America	**4293**	694	152	52	28	15	82.02%
pred. Europe	180	**286**	38	3	13	1	54.89%
pred. Asia	5	8	**86**	0	2	0	85.15%
pred. Africa	1	0	0	**4**	0	0	80.00%
pred. South America	22	15	0	0	**102**	0	73.38%
pred. Oceania	0	0	0	0	0	**0**	0.00%
class recall	95.38%	28.51%	31.16%	6.78%	70.34%	0.00%	

Tabuľka 9.6 – experiment č. 1.4 pre tweety obsahujúce kľúčové slovo *korea,* algoritmus k-NN

9.2 Experimenty vo webovej aplikácii

V tejto časti predostriem výsledky, ktoré boli namerané pri experimentovaní s implementovanou webovou aplikáciou využívajúc pri tom knižnicu LIBSVM. Boli použité viaceré vzorky dát, vždy nazbierané na základe iných kľúčových slov, ktoré mali ukladané tweety vo svojom texte obsahovať.

Prvé experimenty sú vykonané na rovnakej množine dát týkajúcej sa voľby pápeža. Tweety však celkom jednoznačne najčastejšie pochádzajú zo Severnej Ameriky. Naopak v kontinentoch Afrika, Južná Amerika a Oceánia sa tweetuje najmenej. V týchto spomínaných troch kontinentoch je dokonca časť všetkých tweetov tak malá, že bolo vhodné ich spojiť do jednej spoločnej triedy v záujme udržať čo možno najrovnomernejšie rozdistribuovanie dát v jednotlivých kategóriách. Vo výsledku potom aplikácia klasifikuje tweety celkovo do 4 tried. Podarilo sa mi implementovať k-fold validáciu, ktorá vracia takmer rovnaké výsledky ako knižnica LIBSVM. Používam spôsob, v ktorom sa najprv množina dát náhodne zamieša a následne sa z nej postupne vyberajú podmnožiny. Pri tabuľkách teda vždy uvádzam celkovú presnosť, ktorú vypočítala knižnica a celkovú presnosť, ktorú vypočítala moja implementácia.

Výsledok experimentu č. 2.1 zobrazený v tvare confusion matrix pre SVM typu **nu-SVC**, **lineárny kernel** a parameter ***nu*** **nastavený na hodnotu 0.1** a **k-folds validácia na 10** zobrazuje

tabuľka 9.7. Celková presnosť vypočítaná knižnicou LIBSVM po 10-folds krížovej validácii je **65,968%.**

Accuracy: 66,273%

LIBSVM accuracy: 65,968%

	true North America	**true Europe**	**true Asia**	**true South America / Africa / Oceania**	**class precision**
pred. North America	**1166**	151	79	5	83.23%
pred. Europe	246	**88**	23	5	24.31%
pred. Asia	160	18	**248**	2	57.94%
pred. South America / Africa / Oceania	71	16	11	**0**	0%
class recall	70.97%	32.23%	68.7%	0%	

Tabuľka 9.7 – experiment č. 2.1 pre tweety obsahujúce kľúčové slovo *pope*

V **experimente č.2** bol zmenený typ SVM na **C-SVM**. **Parameter C bol nastavený na 1**. Ostatné nastavenia ostali rovnaké ako v prvom experimente. Celková presnosť sa podľa krížovej validácie LIBSVM zvýšila asi o 5%. Výsledky zobrazuje tabuľka č. 9.8. Je možné si všimnúť, že model predpokladá, že väčšina tweetov pochádza zo Severnej Ameriky a tieto si mýli s tweetami z Európy. Tweety z Ázie triafa presnejšie. Problém mu robí okrem Európy aj správne klasifikovať tweety z minoritných kontinentov tj. zo 4. triedy. Tu by pomohlo, aby trénovacia množina obsahovala množstvo vzoriek z tejto triedy porovnateľne veľké s ostatnými triedami. Získanie dostatočného množstva označených vzoriek z tejto triedy by však mohlo zabrať značne dlhé obdobie neustáleho ukladania tweetov. Tento problém je do istej miery vyriešený v druhom type experimentov, používajúc inú vstupnú množinu, v ktorej je vzoriek väčšie množstvo.

Accuracy: 69,201%

LIBSVM Accuracy: 70,074%

	true North America	**true Europe**	**true Asia**	**true South America / Africa / Oceania**	**class precision**
pred. North America	**1334**	19	48	0	95.22%
pred. Europe	315	**31**	16	0	8.56%
pred. Asia	208	1	**219**	0	51.17%
pred. South America / Africa / Oceania	87	3	8	**0**	0%
class recall	68.62%	57.41%	75.26%	0%	

Tabuľka 9.8 – experiment č. 2.2 pre tweety obsahujúce kľúčové slovo *pope* s alternatívnymi nastaveniami

Experiment č. 2.3 bol uskutočnený s rovnakými parametrami ako experiment č. 2.1. Rozdiel bol len vo vstupnej vzorke dát, ktorej správy obsahovali kľúčové slová ohľadom Severnej Kórey, no najmä ich bolo značne viac. Výsledky zobrazuje tabuľka 9.9. Výsledky po 10-fold krížovej validácii sú v tomto prípade výrazne presnejšie, čo ukazuje, že na veľkosti množiny trénovacích dát záleží v dosť značnej miere. Celková presnosť 84,336% je už o dosť prijateľnejšia.

Accuracy: 84,336%

LIBSVM Accuracy: 84,45%

	true North America	**true Europe**	**true Asia**	**true South America / Africa / Oceania**	**class precision**
pred. North America	**7822**	225	16	58	96.32%
pred. Europe	775	**518**	10	1	39.72%
pred. Asia	336	27	**268**	0	42.47%
pred. South America / Africa / Oceania	180	18	2	**265**	56.99%
class recall	85.83%	65.74%	90.54%	81.79%	

Tabuľka 9.9 – experiment č. 2.3 pre tweety obsahujúce kľúčové slovo *korea*

Ďalším experimentom som chcel overiť, či algoritmus nebude pracovať ešte presnejšie, keď na vstup ako trénovacie dáta dostane rovnomernejšie rozložené zastúpenia jednotlivých tried. Pre experiment č. 2.4 som odstránil teda niekoľko tisíc záznamov, ktoré mali pôvod v Severnej Amerike, aby boli počty tweetov v jednotlivých triedach čo možno najpodobnejšie. Počty tweetov ostali

rovnaké, ako ukazuje tabuľka 9.2, zredukoval sa len počet tweetov zo Severnej Ameriky, aby zodpovedalo počtu tweetov z Európy. Výsledky tohto experimentu sú v tabuľke 9.10.

Accuracy: 70,301%
LIBSVM Accuracy: 70,086%

	true North America	**true Europe**	**true Asia**	**true South America / Africa / Oceania**	**class precision**
pred. North America	**1037**	238	27	22	78.32%
pred. Europe	377	**885**	39	3	67.87%
pred. Asia	132	99	**399**	1	63.23%
pred. South America / Africa / Oceania	71	80	17	**297**	63.87%
class recall	64.13%	67.97%	82.78%	91.95%	

Tabuľka 9.10 – experiment č. 2.4 pre rovnomerne rozdistribuované tweety obsahujúce kľúčové slovo *korea*

Celková presnosť sa pri rovnomernejšej distribúcii tweetov v jednotlivých triedach pri 10-fold krížovej validácii zhoršila naspäť na necelých 70%, čo je percento, ktoré sa vo výsledkoch objavuje pomerne často. V porovnaní s predchádzajúcim experimentom to však podstatne vylepšilo presnosť, s akou algoritmus odhaduje texty z jednotlivých tried. Ide teda o výmenu, kde sa o malé percento znížila presnosť predikcie triedy pre Severnú Ameriku, ale naopak sa výrazne zvýšila presnosť predikcie všetkých ostatných tried a vyšplhala sa pri každej nad 60%. Z týchto skutočností je teda možné usúdiť, že je vhodnejšie, keď sa trénovacia množina vytvorí z čo možno najrovnomernejšie rozdistribuovaných dát v jednotlivých kategóriách. Ideálne by už bolo iba získať túto množinu čo najväčšiu, čo by mohlo výsledky ešte o niekoľko percent vylepšiť.

Posledným typom experimentov bolo vyskúšať tieto dve množiny spojiť do jednej a vyskúšať algoritmus klasifikovať takto rôznorodejšiu množinu dát. Celkový počet tweetov bol pri týchto experimentoch 12815. Nastavenie pre prvý experiment tohto typu bolo rovnaké ako v experimente č. 2.2, tj. 10-fold krížová validácia na SVM type C-SVM, s lineárnym kernelom, s parametrom C nastavenom na 1. Ide o najlepšie možné nastavenia, pre vytvorenie modelu pre klasifikáciu tweetov, ktoré sa mi podarilo objaviť. Výsledky sú v tabuľke 9.11.

Accuracy: 85,306%

LIBSVM Accuracy: 85,287%

	true North America	**true Europe**	**true Asia**	**true South America / Africa / Oceania**	**class precision**
pred. North America	**8080**	22	8	11	99.5%
pred. Europe	924	**366**	10	4	28.07%
pred. Asia	378	9	**243**	1	38.51%
pred. South America / Africa / Oceania	172	6	1	**286**	61.51%
class recall	84.57%	90.82%	92.75%	94.7%	

Tabuľka 9.11 – experiment č. 2.5 pre pre tweety s rôznorodým obsahom

Rovnako ako v minulom prípade, tak aj teraz som ďalší experiment spravil na vzorke dát, z ktorej som odstránil tweety zo Severnej Ameriky kvôli rovnomernej distribúcii ich kontinentov. Výsledky sú v tabuľke 9.12 a ukazujú obdobnú zmenu vo výsledkoch ako pri predchádzajúcich experimentoch, kedy boli dáta do tried rovnomernejšie rozdelené.

Accuracy: 70,973%

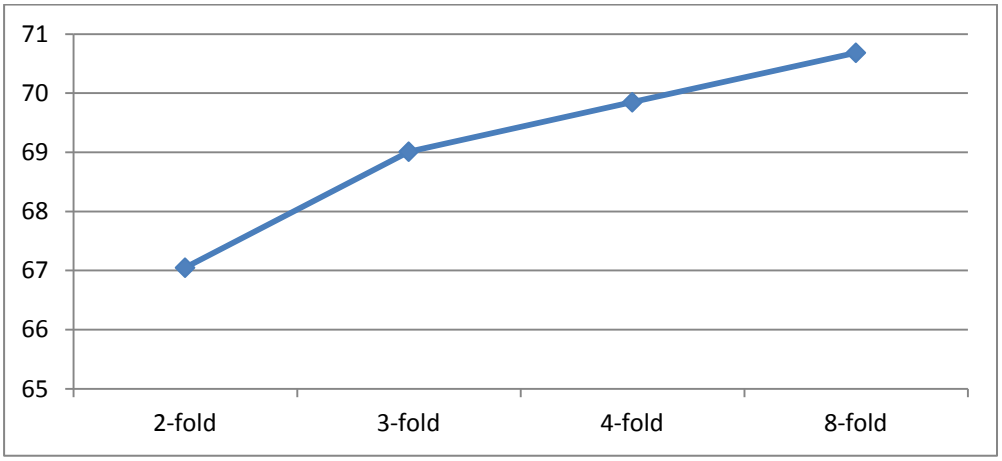
LIBSVM Accuracy: 70,424%

	true North America	**true Europe**	**true Asia**	**true South America / Africa / Oceania**	**class precision**
pred. North America	**1154**	217	32	21	81.04%
pred. Europe	401	**861**	37	5	66.03%
pred. Asia	151	78	**401**	1	63.55%
pred. South America / Africa / Oceania	75	75	17	**298**	64.09%
class recall	64.8%	69.94%	82.34%	91.69%	

Tabuľka 9.12 – experiment č. 2.6 pre rovnomerne rozdistribuované tweety s rôznorodým obsahom

S týmito nastaveniami som spravil sériu experimentov, kde som menil hodnotu *k* v k-fold validácii. Presnosť pri 2-fold validácii nie je taká vysoká ako pri vyšších hodnotách parametru *k*. Zistil som, že presnosť sa ustáli tesne nad číslom 70% pri hodnotách k 8 a viac. Znázorňuje to graf 9.1.

Takto predpripravená vzorka dát je podľa experimentov najvhodnejšia pre vytvorenie modelu, ktorý by mohol byť použitý v budúcnosti na predikciu pôvodu nových tweetov. Boli predostreté tie najlepšie výsledky, ktoré sa mi po viacerých experimentoch podarilo získať. Skúšal som pri tom rôzne kombinácie kernelov a ich parametrov, no dá sa povedať, že len spomínané nastavenia (najmä lineárneho kernelu) dávali relevantné výsledky. Ostatné kernely na tento typ dát nefungovali takmer vôbec, čo sa prejavovalo napr. takým spôsobom, že všetkým, alebo drvivej väčšine tweetov bol kontinent pôvodu predikovaný ako Severná Amerika.



Graf 9.1 – grafické znázornenie závislosti medzi parametrom k pri k-fold krížovej validácii a vypočítanou celkovou presnosťou modelu v percentách

10 Zhodnotenie dosiahnutých výsledkov

Ako sa dalo predpokladať, dolovanie znalostí z textových dokumentov tak krátkych ako sú tweety, je pomerne náročný problém, ktorý ani tie najlepšie algoritmy nie sú vždy schopné dostatočne presne riešiť. Navyše sa ukázalo, že napriek obrovskému počtu správ, ktoré sa na sociálnej sieti Twitter denne vygenerujú, sú asi len 2% všetkých tweetov geograficky označené. Toto množstvo bolo však ešte stále dostačujúcou vzorkou na to, aby boli aspoň tie lepšie algoritmy schopné relatívne presne predikovať kontinent, odkiaľ tweet pochádza.

Po porovnaní niektorých metód pred implementáciou v prostredí RapidMiner som usúdil, že zadaná úloha klasifikácie na základe textu tweetov by mohla byť riešiteľná s pomerne dobrou presnosťou, ktorá by mohla dosahovať aspoň 70%. K tomuto účelu som sa rozhodol využiť mocný algoritmus SVM, ktorý som preskúmal hlbšie a implementoval vo webovej aplikácii za využitia knižnice LIBSVM. Algoritmus SVM dosahuje obecne najlepšie výsledky v sfére dolovania dát, ale z mojich experimentov je pre skúmaný problém presnosťou porovnateľný s algoritmom k-najbližších susedov. Navyše však nemá tak veľké pamäťové nároky, kvôli ktorým by bol algoritmus k-NN len ťažko použiteľný. Celková presnosť algoritmu k-najbližších susedov pre klasifikáciu tweetov sa ukázala byť v rozmedzí od 70 do 85%.

Experimenty vykonané v implementovanom riešení vracali rôzne výsledky, ktoré sa pohybovali zhruba od 70 do 85%. Po spriemerovaní celkových presností jednotlivých experimentov vykonaných algoritmom SVM som sa dostal k hodnote **74,40%**. Ide teda o pomerne vysoké percento, nakoľko sa jednalo o tak krátke texty, ktoré so sebou častokrát niesli len minimum informačnej hodnoty a algoritmus mohol v niektorých prípadoch v podstate len „hádať“ správnu triedu. Táto percentuálna úspešnosť stúpala so zväčšujúcou sa množinou trénovacích dát a dá sa teda predpokladať, že model vytvorený z niekoľkonásobne väčšieho počtu získaných tweetov by dosahoval ešte o niečo lepšie výsledky. Okrem toho sa ukázalo, že je vhodné dáta rozdistribuovať takým spôsobom, aby boli jednotlivé triedy zastúpené čo možno najrovnomernejšie. Toto by predstavovalo ideálnu vzorku pred vytvorenie modelu.

Z výsledkov možno vyčítať, že sa algoritmus vo väčšine prípadov najlepšie naučil určovať tweety pochádzajúce zo Severnej Ameriky. Mal problém s tweetami z Európy, ktoré si mýlil s tweetam zo Severnej Ameriky, no tweety z Ázie predikoval naopak presnejšie. Táto skutočnosť sa dá odôvodniť tým, že tweety z Ázie sú písané rozdielnymi jazykmi. Napriek tomu, že tweetov zo zvyšku sveta bolo najmenej, predikoval ich algoritmus poväčšine presnejšie než tweety z Európy alebo Ázie.

Celkovo si myslím, že by sa výsledky dali ešte vylepšiť, keby sa tweety ukladali na jednej strane dostatočne dlho a na druhej strane by sa použili rôznorodejšie kľúčové slová s rôznych oblastí. Model vytvorený z takýchto dát by mal potenciál byť už celkom flexibilný, nech by sa jednalo o akúkoľvek skúmanú doménu so stabilnou presnosťou aspoň 80%. Získala by sa takto celkom

relevantná vzorka na vytvorenie modelu, ktorý by sa vytvoril po vyskúšaní ďalších, na daný model možno vhodnejších kombinácií vstupných parametrov algoritmu SVM. Je totiž možné, že parametre, ktoré dávali najlepšie výsledky pre množinu dát použitú v tejto práci, by nemuseli byť pre odlišnú a mohutnejšiu množinu tie najvhodnejšie. Bolo by teda rozumné tieto kombinácie pri nových dátach opäť overiť a preskúmať nové kombinácie. Na účely tejto práce sa však ako najúčinnejšie ukázali nasledovné parametre:

- typ SVM: C-SVM
- kernel: lineárny
- C: 1
- k pre k-fold krížovú validáciu: 8 až 10

11 Záver

Sociálne siete môžu mať vo svojich široko rozsiahlych dátach ukryté cenné informácie, ktoré by inak než pomocou špeciálnych techník získavania znalostí z dát boli len ťažko objaviteľné. Takto získané znalosti môžu byť využité pre politické kampane, vylepšenie služieb pre zákazníkov konkrétnych odvetví alebo lepšie zameranie reklamy.

Hlavným cieľom tohto projektu bolo preskúmať tri API troch rôznych sociálnych sietí. Išlo o webové služby Facebook, Twitter a Last.fm. Každá sieť ponúka mierne odlišné možnosti pre dolovanie z ich dát. Facebook kladie na súkromie najväčší dôraz a tak dáva každému užívateľovi možnosť, či svoje dáta povolí zdieľať s tretími stranami. Tým sa stáva menej praktickým pre rozsiahle dolovanie dát. Na druhej strane mikroblogovacia služba Twitter dáva možnosť používať pre vlastné účely dolovania všetky jeho statusy (tweety), ktoré možno zachytávať a ukladať v reálnom čase, čo pri počte viac než 340 miliónov tweetov denne predstavuje dostatočnú množinu dát pre efektívne využitie nejakej dolovacej metódy. Tretie preskúmané API sociálnej siete Last.fm ponúka celkom štandardné metódy pre získanie jej dát. Väčšina dát, ktoré by boli využiteľné pre dolovanie je verejne dostupných, takže by nebolo potrebné sa starať o nadbytočnú réžiu spojenú s tým, aby jednotliví užívatelia povoľovali, ktoré dáta poskytnú tretej strane.

Na základe množstva dostupných dát som zvolil sociálnu sieť Twitter a pomocou na to vhodných nástrojov získal vzorku dát, na ktorej bolo možné robiť experimenty s využitím niektorých dolovacích metód na získavanie znalostí z textových dát. Išlo o využitie niektorej konkrétnej dolovacej metódy pre klasifikáciu a predikciu. Úlohou bolo zistiť a overiť, či je možné odvodzovať skutočnosti na základe krátkeho textu tweetu. Šlo teda o predikciu toho, na akom kontinente tweet vznikol na základe textu, ktorý obsahuje.

Po niekoľkých experimentoch a skúmaní rôznych algoritmov na dolovanie textových dát som bližšie preskúmal algoritmus SVM, ktorý dosahuje obecne najlepšie výsledky. Ten som následne použil v implementácii webovej aplikácie.

Na účely spomínanej úlohy klasifikácie sa mi podarilo implementovať webovú aplikáciu, ktorá je schopná predikcie novovzniknutých, aplikáciou ešte nevidených tweetov do jednotlivých tried reprezentujúcich kontinenty. Využitie takejto aplikácie by mohlo byť napríklad v spravodajstve, kedy by sa pomocou nej dalo odhadovať, v ktorých častiach Zeme sa ľudia najviac zaujímajú o aktuálnu svetovú udalosť. Táto aplikácia by sa dala ďalej rozširovať o iné algoritmy, prípadne o nové úlohy dolovania dát, nakoľko má na to implementované potrebné postupy a predstavuje dobrý základ pre komplexnejšiu aplikáciu tohto typu. Jedným z možných rozšírení by mohlo byť napr. také, ktoré by k lokácii tweetu zisťovalo aj sentiment vzťahujúci sa vždy ku konkrétnej téme. Výsledkom by mohla byť webová, verejne dostupná aplikácia, ktorá by slúžila na postupné zmapovanie

jednotlivých častí sveta a postoja tamojších obyvateľov k aktuálnym svetovým, či už politickým, náboženským alebo iným témam a udalostiam.

Bližšie preskúmanie obsahu, ktoré užívatelia sociálnych sietí generujú, môže totiž odkrývať dôležité vzory pri politických udalostiach, revolúciách a pod. Netreba ich teda podceňovať. Akékoľvek ďalšie skúmanie v tejto oblasti pomocou prostriedkov, ktoré nám ponúka oblasť dolovania znalostí z dát, môže byť teda prospešné či už menším komunitám, konkrétnym krajinám alebo aj celému svetu.

Literatúra

[1] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. *From Data Mining to Knowledge Discovery in Databases, 1997*.

[2] Han, J., Kamber, M., Pei, J. *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2011. ISBN 978-0123814791.

[3] Zendulka J., Bartík V., Lukáš R., Rudolfová I. *Získavaní znalostí z databází*, VUT FIT, Brno, 2009.

[4] Gundecha, P., Liu, H. *Mining Social Media - A Brief Introduction*. Tutorials in Operation Research, INFORMS, 2005.

[5] Kaplan, A.M., Haenlein, M. *Users of the world, unite! The challenges and opportunities of Social Media.* Business Horizons, strany 53(1): 59–68, 2010.

[6] *99 New social media stats for 2012* [online]. Last modified on 10 May 2012. [cit. 2012-12-28]. Dostupné na URL: < http://thesocialskinny.com/99-new-social-media-stats-for-2012>.

[7] Yardi, S., Romero, D., Schoenebeck, G., Boyd, D. *Detecting Spam in a Twitter Network.* First Monday, strana 15(1), 2009.

[8] Chu, Z., Gianvecchio, S., Wang, H., Jajodia, S. *Who is Tweeting on Twitter: Human, Bot, or Cyborg?* Proceedings of the 26th Annual Computer Security Applications Conference, strany 21–30. ACM, 2010.

[9] *Arab Spring* [online]. Last modified on 26 December 2012. [cit. 2012-12-27]. Dostupné na URL: <http://en.wikipedia.org/wiki/Arab_spring>.

[10] *Occupy Wall Street* [online]. Last modified on 27 December 2012. [cit. 2012-12-27]. Dostupné na URL: <http://en.wikipedia.org/wiki/Occupy_Wall_Street>.

[11] *2011 Egyptian revolution* [online]. Last modified on 27 December 2012. [cit. 2012-12-27]. Dostupné na URL: <http://en.wikipedia.org/wiki/2011_Egyptian_revolution>.

[12] Gross, R. Acquisti, A. *Information Revelation and Privacy in Online Social Networks.* In the ACM workshop on Privacy in the electronic society, 2005.

[13] *API* [online]. Last modified on 28 December 2012. [cit. 2012-12-29]. Dostupné na URL: <http://cs.wikipedia.org/wiki/API>.

[14] Sengupta, S. *Facebook's Prospects May Rest on Trove of Data. In* The New York Times. 15 May 2012. Dostupné na URL: <http://www.nytimes.com/2012/05/15/technology/facebook-needs-to-turn-data-trove-into-investor-gold.html>.

[15] *Facebook Graph API* [online]. Last modified on 2 January 2013. [cit. 2013-01-02]. Dostupné na URL: <http://developers.facebook.com/docs/reference/api/>.

[16] *Social graph* [online]. Last modified on 11 December 2012. [cit. 2013-01-02]. Dostupné na URL: <http://en.wikipedia.org/wiki/Social_graph>.

[17] *JSON* [online]. Last modified on 1 January 2013. [cit. 2013-01-02]. Dostupné na URL: <http://www.json.org/>.

[18] Hardt, D. *The OAuth 2.0 Authorization Framework* [online]. Last modified on October 2012. [cit. 2013-01-02]. Dostupné na URL: <http://tools.ietf.org/html/rfc6749>.

[19] *Twitter* [online]. Last modified on 2 January 2013. [cit. 2013-01-03]. Dostupné na URL: <http://en.wikipedia.org/wiki/Twitter>.

[20] Perez, S. *This is what a tweet looks like* [online]. 19 April 2012. Dostupné na URL: <http://readwrite.com/2010/04/19/this_is_what_a_tweet_looks_like>.

[21] Filloux, F. *Datamining Twitter* [online]. The Guardian. 5 December 2011. Dostupné na URL: <http://www.guardian.co.uk/technology/2011/dec/05/monday-note-twitter>.

[22] Fielding, R.T. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000. Dostupné na URL: <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>.

[23] *Twitter documentation* [online]. Dostupné na URL: <https://dev.twitter.com/docs>.

[24] *Last.fm API* [online]. Dostupné na URL: <http://www.last.fm/api>.

[25] Feldman, R., Sanger, J. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006. ISBN 978-0521836579.

[26] Paralič, J., et al. *Dolovanie znalostí z textov*. Prvé vydanie. Technická univerzita v Košiciach, FEI, KKUI, Košice, 2010. ISBN 978-80-89284-62-7.

[27] Apté, C., Damerau, F., Weiss, S.M. 1994. *Towards language independent automated learning of text categorisation models.* In Research and Development in Information Retrieval, strany 23–30, 1994

[28] *Lemmatization* [online]. Last modified on 26 February 2013. [cit. 2013-05-09]. Dostupné na URL: < http://en.wikipedia.org/wiki/Lemmatization>.

[29] *Stemming* [online]. Last modified on 21 March 2013. [cit. 2013-05-09]. Dostupné na URL: <http://en.wikipedia.org/wiki/Stemming>.

[30] *The Porter Stemming Algorithm* [online]. Dostupné na URL: <http://tartarus.org/~martin/PorterStemmer/>.

[31] Furdík, K. 2003. *Získavanie informácií v prirodzenom jazyku s použitím hyper-textových štruktúr.* Doktorandská dizertačná práca. Technická univerzita v Košiciach, FEI, KKUI, Košice, 2003.

[32] *TF-IDF* [online]. Last modified on 2 May 2013. [cit. 2013-05-10]. Dostupné na URL: <http://en.wikipedia.org/wiki/Tf-idf>.

[33] Barber J. *Simple Search: The Vector Space Model* [online]. Dostupné na URL: <http://phpir.com/simple-search-the-vector-space-model/>.

[34] Hill, T., Lewicki, P. *Statistics: Methods and Applications.* StatSoft, Inc., 2005. ISBN 978-1884233593.

[35] Cristianini, N., Shawe-Taylor, J. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press. 2000. ISBN 0-521-78019-5

[36] KaiBo, D., Keerthi, S. (2005). *Which Is the Best Multiclass SVM Method? An Empirical Study*. Proceedings of the Sixth International Workshop on Multiple Classifier Systems. ISBN 978-3-540-26306-7.

[37] *Phirehose* [online]. Dostupné na URL: <https://github.com/fennb/phirehose/wiki/Introduction>.

[38] Green, A. *140dev* [online]. Dostupné na URL: <http://140dev.com>.

[39] *RapidMiner* [online]. Dostupné na URL: <http://rapid-i.com/>.

[40] Kohavi, R. *A study of cross-validation and bootstrap for accuracy estimation and model selection*. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. 1995. (Morgan Kaufmann, San Mateo, CA)

[41] *LIBSVM* [online]. Dostupné na URL: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

[42] Barber, J. *PHP SVM* [online]. Dostupné na URL: <https://github.com/ianbarber/php-svm>.

[43] *Nette* [online]. Dostupné na URL: <http://nette.org/>.

[44] *Dibi* [online]. Dostupné na URL: <http://dibiphp.com/>.

[45] *Support vector machine* [online]. Last modified on 3 May 2013. [cit. 2013-05-10]. Dostupné na URL: <http://en.wikipedia.org/wiki/Support_vector_machine>.

Zoznam príloh

Príloha 1. CD so zdrojovými kódmi aplikácie a elektronickou verziou tejto technickej správy