SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 3310

# Povećanje učinkovitosti dohvaćanja sadržaja putem interneta

Mario Kostelac

Zagreb, srpanj 2013.

*Umjesto ove stranice umetnite izvornik Vašeg rada.*
*Da bi ste uklonili ovu stranicu obrišite naredbu* `\izvornik`.

# SADRŽAJ

# 1. Uvod

Čovjek je društveno biće i od samih početaka se organizira u različite zajednice kako bi poboljšao kvalitetu života. Iako čovjek svakim danom probija granice tehnologije koju koristi i izumljuje nevjerojatne stvari, liječi razne bolesti, razotkriva i mijenja genetski sastav biljaka kako bi povećao prinose, on i dalje zadržava iskonsku potrebu ostati povezan s ljudima koje voli (ili će tek zavoljeti). Ta potreba usmjerava čovjeka da nove tehnologije koristi i za kreiranje novih kanala komunikacije, novih načina da pokaže emocije, načina da dopre do svih do kojih mu je stalo, igra s njima i upoznaje nove ljude.

Razvojem računala čovjek brzo uviđa mogućnost povezivanja različitih računala i dijeljenja podataka među njima. Tako nastaje i USENET (Wikipedia, 2013b), jedno od prvih "virtualnih" mjesta na kojima su se mogle razmjenjivati vijesti, mišljenja i objavljivati znanstveni radovi na dobrobit cijele zajednice. Desetak godina kasnije (1990.) u laboratoriju CERN-a se stvara potreba za naprednijim i jednostavnijim dijeljenjem radova i rezultata istraživanja te tako nastaje dio Interneta bez kojeg je on danas teško zamisliv - *World Wide Web*. Nove tehnologije na Internetu se brzo prihvaćaju, na mrežu se priključuje sve više ljudi i započinje eksponencijalni rast. Internet se brzo širi sa skupine znanstvenika na sve društvene skupine - tinejdžere, odrasle, a u zadnje vrijeme i na djedove i bake. Odlaskom krештećeg zvuka povezivanja dial-up modema i uvođenjem bržih tehnologija, internet stječe još veći potencijal postajanja vrlo važnog kanala svakodnevne komunikacije.

*Facebook, Twitter, Reddit i LinkedIn* su društvene mreže koje prožimaju našu svakodnevicu. Zahvaljujući karakteru tih mreža, pristup Internetu danas znači pravo govora svim pristupnicima (unatoč pokušajima cenzure).
Twitter je 2009. "sudjelovao" u moldavijskoj revoluciji naglim širenjem glasa o njoj i tako postao sjedištem revolucionara (SpiegelOnline, 2009) . Također, protesti oko izbora u Iranu (Times, 2009), tuniška revolucija (Tufekci, 2011) i egipatska revolucija bi zasigurno imale puno manji značaj da nije društvenih mreža.
Dok Twitter pomaže u brzom širenju vijesti, Facebook smanjuje udaljenosti među lju-

dima [1] (Wikipedia, 2013a).
Zbog učestalosti korištenja i značajnosti društvenih mreža, problem pretraživanja i analiziranja sadržaja društvenih mreža postaje sve važniji.

Taj sadržaj pretražuju *crawleri* - programi za sustavno i automatizirano pretraživanje sadržaja, spremanje i daljnju obradu. Iako najpoznatiji *crawler* - Googleov web *crawler*, prilično dobro obavlja svoj posao, zbog različite forme sadržaja javlja se potreba za izradom drugačijih crawlera - onima koji će pretraživati društvene mreže na drugačiji način - korištenjem API-a društvenih mreža [2].

Ovaj rad se bavi razvojem i evaluiranjem nekoliko strategija dohvaćanja sadržaja s Facebooka. Preciznije, rad opisuje strategije povećanja efikasnosti dohvaćanja novog sadržaja Facebook stranica.

## 1.1. Motivacija i ciljevi

Facebook je svojim napretkom privukao pripadnike svih društvenih skupina. Velik broj korisnika privukao je i brandove koji su shvatili da se i ovaj kanal komunikacije može iskoristiti za marketing. Tako na toj mreži pomoću Facebook stranica Coca Cola reklamira svoj novi dizajn boca, Justin Bieber pozdravlja svoje fanove, a KLM objavljuje sniženja karata do Dublina i San Francisca.

S vremenom se javila želja i potreba za prikazom specifičnog sadržaja. Tako nastaju razni servisi takve prirode - servisi za pregledavanje glazbenog sadržaja nastalog na društvenim mrežama (npr. `http://www.hipset.com/`), servisi za pregledavanje sadržaja vezanog uz nove tehnologije, hobije itd.

Automatizirano ekstrahiranje specifičnog sadržaja pruža velike mogućnosti. Zamislimo samo da smo marketinška agencija koja želi učiti od boljih i uspješnijih brandova/kampanja. Sustav upogonjen ovakvim tehnikama uvelike bi olakšao promatranje i detaljniju analizu postupaka naših konkurenata i omogućio nam ubrzano učenje.

Kao drugi primjer možemo navesti servis koji će nam ekstrahiranjem sadržaja vezanog za glazbu prikazati što se trenutno događa u svijetu glazbe, što slušaju naši prijatelji i što bi se nama moglo svidjeti na temelju onoga što smo već slušali.

Kako sam imao priliku sudjelovati na projektiranju i izradi jednog sličnog servisa (`http://www.buzzwise.com/`, u daljnjem tekstu samo *buzzwise*), pojavila se

[1] Neki radovi pokazuju da je bilo kojoj osobi moguće preko manje od 6 osoba doći do bilo koje druge osobe

[2] Danas većina sustava pruža mogućnost vanjskom svijetu za komunikaciju sa sustavom preko određenog sučelja. Takvo sučelje nazivamo API

potreba za razumijevanjem principa efikasnog dohvaćanja sadržaja s Facebooka.

Osnovna prepreka pri dohvaćanju sadržaja s Facebook stranica su ograde koje postavlja Facebookov API - 600 poziva u 10 minuta. Cilj rada je, koristeći zadane granice, razviti tehniku dohvaćanja koja će u određenom vremenskom intervalu dohvatiti što je više moguće svježeg sadržaja kojeg generiraju Facebook stranice. Opisane tehnike će se uspoređivati s tehnikom upotrijebljenom na *buzzwise*-u te će se pokušati ostvariti bolji rezultati. Referenta tehnika i novo razvijene tehnike će biti detaljno opisane u nastavku rada.

Problem povećanja efikasnosti spada u probleme višekriterijske optimizacije. Ciljevi koje želimo postići su:

- povećanje količine dohvaćenog svježeg sadržaja
- smanjenje prosječnog vremena proteklog između nastanka i dohvaćanja sadržaja - kašnjenja

Iako je optimizacija višekriterijska, nije nužno da poboljšanje jednog kriterija implicira smanjenje drugog u cijelom prostoru rješenja.

Sve razvijene tehnike teže učenju pravilnosti u ponašanju Facebook stranica kako bi na temelju postojećeg znanja dobro predvidjele buduća ponašanja.

## 1.2. Organizacija rada

Rad je organiziran u 6 poglavlja.

U Poglavlju 2 se opisuje Facebook API kao platforma na kojoj se temelji cijelo programsko ostvarenje i čija ograničenja stvaraju potrebu za razvijanjem metoda kakve rad proučava. Opisuju se ključne metode za realizaciju ovog rada i dostupne varijacije.

Poglavlje 3 uvodi tehnike dohvaćanja novog sadržaja s Facebooka. U početku detaljnije opisuje problem koji se pokušava riješiti, a nakon toga uvodi referentnu tehniku koju se pokušava nadmašiti implementiranjem boljih tehnika. Nastavak poglavlja detaljnije opisuje implementirane tehnike.

Poglavlje 4 donosi pregled razvijenih alata koji su sudjelovali u realizaciji ovog rada.

Poglavlje 5 prikazuje rezultate implementiranih tehnika. Nakon prezentacija rezultata slijede usporedbe različitih tehnika, ocjene uspješnosti i diskusija u kojoj se iznose ideje za daljnjim poboljšanjima koristeći prikazane tehnike.

Konačno, Poglavlje 6 sažima cijeli rad i na jednom mjestu donosi sve što se postiglo tijekom rada i istraživanja potrebnih da bi se ovaj rad završio.

# 2. Društvena mreža Facebook

U ovom poglavlju se ukratko opisuje povijest Facebooka i kako je započeo razvoj ove društvene mreže. Nakon retrospektive se uvodi pojam Facebook API-a te se opisuju dijelovi od kojih se danas Facebook API sastoji. Također, detaljnije se opisuje dio Graph API-a, objekti i relacije koji su potrebni za razumijevanje rada kao cjeline.

## 2.1. Facebook - najveća društvena mreža na Internetu

Facebook je trenutno najpopularnija društvena mreža na Internetu (Wikipedia, 2011). Nastala je 2004. godine u studentskoj sobi Marka Zuckerberga, a u početnom timu su bili Mark Zuckerberg, Eduardo Saverin, Andrew McCollum, Dustin Moskovitz i Chris Hughes. Facebooku je prethodio Markov projekt Facemash, ali pošto je na cijelom sveučilištu izbio skandal zbog prirode takvog servisa (uspoređivanje djevojaka i glasanja koja je privlačnija), Mark je odlučio nazvati novi servis drugačijim imenom.

U početku je Facebook bio otvoren ekskluzivno studentima s Harvarda. Rastom popularnosti na Harvardu osnivači uviđaju potencijal otvaranja mreže i početkom 2005. postupno otvaraju mrežu sveučilište po sveučilište (prva verzija se može vidjeti na `http://www.thefacebook.us/`).

Otvaranje prolazi bolje od očekivanog, broj korisnika neprestano raste i Facebook počinje nadilaziti granice društvenih mreža koje su do tada postojale. Ubrzo se javlja i potreba za većim prihodima od reklama pa se uvodi opcija za Facebook stranice (Facebook Page) raznih brandova. Dovoljan pokazatelj važnosti ovog dijela Facebooka je podatak da je već početkom 2007., kada je Facebook imao manje od 100 milijuna korisnika, posluživao više od 100 tisuća komercijalnih stranica. Svi brandovi su htjeli doći do svojih postojećih korisnika i osvojiti nova tržišta.

## 2.2. Facebook API

Facebook s vremenom prestaje biti samo web stranica na koju dolaze milijuni korisnika i međusobno se povezuju. Facebook postaje platforma za razvoj drugih aplikacija. Da bi Facebook omogućio razvoj aplikacija na svojoj društvenoj platformi, tim programera izdaje sučelje vanjskog svijeta prema podacima na Facebooku - Facebook API.

Danas se Facebook API sastoji od više sučelja (Facebook, 2013):

**Graph API** - jednostavni API temeljen na HTTP-u koji daje pristup Facebookovom društvenom grafu, uniformno predstavljajući objekte u grafu i odnose među njima. Većina ostalih API-ja se temelji na Graph API-u

**FQL** - Facebook Query Language ili skraćeno FQL, omogućuje korištenje sučelja sličnog SQL-u radi dohvaćanja informacija iz Graph API-a. FQL podržava i neke naprednije opcije koje Graph API još nije implementirao, npr. podupite

**Dijalozi** - jednostavan API kroz koji se integriraju web stranice i Facebook pageovi/-profili ljudi. Funkcionira na način da se za različite dijaloge dobiva različit JavaScript kod koji se zalijepi unutar koda web stranice

**Chat** - Chat je integrirani dio Facebooka. Ovaj API dopušta integraciju chat-a s drugim servisima

**Ads API** - omogućava izradu vlastite aplikacije kao alternative pretpostavljenom Ads Manageru.

Kako je Graph API u pozadini većine drugih API-a i kako je on dovoljan za ostvarenje, samo će se on detaljnije obrađivati u nastavku rada.

## 2.3. Graph API

Graph API čini samu jezgru Facebooka. Implementiran je kao *HTTP stateless API* [1] što omogućava iznimno jednostavnu komunikaciju između vanjskih aplikacija i Graph API-a. Kako mu samo ime govori, Graph API radi s grafovima i u srži sadrži samo objekte (čvorove) i njihove relacije. Primjer jednog takvog objekta (u JSON [2] formatu) se nalazi ispod odlomka:

[1] HTTP stateless API - zahtjevi se šalju preko HTTP protokola i međusobno su sasvim neovisni. Ovakvo svojstvo iziskuje autentifikaciju pri svakom pozivu.

[2] JSON - Javascript Object Notation, format dokumenta koji služi za opisivanje JavaScript objekata. Danas se koristi mnogo šire od početne domene pa tako i za vraćanje odgovora na određene zahtjeve

```
{
  "id": "771789554",
  "name": "Mario Kostelac",
  "first_name": "Mario",
  "last_name": "Kostelac",
  "link": "http://www.facebook.com/mariokostelac",
  "username": "mariokostelac",
  "quotes": "Like is what you make of it.",
  "gender": "male",
  "timezone": 2,
  "locale": "en_US",
  "languages": [
    {
      "id": "106059522759137",
      "name": "English"
    },
    {
      "id": "109956882360328",
      "name": "Croatian"
    }
  ],
  "verified": true,
  "updated_time": "2013-03-07T21:22:50+0000"
}
```


Vidljivo je da je objekt opisan svojim obilježjima. Tako *name, first_name, last_name, link, username* itd. pobliže opisuju dani objekt. Zanimljivo je pogledati i "obilježje" *languages*. Iako se na prvi pogled čini kao obilježje, ono to zapravo i nije. Član languages je relacija između prikazanog objekta i jezika. Navedena lista govori da su objekti s ID-om *771789554* i objekti *106059522759137* i *109956882360328* povezani u grafu. Pošto je prvi objekt profil osobe, a druga dva objekta su jezici, povezanost među njima interpretiramo kao činjenicu da osoba koju predstavlja profil govori ta dva jezika.

Društveni graf Facebook mreže je visokog stupnja (svaka komponenta ima puno veza prema drugim objektima) pa se iz tog razloga prikazuju samo neke relacije. Ostale veze dobivamo zasebnim zahtjevom. Npr. zahtjevom za relacijom prijatelja već

viđenog objekta dobivamo sljedeći odgovor (zbog veličine prikazan samo dio odgovora):

```
{
  "data": [
    {
      "name": "Ivan␣Brezak␣Brkan",
      "id": "501341420"
    },
    {
      "name": "Jonathan␣Gray",
      "id": "501846542"
    },
    {
      "name": "Stipe␣Modric",
      "id": "521870758"
    },
    {
      "name": "Melita␣Mihaljevic",
      "id": "526274179"
    },
    ...
```

Pristupanje objektima i njihovim relacijama pomoću Graph API-a je krajnje jednostavno. Iza URI-a `http://graph.facebook.com/` se nadoda ID objekta i kao odgovor API vraća opis objekta u JSON formatu. Za pristup određenim relacijama se dodaje još i ime relacije. Tako za pristup prijateljima objekta s ID-om 771789554 dobivamo URL `http://graph.facebook.com/771789554/friends`.

Važno je napomenuti da Graph API daje samo ograničen skup informacija neautoriziranim aplikacijama (ovisno o generalnim postavkama i postavkama privatnosti pojedinog korisnika). Pristup širem skupu informacija (naravno, opet ne cijelom skupu) se dobiva kreiranjem Facebook aplikacije koja tada dobiva autorizacijske podatke i ulazi u društveni graf kao zaseban objekt.

Graph API omogućava i kreiranje novih objekata i veza među njima, ali taj aspekt se ne koristi u ovom radu pa neće biti analiziran.

U nastavku teksta će poneka referenciranja na API ili Facebook API značiti referenciranje na Facebook Graph API, no to će već biti jasno iz danog konteksta.

## 2.4. Potrebni objekti i relacije

Tehnike koje se razvijaju u okviru ovog rada pokušavaju na temelju prijašnjih objava predvidjeti kada je najveća vjerojatnost da će se objaviti nešto novo. One uče samo na temelju prošlih objava te nam je u skladu s tim potreban pristup sljedećim entitetima:

**objekti razreda Page** - objekt Facebook Stranice za koju želimo naučiti navike i predviđati sljedeće objave - dostupno preko `http://graph.facebook.com/%pageID%/`

**relacija posts za navedenu klasu objekata** - relacija nam govori koje objave pripadaju određenoj stranici - dostupno preko `http://graph.facebook.com/%pageID%/posts`

Primjer rezultata: dio dohvaćenih rezultata s `http://graph.facebook.com/Palantir/posts`


```
{
  "data": [
    {
      "id": "134287640500_10152792255000501",
      "from": ...,
      "message": "A week long dive into the design future ...",
      "picture": "http://photos-f.ak.fbcdn.net/hphotos-ak...",
      "link": "http://www.facebook.com/photo.php?fbid=101...",
      "name": "Design In Action",
      "caption": "A week long dive into the design future ...",
      "icon": "http://static.ak.fbcdn.net/rsrc.php/v2/yz/...",
      "actions": ...,
      "privacy": ...,
      "type": "photo",
      "status_type": "added_photos",
      "object_id": "10152792249090501",
      "created_time": "2013-04-24T19:08:05+0000",
      "updated_time": "2013-04-24T19:08:05+0000",
      "likes": ... ,
```

## 2.5. Ograničenja API-a i potreba za strategijama dohvaćanja

Kako bi Facebook privukao što više developera, kao i drugi servisi te vrste, korištenje API-a ne naplaćuje ni u kojem smislu. Nažalost, i Facebook pogone računala konačne snage pa se i njihovi inženjeri suočavaju s problemima složenosti i konačnih resursa. Zbog navedenih problema postavljaju se ograničenja na broj poziva. Dok neke društvene mreže postavljaju ograničenja vrlo nisko (Twitter za neautorizirane aplikacije 300 poziva na sat, (Čajić, 2012)), Facebook za relevantnu tablicu postavlja ograničenja od 600 poziva u 10 minuta po aplikaciji (svibanj 2013.). Ovo ograničenje se čini dosta nesputavajuće (prosječno 1 poziv u sekundi), ali porastom broja stranica koje želimo pratiti vrlo brzo nailazimo na problem zastarjelih podataka i praznih poziva na API. [3]

Iako možemo kreirati više aplikacija i paralelizirati dohvaćanje, struka nas navodi da rad svake pojedine instance optimiziramo kako bi računala odrađivala što manje poziva u kojima neće biti dohvaćeno ništa. Također, svaki poziv troši određene resurse pa nam pametno korištenje poziva u nekom trenutku može smanjiti potrebu za novim računalima, što direktno povlači i smanjenje potrebnih financijskih sredstava koje trošimo na računalnu snagu.

Radi potpunosti informacije, treba spomenuti da Facebook svojim partnerima nudi mogućnost uklanjanja tih ograničenja. Nažalost, ako projekt nije dovoljno prosperitetan da Facebook prepozna potencijalni rast vrijednosti svojih dionica koju bi ostvario suradnjom, aplikacija se mora nositi sa spomenutim ograničenjima.

[3] Prazni pozivi su svi oni koji nisu uspjeli dohvatiti niti jednu novu objavu. Oni se također ubrajaju u sumu učinjenih poziva.

# 3. Problem i tehnike dohvaćanja

U ovom poglavlju se opisuju tehnike evoluiranja strategija dohvaćanja sadržaja s Facebooka.

Na samom početku se elaborira problem i opisuje njegova težina.

U drugom dijelu poglavlja se uvodi jednostavna tehnika implementirana na *buzzwise* sustavu i opisuju se njena ograničenja te problemi s kojima se susreće. Nakon toga slijedi detaljniji opis tehnike preuzete iz diplomskog rada Andrije Čajića (Čajić, 2012). Vrlo lako će se uvidjeti analogija i razlog zašto se izvedena tehnika može primijeniti na problem čije se rješenje traži.

Na kraju poglavlja se donosi primjer u kojem se na malom skupu podataka demonstrira rad tehnike *air balloon*.

U nastavku rada će se ponekad referentna tehnika referencirati kao *buzzwise*, a tehnika preuzeta iz diplomskog rada kao *air balloon* tehnika.

## 3.1. Problem

Problem kojim se bavi ovaj rad spada u probleme raspoređivanja, tj. izrade rasporeda dohvaćanja sadržaja iz različitih izvora.

Raspolažemo poznatim skupom izvora sadržaja koji ga obnavljaju neovisno jedni o drugima. Cilj rada je pronaći tehniku koja prepoznaje pravilnosti u pojavnostima trenutaka obnove sadržaja i na temelju prepoznatih uzoraka ponašanja kreira odgovarajući raspored dohvaćanja kako bi dohvatio što kvalitetniji sadržaj (definicija kvalitete u nastavku poglavlja).

Ovaj rad opisuje primjenu tehnika raspoređivanja na dohvaćanja sadržaja s Facebook stranica. Izvore sadržaja predstavljaju Facebook stranice o kojima sustav na početku rada nema nikakva znanja osim lokacija na kojima se njihov sadržaj može dohvatiti. Obnovu sadržaja predstavljaju nove objave koje određen izvor objavi. Tijekom rada sustav uči o izvorima i na temelju toga izrađuje raspored dohvaćanja.

Sada kada su definirani osnovni pojmovi u sustavu (i njihove konkretizacije), još

samo ostaje definirati pojam kvalitetnog sadržaja. Iako postoje čitave znanstvene discipline koje se bave definiranjem kvalitete i mjerenjem iste, vrlo je lako spoznati da je kvalitetu teško definirati i da je domenski specifičan pojam.

Tako ćemo u ovom slučaju kvalitetu definirati kao zadovoljstvo imaginarnog korisnika koji bi koristio sustav koji mu prikazuje sadržaj stranica koje mu se sviđaju (preferencije je korisnik eksplicitno izrazio). Da zadatak ne bi bio prejednostavan, zamislimo i da sustav koristi puno različitih korisnika pa sustav nije u mogućnosti sve resurse usmjeriti usluživanju jednog korisnika.

Očekujemo da je korisnik zadovoljniji što mu je sadržaj svježiji. Također, korisnik je zadovoljniji što mu je prikazan veći dio ukupno generiranog sadržaja.

Iz dviju navedenih pretpostavki slijedi da želimo optimizirati dva parametra:

- maksimizirati količinu dohvaćenog sadržaja
- minimizirati vrijeme proteklo od kreiranja do dohvaćanja sadržaja.

Uzevši u obzir ograničenje API-a, nije teško zaključiti da se radi o problemu višekriterijske optimizacije. Višekriterijska optimizacija je takva optimizacija koja optimiranjem jednog parametra može narušavati optimizaciju ostalih parametara.

U ovom slučaju, ako bi htjeli samo minimizirati vrijeme proteklo od kreiranja do dohvaćanja dohvaćanja sadržaja, mogli bi zanemariti sve izvore osim jednog i sve pozive iskoristiti na njega. Čak i uniformna razdioba poziva (1 poziv po sekundi) bi dala sjajne rezultate. Nažalost, ovakvim pristupom bi dohvatili neznatan dio ukupnog sadržaja i time znatno narušili uspješnost optimizacije prvog kriterija.

Opisanim problemom bavi se i rad Optimal Crawling Strategies for Web Search Engines (Wolf et al., 2002), ali u domeni web tražilica. Nakon što određena tražilica sazna za određen izvor i doda ga u bazu izvora, potrebno je periodično provjeravati postoje li promjene u sadržaju kojeg izvor emitira.

## 3.2. Referentna tehnika

Referentna tehnika čije se rezultate želi poboljšati je tehnika implementirana na sustavu `http://www.buzzwise.com`, u čijoj sam izradi izravno sudjelovao.

Ideja ove tehnike je vrlo jednostavna - od cijelog skupa izvora sadržaja uzet ćemo fiksni podskup i dohvaćati samo njihov sadržaj. Na početku rada ćemo sve izvore dodati u red (*engl. queue*) i redom ih vaditi te dohvaćati njihov sadržaj u nadi da je kreiran neki novi sadržaj neposredno prije nego smo napravili poziv na API. U trenutku kada stranicu izvadimo iz reda i napravimo zahtjev za dohvatom njenog sadržaja, stavljamo

je na kraj reda kako bi isti zadatak ponovno izvršili nakon što obavimo sve zadatke koji su trenutno u redu čekanja. Navedena tehnika osigurava neprestano dohvaćanje sadržaja i osigurava jednak udio resursa svakom izvoru sadržaja.

Veličina podskupa izvora čiji sadržaj dohvaćamo (označimo veličinu s $A$) izravno utječe na prosječno kašnjenje (označimo je s $B$). Matematički zapisano, vrijedi $A/B = konstanta$, tj. navedene veličine su obrnuto proporcionalne.

## 3.3. Tehnika air balloon

Tehnika kojom ćemo pokušati dobiti rezultate bolje od referentne tehnike nazvat ćemo tehnika *air balloon* (Čajić, 2012). Ime je dobila zbog analogije sa zračnim balonom - ponašanje tehnike će se zasnivati na hlađenju i zagrijavanju kao što se u stvarnosti upravlja zračnim balonom.

Ideja kojom se dolazi do ove tehnike je vrlo jednostavna i možemo je objasniti kao nadgradnju na *buzzwise* tehniku raspoređivanja.

Krenut ćemo od zahtjeva da želimo povećati podskup izvora čiji sadržaj dohvaćati bez linearnog povećanja kašnjenja. Iskustveno znamo da različiti izvori (bile to web stranice, Facebook stranice, papirnati magazini...) emitiraju sadržaj različitim frekvencijama. Ako bi na neki način uspjeli kategorizirati stranice u dvije kategorije (jedna u kojoj su stranice koje obnavljaju sadržaj frekvencijom $< f$, druga u kojoj se nalaze stranice koje obnavljaju sadržaj frekvencijom $>= f$), mogli bi pronaći tehniku koja će nam omogućiti da više vremena/resursa trošimo na stranice koje češće obnavljaju sadržaj. Pažljivom kategorizacijom nije teško postići povećanje količine dohvaćenog sadržaja bez povećanja kašnjenja. Moguće je postići i smanjenje kašnjenja, ali, naravno, za to ne postoji garancija. Također, potrebno je napomenuti da je pretpostavka da je skup izvora stranica reprezentativan skup Facebook stranica, tj. da u njemu postoje stranice koje obnavljaju sadržaj različitim frekvencijama.

Vrlo vjerojatno je da ćemo daljnjim razmatranjem zaključiti da bi mogli dvije postojeće kategorije podijeliti na još nekoliko i tako još finije kategorizirati izvore. Svaku kategoriju izvora možemo shvatiti kao prioritet kojim se sadržaj dohvaća, što povlači da je kategoriziranje izvora istovjetno dodjeljivanju prioriteta.

Da je ovakva tehnika učinkovita u dohvaćanju sadržaja s društvene mreže Twitter, pokazala je platforma *TwitterEcho* (Bošnjak et al., 2012).

Daljnje poboljšanje ove tehnike možemo pronaći ako se sjetimo da stranice mijenjaju svoje navike. Brandovi propadaju i prestaju obnavljati sadržaj, reklamne kampanje imaju određen vijek trajanja, količina entuzijazma vezana za određene pokrete se

mijenja itd. Svaka ovakva promjena izravno utječe na promjenu frekvencije emitiranja nekog izvora. Ako ne bi kontrolirali svaku pojedinu stranicu i provjeravali ima li ona i dalje dobar prioritet, lako je moguće da bi nakon nekog vremena imali zastarjele podatke i sustav bi određene resurse trošio zbog loše podešenih prioriteta. Također je potrebno uvidjeti da ovakve promjene obavlja čovjek manualnim radom, što ne želimo.

Uz različitu frekvenciju obnavljanja korisno je primijetiti da stranice ne obnavljaju sadržaj tijekom cijelog dana već u određenim rasponima. Razlog je jednostavan - taj sadržaj obnavljaju ljudi koji imaju svoje radno vrijeme. Tako dolazimo do novog poboljšanja. Podijelit ćemo dan na nekoliko dijelova i pretpostaviti:

- ako u posljednje vrijeme određena stranica obnavlja sadržaj često oko nekog vremena, vrlo je vjerojatno da će i ubuduće biti tako
- ako u nekom vremenskom rasponu stranica ne obnavlja sadržaj, vrlo je vjerojatno da to neće raditi ni u budućnosti
- navike ljudi koji obnavljaju stranice se mogu mijenjati.

Podijelimo li dan na 24 sata i definiramo sate kao dijelove dana po kojima pratimo aktivnosti, dobit ćemo konačnu tehniku.

Dobivenom tehnikom riješili smo problem manualnog rada čovjeka i problem fiksnog broja prioriteta.

U nastavku poglavlja bit će objašnjeni detalji tehnike.

### 3.3.1. Pretinci i aktivnost stranice

Aktivnost stranice (vjerojatnost da će sadržaj biti obnovljen) bilježi se po pretincima - svaki sat u danu predstavlja poseban pretinac. To znači da je svaki izvor sadržaja opisan s 24 vrijednosti koje nam govore koliko je vjerojatno da će se obnoviti sadržaj u određenom satu (vrijednost u svakom pretincu označava aktivnost pripadnog sata). Zbog jednostavnosti vjerojatnost nije u rasponu $[0, 1]$ već u intervalu $< 0, \infty >$, ali je zadržana semantika da veći broj znači veću vjerojatnost obnove.

Da bi sustav iskoristio prvu pretpostavku i uključio ju u učenje, uvodimo postupak zagrijavanja.

### 3.3.2. Zagrijavanje aktivnosti

Zagrijavanje je postupak koji *air balloon* tehnici izravno omogućuje pamćenje navika određenog izvora. Ono zahvaća određeni pretinac (dakle, zagrijavamo pretince, neki od njih 24) i povećava aktivnost izvora u tom pretincu.

Zagrijavanje koristimo s prvom pretpostavkom: za svaki novi sadržaj ćemo izvoru oko vremena kada je sadržaj kreiran povećati aktivnost. Također, u obzir ćemo uzeti i koliko je star sadržaj kojeg smo dohvatili. U slučaju da smo dohvatili sadržaj star mjesec dana, manje ćemo povećati aktivnost nego u slučaju da smo dohvatili sadržaj star 2 sata. Objašnjenje ovakvog postupka je vrlo jednostavno - u vremenu između kreiranja sadržaja i sadašnjeg trenutka su se karakteristike izvora mogle promijeniti (treća pretpostavka).

Matematički model zagrijavanja preuzet je iz rada (Čajić, 2012) i zadovoljava sve što smo već opisali:

$$a_{nakon} = a_{prije} + a_{prije} * e^{\frac{-vrijeme}{inercija}} \tag{3.1}$$

gdje,
$a$ predstavlja aktivnost zahvaćenog pretinca,
$vrijeme$ predstavlja vrijeme proteklo od kreiranja sadržaja do njegovog dohvaćanja i
$inercija$ predstavlja parametar koji definira težinu/lakoću učenja.

Iako se može pričiniti kako je ideja zagrijavanja genijalna, samim zagrijavanjem nećemo daleko dogurati. Vrijednosti aktivnosti samo će rasti, što znači da će stranicama koje su u početnom stanju prve došle na red porasti vrijednosti aktivnosti (ako su kreirale neki sadržaj) te će ostale stranice doći u podređen položaj i možda nikada neće doći na red. Zbog tog problema uvodimo obratan postupak - *postupak hlađenja*.

Razlog zbog kojeg je odabrana eksponencijalna funkcija bit će objašnjen u dijelu poglavlja u kojem se obrađuje hlađenje aktivnosti.

### 3.3.3. Postupak hlađenja

Postupak hlađenja simulira proces obrnut procesu zagrijavanja. Ako se prisjetimo pretpostavki koje smo primijetili u sustavu koji promatramo i ako promotrimo proces zagrijavanja, lako zaključujemo da samim procesom zagrijavanja možemo iskoristiti prvu i dio druge pretpostavke. Ako stranica redovito obnavlja sadržaj u određenom dobu dana, sustav će zagrijavati aktivnost baš u tom dobu. U ostalim dijelovima dana aktivnosti neće biti zagrijavane i njihova vrijednost će biti niža od vrijednosti aktivnosti u kojima se obično sadržaj obnavlja.

Uz primjedbu da na neki način moramo spriječiti da se vrijednosti aktivnosti samo povećavaju, bilo bi dobro iskoristiti i treću pretpostavku, tj. modelirati tehniku tako da je sposobna uzeti u obzir i promjene ponašanja izvora. Nasreću, oba problema možemo riješiti jednim procesom - procesom hlađenja.

Proces hlađenja ćemo simulirati tako da prije računanja aktivnosti pokrenemo hlađenje i po nekom modelu smanjujemo vrijednosti aktivnosti po pojedinim pretincima.

Model hlađenja analogan je modelu zagrijavanja. Pratit ćemo vrijeme proteklo od zadnjeg hlađenja i u skladu s tim mijenjati vrijednosti aktivnosti u svim pretincima.

Izraz koji definira nove vrijednosti pojedinih aktivnosti je:

$$a_{nakon} = a_{prije} * e^{\frac{-vrijeme}{inercija}} \tag{3.2}$$

gdje
$i$ predstavlja indeks pretinca za koji mijenjamo aktivnost
$vrijeme$ označava vrijeme proteklo od zadnjeg hlađenja
$inercija$ predstavlja parametar koji definira težinu/lakoću učenja.

Ako primijetimo da ćemo i hlađenje i zagrijavanje simulirati na računalima koji rade u diskretnim trenucima, jasno je zašto je odabran ovaj model. Eksponencijalna funkcija će nam pomoći u simuliranju kontinuiranog hlađenja iako ga izvodimo u konkretnim trenucima. Svojstvo eksponencijalne funkcije (Husch, 2001) da $a^r * a^s = a^{r+s}$ nam omogućava sljedeću stvar:
aktivnosti ćemo hladiti u diskretnim trenucima (točno prije nego trebamo izračunati vrijednost aktivnosti stranice), a rezultat će biti jednak kao da se cijeli proces odvijao sasvim kontinuirano.

### 3.3.4. Redoslijed dohvaćanja stranica

Procesima hlađenja i zagrijavanja smo došli do tehnika koja će u obzir uzeti sve pretpostavke koje smo naveli, ali oni sami po sebi nisu od velikog značaja. U sustav još moramo uvesti strategiju kojom ćemo dohvaćati stranice. Ako bolje razmotrimo problem, shvatit ćemo da problem i jest kreiranje strategije dohvaćanja sadržaja. Zbog uvedenih procesa nije potreban veliki trud da bi dovršili ovu strategiju i potrebno je još samo uvesti pojam *akumulirane aktivnosti*.

*Akumulirana aktivnost* je sumirana aktivnost od zadnjeg dohvaćanja do trenutka ponovnog dohvaćanja. Dohvaćanjem sadržaja neke stranice se ova vrijednost postavlja na 0 i njeno računanje kreće ispočetka. Algoritam računanja akumuliranje aktivnosti prolazi po grafu i računa površine ispod krivulje koja opisuje aktivnost stranice po pojedinim satima.

Akumulirana vrijednost će izravno utjecati na redoslijed dohvaćanja - veća akumulirana aktivnost znači i veći prioritet u redu čekanja.

Činjenica da najveća aktivnost znači prvo mjesto u redu čekanja i da nakon dohvaćanja resetiramo akumuliranu vrijednost znači da nakon dohvaćanja sadržaja određene stranice tu istu stranicu bacamo na posljednje mjesto u redu čekanja.

Ovakvo ponašanje smo već vidjeli kod *buzzwise* tehnike čije rezultate želimo poboljšati pa se postavlja pitanje koja je razlika između ove tehnike i tehnike *buzzwise*.

Dok se kod *buzzwise* tehnike sve stranice jednoliko penju prema vrhu reda čekanja (pretpostavimo da se uzima s vrha), *air balloon* tehnika praćenjem aktivnosti i računanjem vrijednosti akumulirane aktivnosti različitim stranicama omogućuje da se različitim brzinama penju prema vrhu reda čekanja. Brzine penjanja stranica prema vrhu su svojstvene svakoj stranici i ovise samo o prijašnjim navikama obnavljanja sadržaja, a to je točno ono to smo htjeli postići novom tehnikom.

Prisjetimo se da i da smo htjeli kreirati neograničen broj stupnjeva prioriteta. Ovakvim računanjem akumulirane aktivnosti smo to i postigli - vrijednost je iz skupa pozitivnih brojeva.

Sada kada je detaljiziran cijeli postupak dohvaćanja, potrebno je objasniti još neke detalje - početne vrijednosti aktivnosti i parametar inercije.

### 3.3.5. Početne vrijednosti

Semantika početnih vrijednosti aktivnosti pojedinih stranica i nije najjasnija. Ono što bismo htjeli je da početna vrijednost aktivnosti ima jednako značenje kao i vrijednosti aktivnosti u bilo kojem budućem trenutku, tj. da dobro ocrtava prošlo ponašanje nekog izvora. Naravno, ispunjenje ovakve pretpostavke je teško ostvarivo. Ovakvu tehniku smo i kreirali da bi u određenim trenucima znali vrijednosti takvih aktivnosti. Ako bi u određenom trenutku znali te vrijednosti, to bi značilo da takvu tehniku već imamo razvijenu, što povlači da takvu tehniku ne trebamo razvijati.

Unatoč nedefiniranosti početnih vrijednosti, znamo da bi one trebale biti pozitivne. Iskorišteni model zagrijavanja i hlađenja bi negativne vrijednosti samo povećavao i dobili bi rezultate točno obrnute od željenih - aktivnije stranice bi bile rjeđe dohvaćane.

Ako bi početna vrijednost bila nula, model zagrijavanja i hlađenja nikada ne bi promijenio početnu vrijednost i sustav bi bio beskonačno inertan. Iz istog razloga sustav nikada ne smije dozvoliti pad vrijednosti aktivnosti u pojedinom pretincu na 0.

Zbog jednostavnosti su početne aktivnosti u svim pretincima postavljene na 1.

### 3.3.6. Parametar inercije

Parametar inercije je vrijednost koja opisuje koliko je sustav podložan brzom učenju / zaboravljanju. Iz samog matematičkog modela hlađenja i zagrijavanja nije lako utvrditi koja bi vrijednost mogla dati dobre rezultate pa će se u poglavlju rezultata pokretati više simulacija s različitim vrijednostima parametara inercije kako bi približno utvrdili najbolju vrijednost za dani skup izvora.

### 3.3.7. Primjer rada algoritma

Kako bi se dodatno pojasnio rad ovog algoritma, u nastavku se donosi primjer rada na malom skupu podataka. Za trajanje poziva se pretpostavlja da traje beskonačno kratko, tj. da sustavu na oduzima vrijeme za svoje izvođenje. Parametar inercije je postavljen na 168 sati.

Skup podataka se sastoji od 3 stranice i nekoliko njihovih objava. Stranice su nazvane *Stranica1*, *Stranica2* i *Stranica3*. *Stranica1* najčešće objavljuje novi sadržaj, *Stranica2* ga objavljuje otprilike duplo manje, a *Stranica3* svoj sadržaj obnavlja samo jednom. Podsustav koji simulira Facebook API je ograničen na 2 poziva u 10 minuta (granica je snižena jer je skup izvora malen).

Vremena objava:

```
Stranica 1:
2013-06-05 08:05
2013-06-05 08:15
2013-06-05 08:18
2013-06-05 08:22
2013-06-05 08:25
2013-06-05 08:31
2013-06-05 08:37

Stranica2
2013-06-05 08:03
2013-06-05 08:15
2013-06-05 08:27
2013-06-05 08:32

Stranica3
```

```
2013-06-05 08:17
```

Simulacija cijelog sustava počinje 5.6.2013. godine u 8:00 sati, a završava kratko nakon zadnje promjene sadržaja u sustavu.

```
Vrijeme u sustavu: 2013-06-05 08:00:00
  sustav bira stranicu : (Stranica1, 0)
  sustav dohvaca objave za Stranicu1 : {}
```

"Sustav bira" označava radnju dohvaćanja stranice s najvećom akumuliranom vrijednosti. Dohvaćen je par *(Stranica1, 0)*, što znači da je dohvaćena *Stranica1*, a njena akumulirana aktivnost je 0. Stranica još nema akumuliranu aktivnost jer je simulacija tek započela i vrijeme simulacije je jednako vremenu početka simulacije.

```
  sustav bira stranicu : (Stranica2, 0)
  sustav dohvaca objave Stranice2 : {}
```

Sustav je obavio 2 poziva i mora čekati da prođe 10 minuta kako bi mogao obaviti dva nova poziva.

```
Vrijeme u sustavu: 2013-06-05 08:10:00
  sustav hladi Stranicu1
  sustav hladi Stranicu2
  sustav hladi Stranicu3
```

Kako je prošlo nešto vremena (točnije, 10 minuta) otkad su zadnji put izračunate akumulirane aktivnosti za svaku stranicu, sustav mora osvježiti aktivnosti po satima (ohladiti ih) i ponovno izračunati akumulirane aktivnosti.

```
  sustav bira stranicu : (Stranica1, 0.167)
  sustav dohvaca objave Stranice1 : {2013-06-05 08:05}
  sustav zagrijava aktivnosti Stranice1
    b[8] : 0.999, b[9] : 0.999
    a[8] : 1.915, b[9] : 1.082
```

Sustav je dohvatio jednu objavu Stranice1 i zbog toga zagrijava njene aktivnosti. Kako je dohvaćena objava kreirana u 08:03, zagrijavamo aktivnosti u pretincima 8 i 9, obrnuto proporcionalno blizinom objave svakom satu. $b[x]$ označava aktivnost u pretincu $x$ prije zagrijavanja, a $a[x]$ aktivnost u pretincu $x$ nakon zagrijavanja. Slijedi izračun kojim smo došli do novih vrijednosti aktivnosti (iskorišten model opisan u prethodnim poglavljima).

$inercija = 168$

$vrijeme1 = \frac{60-5}{60}$
$vrijeme2 = 1 - vrijeme1$
$a[8] = b[8] + b[8] * e^{\frac{-vrijeme1}{inercija}}$
$a[9] = b[9] + b[9] * e^{\frac{-vrijeme2}{inercija}}$

Do sada su objašnjeni svi osnovni postupci pa će se u nastavku pokazati samo bilješke sustava u formatu kakvom su bili prikazivani do sada.

```
  sustav bira stranicu : (Stranica2, 0.167)
  sustav dohvaca objave Stranice2 : {2013-06-05 08:03}
  sustav zagrijava aktivnosti Stranice2
    b[8] : 0.999, b[9] : 0.999
    a[8] : 1.948, b[9] : 1.049

Vrijeme u sustavu: 2013-06-05 08:20:00
  sustav hladi Stranicu1
  sustav hladi Stranicu2
  sustav hladi Stranicu3

  sustav bira stranicu : (Stranica3, 0.333)
  sustav dohvaca objave Stranice3 : {2013-06-05 08:17}
  sustav zagrijava aktivnosti Stranice3
    b[8] : 0.998, b[9] : 0.998
    a[8] : 1.714, b[9] : 1.281

  sustav bira stranicu : (Stranica1, 0.238)
  sustav dohvaca objave Stranice1 : {2013-06-05 08:15, 08:18}
  sustav zagrijava aktivnosti Stranice1 (prva objava)
    b[8] : 1.913, b[9] : 1.081
    a[8] : 2.663, b[9] : 1.331
  sustav zagrijava aktivnosti Stranice1 (druga objava)
    b[8] : 2.663, b[9] : 1.331
    a[8] : 3.363, b[9] : 1.631

Vrijeme u sustavu: 2013-06-05 08:30:00
  sustav hladi Stranicu1
  sustav hladi Stranicu2
```

```
  sustav hladi Stranicu3

  sustav bira stranicu : (Stranica2, 0.474)
  sustav dohvaca objave Stranice2 : {2013-06-05 08:15, 08:27}
  sustav zagrijava aktivnosti Stranice2 (prva objava)
    b[8] : 1.944, b[9] : 1.047
    a[8] : 2.693, b[9] : 1.297
  sustav zagrijava aktivnosti Stranice2 (druga objava)
    b[8] : 2.693, b[9] : 1.297
    a[8] : 3.243, b[9] : 1.746

  sustav bira stranicu : (Stranica1, 0.440)
  sustav dohvaca objave Stranice1 : {2013-06-05 08:22, 08:25}
  sustav zagrijava aktivnosti Stranice1 (prva objava)
    b[8] : 3.359, b[9] : 1.630
    a[8] : 3.993, b[9] : 1.996
  sustav zagrijava aktivnosti Stranice1 (druga objava)
    b[8] : 3.993, b[9] : 1.996
    a[8] : 4.575, b[9] : 2.412

Vrijeme u sustavu: 2013-06-05 08:40:00
  sustav hladi Stranicu1
  sustav hladi Stranicu2
  sustav hladi Stranicu3

  sustav bira stranicu : (Stranica1, 0.552)
  sustav dohvaca objave Stranice1 : {2013-06-05 08:31, 08:37}
  sustav zagrijava aktivnosti Stranice1 (prva objava)
    b[8] : 4.571, b[9] : 2.410
    a[8] : 5.054, b[9] : 2.926
  sustav zagrijava aktivnosti Stranice1 (druga objava)
    b[8] : 5.054, b[9] : 2.926
    a[8] : 5.437, b[9] : 3.543

  sustav bira stranicu : (Stranica3, 0.474)
  sustav dohvaca objave Stranice3 : {}
```

```
Vrijeme u sustavu: 2013-06-05 08:50:00
  sustav hladi Stranicu1
  sustav hladi Stranicu2
  sustav hladi Stranicu3

  sustav bira stranicu : (Stranica2, 0.747)
  sustav dohvaca objave Stranice2 : {2013-06-05 08:32}
  sustav zagrijava aktivnosti Stranice2
    b[8] : 3.237, b[9] : 1.743
    a[8] : 3.703, b[9] : 2.275

  sustav bira stranicu : (Stranica1, 0.669)
  sustav dohvaca objave Stranice1 : {}

Vrijeme u sustavu: 2013-06-05 09:00:00
  sustav hladi Stranicu1
  sustav hladi Stranicu2
  sustav hladi Stranicu3
  ...
```

Sustav je dohvatio sve objave i njegov daljnji rad se sastoji samo od hlađenja aktivnosti i praznih poziva.

# 4. Implementacija

Poglavlje se bavi implementacijom sustava koji podržava simulaciju i testiranje obrađenih tehnika te problemima koji su se tijekom implementacije pojavljivali.

Na početku se donosi popis svih podsustava razvijenih tijekom rada na ovom dokumentu i navode se tehnologije u kojima su podsustavi rađeni te alati u kojima se razvoj odvijao. Nakon toga se svaki podsustav opisuje detaljnije i donosi se njegova struktura.

Na kraju poglavlja slijede objašnjenja implementacija pojedinih tehnika - *air balloon* i *buzzwise*.

## 4.1. Podsustavi

Implementacija sustava sastoji se od četiri podsustava:

- podsustav za dohvat podataka
- podsustav za simuliranje i razvijanje tehnika dohvaćanja
- podsustav za slijedno pokretanje i analizu rezultata
- baza podataka

Podsustav za dohvat podataka je dio sustava zadužen za dohvaćanje popisa popularnih Facebook stranica i njihovih objava (njihovog sadržaja). Njegovo izvođenje mora biti gotovo prvo i tek nakon kraja njegovog rada ostali sustavi mogu početi s radom. Ovaj podsustav je jedini dio sustava koji komunicira s Facebook Graph API-om - svi ostali dijelovi se bave simulacijama (između ostalog, i simulacijom Graph API-a).

Podsustav za simuliranje i razvijanje tehnika dohvaćanja pruža okruženje za simuliranje postojećih i razvoj novih strategija dohvaćanja sadržaja s Facebook stranica. Ovaj podsustav čini glavninu implementacije većeg sustava.

Podsustav za slijedno pokretanje i analizu rezultata služi za slijedno pokretanje sustava za simulaciju s različitim parametrima. Nakon što sustav za simuliranje završi, ovaj podsustav analizira i pakira rezultate.

Baza podataka služi kao spremište podataka.

## 4.2. Podsustav za dohvat podataka

Zadatak podsustava za dohvat podataka je skupljanje popisa popularnih Facebook stranica i dohvaćanja njihovog sadržaja.

Ovaj dio sustava je implementiran kao skup skripti napisanih u PHP-u.

Skripta za dohvaćanje popisa stranica iz tekstualne datoteke čita popis URL-ova na kojima bi se mogle nalaziti poveznice na Facebook stranice. Skripta dohvaća sadržaj tih URL-ova, parsira sadržaj i pronađene Facebook stranice sprema u bazu podataka. Parsiranje dohvaćenog sadržaja (HTML dokument) i olakšano pretraživanje prepušteno je biblioteci phpQuery [1]. Postupak koji ova skripta obavlja naziva se *scraping* (SCHRENK, 2007).

Skripta za dohvaćanje sadržaja od baze podataka preuzima popis dohvaćenih stranica i kontaktira Facebook Graph API kako bi preuzela njihov sadržaj. Preuzima se sav sadržaj od datuma zadanog kao parametar pri pokretanju do datuma kada je skripta pokrenuta. Sav dohvaćeni sadržaj se sprema u bazu podataka.

## 4.3. Podsustav za slijedno pokretanje i analizu podataka

Zadatak ovog dijela sustava je olakšati slijedno pokretanje simulacija s različitim parametrima i skupljanje podataka o simulaciji nakon što ona završi.

Cijeli podsustav implementiran je kao nekoliko skripti u jeziku PHP. Podsustav funkcionira tako da čita JSON datoteku u koju korisnik unosi postavke poslova i pokreće te poslove jedan za drugim. Završetkom svakog posla pokreće se skripta za analizu rezultata koja analizira rezultate po tjednima i zapisuje ih datoteku *stats.txt*. Po završetku sve datoteke smjesti u direktorij koji nazove onako kako smo definirali u JSON datoteci.

Primjer takve JSON datoteke je dan u nastavku:

```
{
  "runs": [
```

[1] phpQuery - biblioteka koja omogućuje jednostavno parsiranje HTML datoteka i selektiranje elemenata imitacijom jQuery-a

```
    {
      "name": "run1",
      "method": "buzzwise",
    },
    {
      // ime pokretanja i direktorija gdje ce biti rezultati
      "name": "run2",
      // tehnika koja se koristi za generiranje rasporeda
      "method": "airballoon",
      // parametri pri pokretanju simulacije, opcionalni
      "params": {
        // parametar inercije za tehniku airballoon
        "inertia": 2440,
        "start": "2012-10-01",
        "end": "2012-10-10"
      }
    }
  ]
}
```

## 4.4. Baza podataka

Baza podataka je središnje spremište sustava i koristi se za spremanje popisa Facebook stranica i njihovog sadržaja.

Kao i za sve ostale podsustave, odlučio sam koristiti open source tehnologije. Kako s *MySQL* bazom podataka imam već dosta iskustva, odlučio sam je koristiti kao glavno spremište podataka i u ovom sustavu.

Baza podataka je vrlo jednostavna - sastoji se od dvije tablice: *pages* i *posts*.

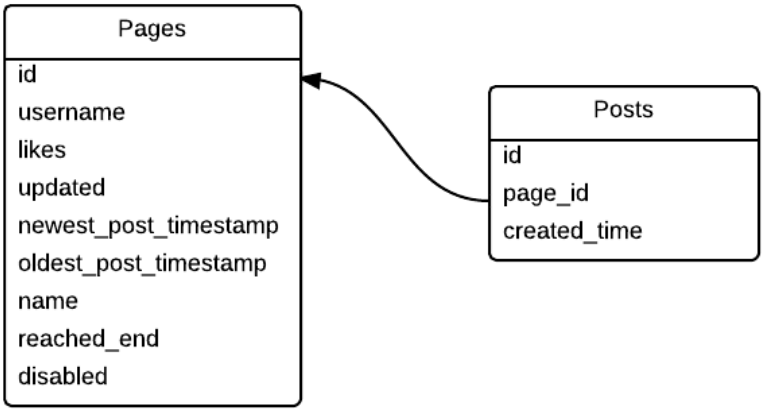
Tablica pages čuva podatke o stranicama - izvorima sadržaja, a tablica posts čuva potrebne informacije o objavama Facebook stranica - izvora sadržaja.

U nastavku je dan dijagram entiteta i njihov opis.

*Tablica Pages*

**id** - ID stranice, identičan ID-u stranice na Facebooku

**username** - korisničko ime Facebook stranice




**Slika 4.1:** Shema entiteta i veza među njima u bazi podataka

**likes** - broj korisnika kojima se Facebook stranica sviđa

**name** - puno ime stranice na Facebooku

. . . - ostala polja su privremena polja potrebna u procesu dohvaćanja sadržaja

*Tablica Posts*

**id** - ID objave, identičan ID-u objave na Facebooku

**page_id** - ID izvora podataka, stvara vezu između izvora i objave

**created_time** - vrijeme kreiranja objave

## 4.5. Podsustav za simuliranje i razvijanje strategija

Podsustav za simuliranje i razvijanje tehnika je središnji dio sustava. Njegov zadatak je pružanje okoline koja će pomoći u razvijanju i simuliranju strategija dohvaćanja sadržaja.

Izvođenju ovog podsustava prethodi završetak rada podsustava za dohvaćanje podataka. Sustav koristi bazu podataka kao spremište, tj. iz njega dobiva podatke koji su potrebni za simulaciju, a pokrenut je pomoću podsustava za slijedno pokretanje i analizu.

Kako su simulacije računski intenzivne, implementacija u bilo kojem skriptnom jeziku bi bila neprimjerena. Zbog toga je ovaj podsustav implementiran u programskom jeziku Java (verzija 7).

U odlomku o sustavu za dohvaćanje je spomenuto da je to jedini podsustav koji kontaktira Graph API. Naime, da bi simulacija imala ikakvog smisla, u podsustav za simuliranje mora biti uključen simulator Facebook Graph API-a. U tu svrhu je izgrađen dio *FacebookStubAPI* koji simulira dohvaćanja i postavlja ograničenja. Umjesto da se podaci dohvaćaju izravno od Facebooka, *Stub API* dohvaća podatke iz lokalne baze podataka što čitav proces simulacije ubrzava nekoliko redova veličine. Također, *Stub API* bilježi i kada je zahtjev poslan tako da je u mogućnosti simulirati ograničenje od 600 poziva u 10 minuta.

*Početno i završno vrijeme* su granice za koje želimo pokrenuti simulaciju, a zadaju se pri definiranju poslova podsustavu za pokretanje.

Simulacija se provodi tako da se oponaša rad sustava za dohvaćanje u nekom intervalu. Prije početka simulacija *Stub API*-u postavljamo početno vrijeme kao trenutno vrijeme sustava za simulaciju. Sustav počinje dohvaćati sadržaj stranica stvoren do tog trenutka (naime, *Stub API* pazi da vraća samo sadržaj koji je kreiran prije trenutnog vremena sustava za simulaciju) i u trenutku kada dosegne ograničenje, pomiče trenutno vrijeme. Zbog praktičnosti i brzine se kao novo trenutno vrijeme simulacije postavlja vrijeme u kojem bi bio prvi neodbijeni poziv na Graph API. Također zbog jednostavnosti, zanemaruje se trajanje poziva na API, tj. definira se da poziv traje beskonačno malo vremena. Naravno, u stvarnosti poziv traje neko vrijeme i za točnije rezultate bi možda bilo potrebno i ovaj parametar uzeti u obzir.
U nastavku se nalazi pseudokod općeg algoritma za simulaciju:

```
1  dohvati skup stranica za koje se simulacija izvodi;
2  vrijeme ← pocetno vrijeme;
3  while vrijeme < zavrsno vrijeme do
4  |  trenutna stranica ← dohvati iz reda cekanja;
5  |  objave ← dohvati nove objave(trenutna stranica);
6  |  if probijeno ogranicenje then
7  |  |  vrijeme ← najraniji trenutak u kojem je poziv
   |  |  dopusten;
8  |  |  nastavi;
9  |  end
10 |  foreach objava in objave do
11 |  |  kasnjenje ← vrijeme − objava.vrijemeObjavljivanja;
12 |  |  zabiljezi vrijeme kasnjenja(kasnjenje);
13 |  end
14 |  dodaj u red cekanja(trenutna stranica);
15 end
```

Glavna razlika između strategija dohvaćanja su implementacije reda čekanja. Strategije se mogu razlikovati i u još nekim pojedinostima koje će biti navedene u nastavku.

## 4.6. Buzzwise

*Buzzwise* strategija je vrlo jednostavna i minimalno nadograđuje opći pseudokod. Bitna nepoznanica s obzirom na pseudokod je implementacija reda čekanja. Kako je već navedeno u prijašnjim poglavljima, koristi se struktura podataka red (*engl. queue*) - struktura kod koje se podaci ubacuju na kraj, a izvlače se s početka strukture.

Također, bitno je napomenuti da je pretpostavljeno da ova tehnika u obzir uzima samo podskup skupa Facebook stranica.

## 4.7. Air balloon

*Air balloon* tehnika je značajno složenija od *buzzwise* tehnike i zahtijeva više intervencija u generičkom pseudokodu kako bi sama tehnika proradila. Prvo će se navesti potrebne intervencije, a nakon toga detaljnije opisati svaka od njih.

- Implementirati proces računanja akumulirane aktivnosti,
- implementirati red čekanja,

– implementirati proces hlađenja,

– implementirati proces zagrijavanja.

### 4.7.1. Računanje akumulirane aktivnosti

Računanje akumulirane aktivnosti je vrlo jednostavno s obzirom da je aktivnost svake stranice reprezentirana jednodimenzionalnim poljem s 24 broja dvostruke preciznosti. Čitava implementacija takvog algoritma nalazi se na kraju rada kao **Dodatak A**.

### 4.7.2. Implementacija reda čekanja

Umjesto vlastite implementacije, za ostvarenje reda čekanja iskorišten je *Javin Priority Queue* [2]. On svojim ponašanjem odgovara zahtjevu da se element s ekstremnom vrijednosti prvi izvlači iz reda. Iako implementacija vraća element s minimumom, a mi želimo uvijek izvlačiti element s maksimalnom akumuliranom vrijednosti, pomoću trika možemo iskoristiti ovu implementaciju za naše potrebe. Implementiran je *Javin komparator* [3] koji služi kao maska prioritetnom redu koja sve vrijednosti akumuliranih aktivnosti množi s $-1$. Tim postupkom maksimum preslikavamo u minimum, a minimum u maksimum.

### 4.7.3. Implementacija procesa zagrijavanja

Proces zagrijavanja je konceptualno opisan kao proces koji povećava vrijednost aktivnosti oko vremena kada je određeni sadržaj kreiran. Međutim, ta definicija nije skroz jasna i pretpostavlja se da aktivnost pamtimo kao kontinuirani, a ne diskretni signal. Ako aktivnosti pamtimo po satima, pitanje je gdje ćemo povećavati aktivnost ako je sadržaj kreiran u 18:30h (hoćemo li povećati aktivnost oko 18h ili 19h?). Praktično, povećat ćemo aktivnosti i oko 18h i oko 19h, ravnomjerno. Postupak koji određuje u kojem omjeru ćemo raspodijeliti novonastalu aktivnost po satima naziva se linearna interpolacija (zapravo radimo obrnut proces). Imamo početnu i završnu točku (puni sat u kojem je sadržaj kreiran i sat iza toga), a računamo vrijednost parametra koji bi interpolacijom dao vrijeme u kojem je sadržaj kreiran.

[2] `http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html`

[3] `http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html`

**ulazni podaci**: vrijemeObjave, vrijemeSimulacije

```
1 trenutnoVrijeme ← vrijemeSimulacije/(1000 * 60 * 60);
2 vrijemeObjave ← objava.dohvatiVrijeme()/(1000 * 60 * 60);
3 prethodniSat ← vrijemeObjave.satObjave();
4 minute ← vrijemeObjave.minuteObjave();
5 sljedeciSat ← (prethodniSat + 1)%24;
6 eksponent = e**[(vrijemeObjave-trenutnoVrijeme)/inercija ];
7 aktivnost [prethodniSat ] += eksponent *minute/60;
8 aktivnost [sljedeciSat ] += eksponent *(60-minute)/60;
```

### 4.7.4. Implementacija procesa hlađenja

Implementacija procesa hlađenja je jednostavnija od procesa zagrijavanja utoliko što ne moramo koristiti dodatne postupke kao što je linearna interpolacija. Hlađenje se odvija tako što se umanje sve vrijednosti aktivnosti po modelu koji smo opisali u prethodnom poglavlju. Konkretna implementacija se može pronaći na kraju rada kao **Dodatak B**.

# 5. Rezultati

Ovo poglavlje obrađuje rezultate dobivene primjenom opisanih tehnika na skupu podataka pripremljenom za potrebe ovog rada.

Prvi dio poglavlja donosi uvid u skup podataka nad kojim se tehnike primjenjuju i način na koji se došlo do tih podataka. U kasnijem dijelu poglavlja se donose rezultati primijenjenih tehnika pokrenutih različitim parametrima učenja. Sve tehnike se uspoređuju s referentnom tehnikom *buzzwise*. Donosi se i usporedba svih tehnika na jednom mjestu.

Na kraju poglavlja se donosi kratka diskusija o dodatnim tehnikama se može postići napredak i poboljšati rezultati.

## 5.1. Skup podataka

Za isprobavanje tehnika potrebno je imati neki skup podataka na kojem će se izvoditi simulacije. Podaci za potrebe ovog rada su prikupljeni pomoću sustava za prikupljanje podataka (opisan u poglavlju implementacije). Podaci se sastoje od popisa stranica, dodatnih informacija o njima te objava i samo nužnih podataka potrebnih za izvođenje.

Dio popisa stranica preuzet je iz baze podataka sustava *buzzwise*, a dio je prikupljen podsustavom za dohvat podataka. Ostatak popisa je prikupljen postupkom scrapinga (SCHRENK, 2007) web stranice koja se bavi skupljanjem podataka o najpopularnijim stranicama na društvenim mrežama - `http://www.fanpagelist.com/`.

Objave dotičnih stranica su prikupljene isključivo podsustavom za dohvat podataka. Podsustavu su zadani takvi parametri da je dohvatio sve objave popisanih stranica od 1. listopada 2012. godine do 27. ožujka 2013. godine (vrijeme pokretanja tog podsustava).

U konačnom skupu podataka nalazi se $4,106$ Facebook stranica čiji su stupnjevi popularnosti od najpopularnijih do srednje popularnih stranica.

U skupu objava se nalazi $1,106,496$ objava.

Cijeli skup podataka u spremištu podataka (MySQL baza) teži $450$ MB.

## 5.2. Referentna tehnika

U poglavlju vezanom za implementacije različitih tehnika je detaljno objašnjen i princip rada referentne tehnike te se zbog toga sada navode samo parametri i rezultati. Navedeni parametri su stvarni parametri koji vrijede u sustavu *buzzwise*.

Referentna tehnika u obzir uzima $3,000$ najpopularnijih stranica, što znači da je ostalih $1,106$ stranica iz skupa stranica u potpunosti zanemareno u postupku simulacije.

Broj stranica koje će dohvaćati ova tehnika se određuje u ovisnosti o željenom vremenu proteklom od stvaranja do dohvaćanja neke objave. Pokazalo se da čitav skup stranica distribuira objave po uniformnoj razdiobi pa stoga do broja od $3000$ stranica dolazimo iz zahtjeva da prosječno vrijeme od stvaranja do dohvata bude niže od $30$ minuta (osnove vjerojatnosti i statistike nam tvrde sljedeće - ako imamo $3600$ poziva u sekundi i u obzir uzmemo $3600$ stranica, očekivano vrijeme proteklo od stvaranja do dohvata iznosi 1800 sekundi - 30 minuta) (Ross, 2006).

Pokretanjem tehnike na cijelom skupu podataka postignuti su sljedeći rezultati: referentna tehnika *buzzwise* je dohvatila $\sim 810,000$ objava s prosječnim kašnjenjem (vremenom proteklim od kreiranja do dohvata) od $1,530$ sekundi (25 minuta i 30 sekundi).

Dobivenim prosječnim kašnjenjem možemo biti zadovoljni (tako smo i očekivali), ali svakako treba uzeti u obzir da je ova tehnika dohvatila samo dio sveukupnih podataka - $73\%$.

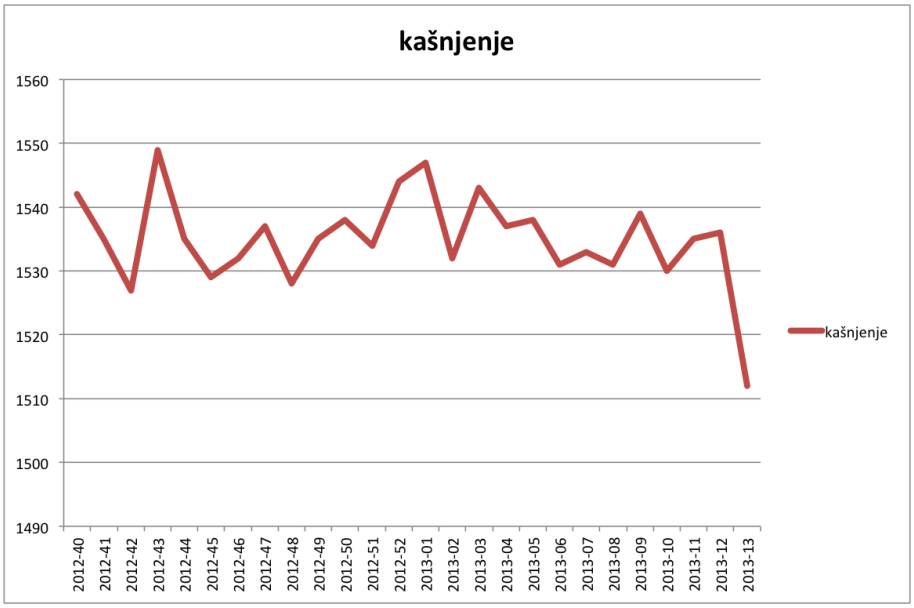
Zbog boljeg uvida u stanje u sustavu prosječne vrijednosti ćemo promatrati po tjednima.

Nadolazeća tehnika bi trebala biti u stanju dohvatiti cijeli skup podataka uz približno jednako kašnjenje.

## 5.3. Tehnika air balloon

Kako je predmet rada istraživanja korisnosti ove tehnike u dohvaćanju sadržaja s Facebook stranica, sam postupak simulacije je pokretan više puta s različitim parametrima. Ovakvo pokretanje bi nam trebalo pokazati koliko ova tehnika može biti dobra, ali isto tako i koliko je ona otporna (ili neotporna) na promjene navika Facebook stranica i promjene parametara pod kojima sustav funkcionira.

Parametri s kojima se pokretala ova tehnika navedeni su u 5.1.




**Slika 5.1:** Pregled prosječnog kašnjenja po tjednima tehnike buzzwise

Pogledom na tablicu 5.1 vidimo i dodatne prednosti ove tehnike - broj dohvaćenih objava.

**Tablica 5.1:** Parametri pri pokretanju tehnike *air balloon*

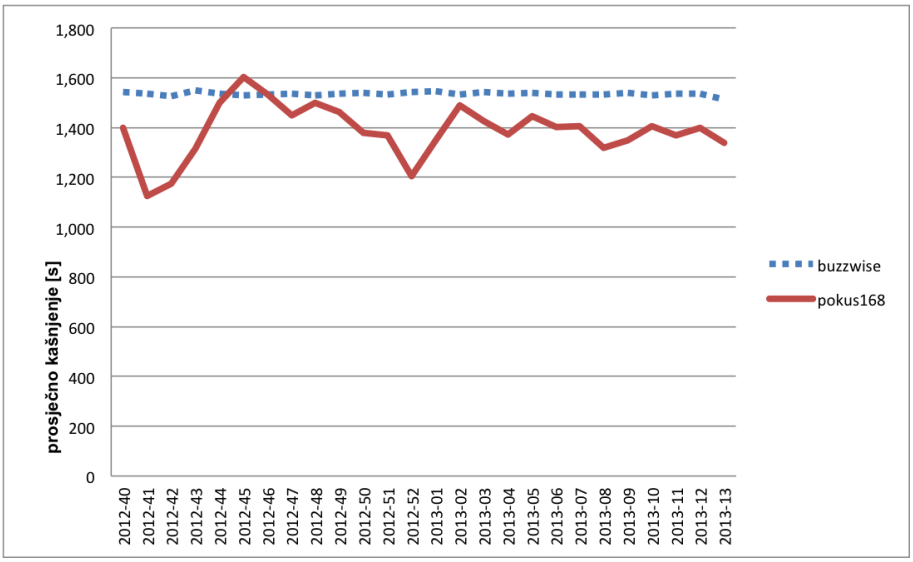
| parametar | vrijednost |
|---|---|
| veličina skupa stranica | $4,106$ |
| broj objava | $1,101,205$ |
| inercija | ovisna o pokusu |

Zbog lakšeg razumijevanja i usporedbe je svako pokretanje obrađeno u zasebnom odlomku i rezultati se uspoređuju zasebno.

### 5.3.1. Pokus168

Pokus168 je simulacija s parametrom inercije postavljenim na 168 sati (7 dana). U nastavku su priloženi grafovi i komentar.

Na slici 5.2 je vidljivo da *air balloon* tehnika u početku uspijeva dohvaćati s manjim kašnjenjem od *buzzwise* tehnike, ali nakon nekoliko tjedana prosječno vrijeme kašnjenja poraste iznad kašnjenja tehnike *buzzwise*. Očito je da je parametar inercije od 168 sati na neki način neprikladan za ovu skupinu izvora pa sustav loše reagira na promjene ponašanja izvora.




**Slika 5.2:** Pregled prosječnog kašnjenja po tjednima simulacije Pokus168

Iako *Pokus168* u nekom trenutku dohvaća sporije od tehnike *buzzwise*, treba primijetiti da nakon naglog rasta opet počinje učiti i ponovno dohvaćati brže od tehnike *buzzwise* te da dohvaća 37% više postova nego *buzzwise* tehnika.

U tablici 5.2 je pokazano i kako je srednje vrijeme kašnjenja na cijelom vremenskom intervalu $9.6\%$ manje.

**Tablica 5.2:** Usporedba rezultata buzzwise i pokus168 simulacije

| | Buzzwise | Pokus168 |
|---|---|---|
| broj dohvaćenih objava | $806,270$ | $1,100,382$ |
| prosječno kašnjenje (sekunde) | $1,535$ | $1,387$ |

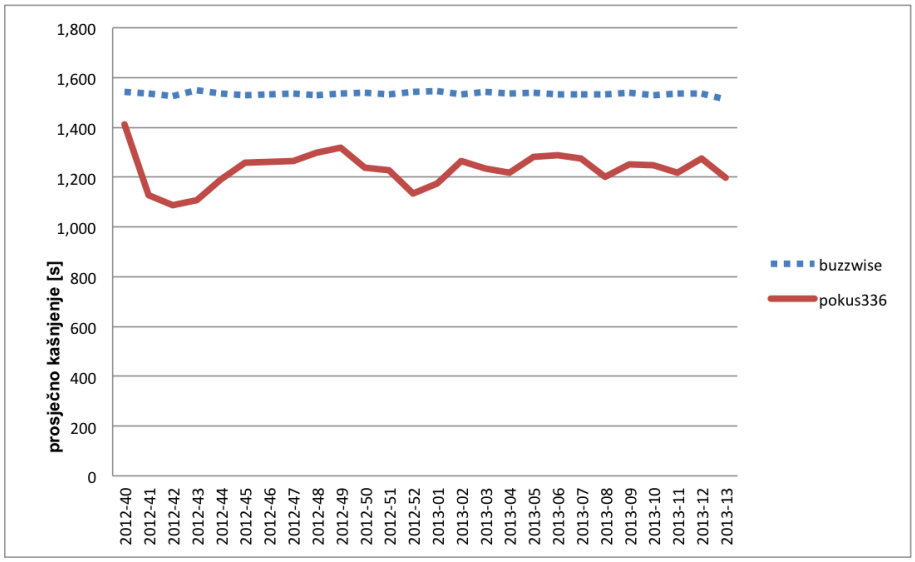
### 5.3.2. Pokus336

Pokus336 je simulacija s parametrom inercije postavljenim na 336 sati (14 dana).

Graf 5.3 nam pokazuje da je *air balloon* tehnika s parametrom inercije od 336 sati dala znatno bolje rezultate od *buzzwise* tehnike. U svakom tjednu je *air balloon* tehnika imala manje kašnjenje od *buzzwise* tehnike.

Pogledom na tablicu 5.3 u nastavku vidimo i dodatne prednosti ove tehnike - broj dohvaćenih objava.

*Air balloon* tehnika u ovoj simulaciji dohvaća $\sim 300,000$ više objava ($37\%$ više) od *buzzwise* tehnike, a prosječno kašnjenje je $19.7\%$ manje.




**Slika 5.3:** Pregled prosječnog kašnjenja po tjednima simulacije Pokus336

**Tablica 5.3:** Usporedba rezultata buzzwise i pokus336 simulacije

| | Buzzwise | Pokus336 |
|---|---|---|
| broj dohvaćenih objava | $806,270$ | $1,100,371$ |
| prosječno kašnjenje (sekunde) | $1,535$ | $1,232$ |

Kako vidimo da je povećanje inercije dalo bolje rezultate, pokrenuli smo i simulaciju s dvostruko većom inercijom.

### 5.3.3. Pokus672

*Pokus672* je oznaka simulacije *air balloon* tehnike s pripadajućim parametrom inercije od 672 sata (28 dana).

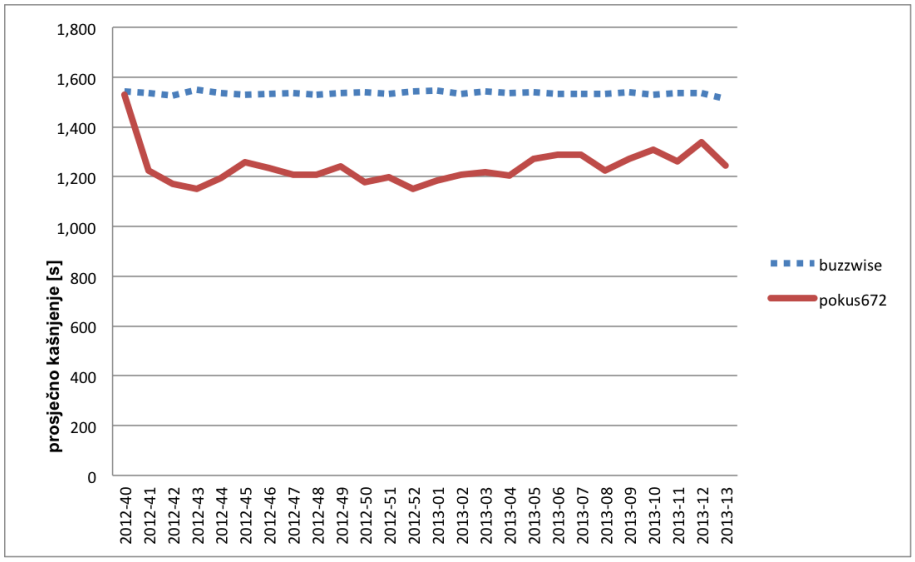
Iz grafa 5.4 je vidljivo da *air balloon* tehnika konstantno dominira nad *buzzwise* tehnikom, tj. u svakom tjednu daje bolje rezultate.

Generalni uvid o broju dohvaćenih objava i prosječnom kašnjenju prikazan je u tablici 5.4.

**Tablica 5.4:** Usporedba rezultata buzzwise i pokus672 simulacije

| | Buzzwise | Pokus672 |
|---|---|---|
| broj dohvaćenih objava | $806,270$ | $1,100,490$ |
| prosječno kašnjenje (sekunde) | $1,535$ | $1,241$ |

*Air balloon* tehnika s ovim parametrima je uspjela dohvatiti $\sim 300,000$ više objava




**Slika 5.4:** Pregled prosječnog kašnjenja po tjednima simulacije Pokus672

s prosječnim kašnjenjem koje je 294 sekundi manje (5 minuta i 54 sekunde). Relativno govoreći, *air balloon* tehnika je dohvatila $37\%$ više objava s $19\%$ manjim prosječnim kašnjenjem. Iako je vidljivo da je ova simulacija dala nešto lošije rezultate od simulacije *Pokus336*, prikazani su rezultati simulacije s još većom inercijom kako bi se uočilo kretanje prosječnog vremena s obzirom na postavljenu inerciju.

### 5.3.4. Pokus960

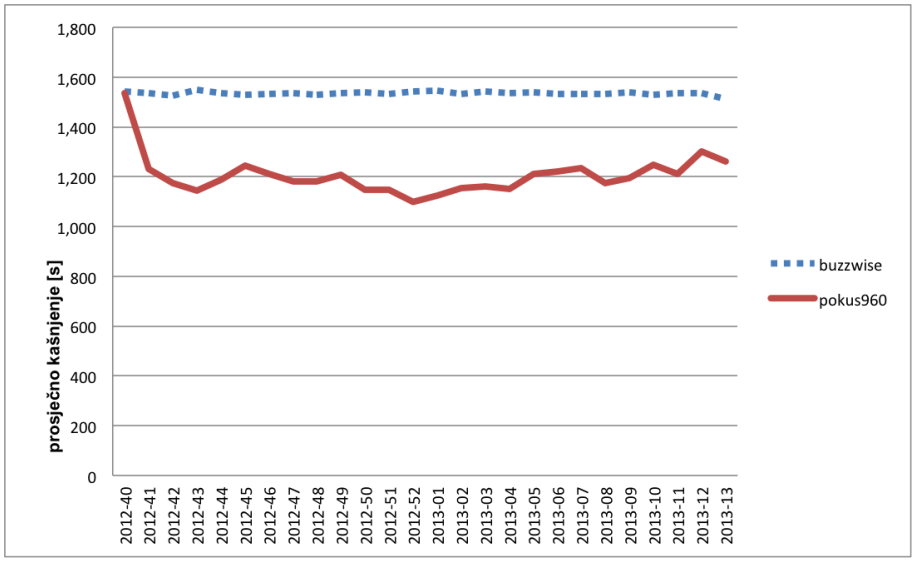
*Pokus960* je oznaka simulacije *air balloon* tehnike s pripadajućim parametrom inercije od 960 sati (40 dana).

Očekivano, i ova inačica tehnike *air balloon* znatno dominira na referentnom *buzzwise* tehnikom.

**Tablica 5.5:** Usporedba rezultata buzzwise i pokus960 simulacije

| | Buzzwise | Pokus960 |
|---|---|---|
| broj dohvaćenih objava | $806,270$ | $1,100,300$ |
| prosječno kašnjenje (sekunde) | $1,535$ | $1,205$ |

Vidljivo je da povećanje parametra inercije dodatno snižava prosječno kašnjenje. *Pokus960* dohvaća $37\%$ više objava (kao i *Pokus672*), ali još dodatno snižava prosječno kašnjenje. Prosječno kašnjenje je manje za 40 sekundi od *Pokus672*, što znači da se svaka objava otkrije prosječno 40 sekundi ranije, što je poželjno poboljšanje (iako




**Slika 5.5:** Pregled prosječnog kašnjenja po tjednima simulacije Pokus960

ne znatno). Tehnika *air balloon* pokrenuta s parametrom inercije od 960 sati daje $\sim 21.5\%$ manje prosječno kašnjenje od tehnike *buzzwise*.

### 5.3.5. Usporedba svih tehnika simulacija

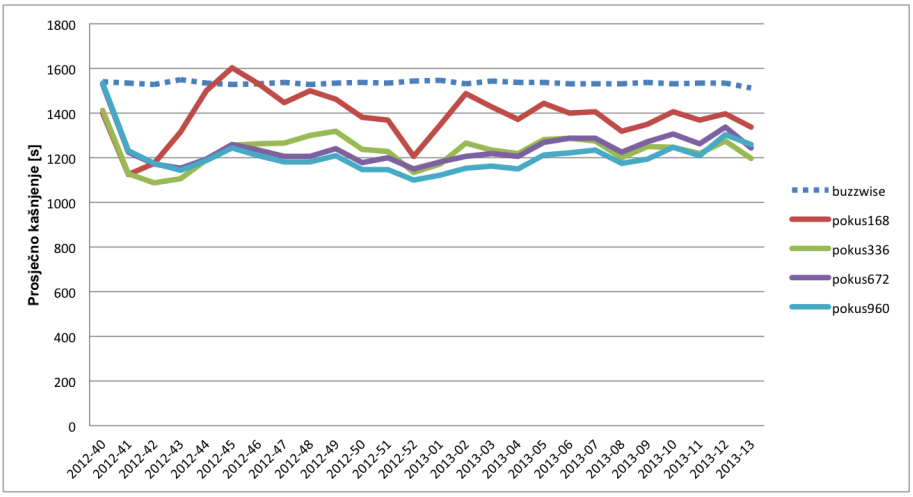
Radi bolje preglednosti donosi se tablični prikaz svih simulacija (tablica 5.6) te grafičke usporedbe prosječnih vremena kašnjenja (slika 5.6) i dohvaćenih objava po tjednima (slika 5.7). Dobiveni rezultati se neće dodatno komentirati pošto su komentari dani u obradi svake simulacije zasebno.

**Tablica 5.6:** Usporedba rezultata svih simulacija

| simulacija | broj dohvaćenih objava | prosječno kašnjenje |
|---|---|---|
| buzzwise | $806,270$ | $1,535$ |
| Pokus168 | $1,100,382$ | $1,387$ |
| Pokus336 | $1,100,371$ | $1,232$ |
| Pokus672 | $1,100,490$ | $1,240$ |
| Pokus960 | $1,100,400$ | $1,205$ |

### 5.3.6. Računanje optimalne vrijednosti parametra inercije

Inercija je jedini parametar koji je potrebno ručno zadati tehnici *air balloon*. Nažalost, iz priloženih rezultata je vidljivo kako parametar inercije značajno utječe na rezultate i




**Slika 5.6:** Pregled prosječnog kašnjenja svih simulacija


**Slika 5.7:** Pregled dohvaćenih postova buzzwise i air balloon metode

zbog toga bi bilo dobro naći približno optimalnu vrijednost ovog parametra.

U računanju odgovarajuće vrijednosti ovog parametra će nam pomoći algoritam *zlatnog reza* (engl. Golden section search) [1], inačica algoritma ternarnog pretraživanja koja će nam omogućiti brže pretraživanje prostora parametra inercije tako što određene vrijednosti iskorištava u više koraka i time smanjuje potreban broj simulacija (simuliranje na cijelom vremenskom intervalu traje preko 30 minuta). Važno je napomenuti

[1] više o samom algoritmu se može doznati na `http://en.wikipedia.org/wiki/Golden_section_search`

kako je pretpostavljeno da je srednje vrijeme kašnjenja unimodalna funkcija na prostoru kojeg pretražujemo, tj. da postoji samo jedan lokalni ekstrem - minimum.

Da bismo dodatno ubrzali traženje odgovarajuće vrijednosti, u izračun je uključen svaki treći izvor, ali je također i ograničenje poziva proporcionalno manje - 200 poziva u 10 minuta.

U tablici 5.7 je prikazano nekoliko koraka navedenog algoritma. Vrijednosti $a$, $b$, $c$ i $d$ su vrijednosti parametara inercije za koje radimo evaluaciju. $a$ i $b$ su rubovi intervala, a $c$ i $d$ točke unutar intervala. $f(x)$ označava prosječno kašnjenje za inerciju $x$.

**Tablica 5.7:** koraci *algoritma zlatnog reza* algoritma u potrazi za optimalnom vrijednosti inercije

| a | b | c | d | f(a) | f(b) | f(c) | f(d) |
|---|---|---|---|---|---|---|---|
| 24 | 3,600 | 1,390 | 2,234 | 2,041 | 1,189 | 1,192 | 1,189 |
| 1,390 | 3,600 | 2,234 | 2,756 | 1,208 | 1,189 | 1,192 | 1,189 |
| 2,234 | 3,600 | 2,756 | 3,072 | 1,192 | 1,189 | 1,189 | 1,191 |

Provođenjem tri koraka navedenog algoritma ocrtava se oblik funkcije prosječnog kašnjenja ovisnog o vrijednosti parametra inercije. Zaključujemo da je potrebno odabrati dovoljno veliku vrijednosti parametra inercije kako bi se prosječno kašnjenje spustilo na približno optimalnu razinu. Potvrda da dodatno povećanje parametra inercije neće dodatno smanjiti vrijeme kašnjenja su simulacije pokrenute s parametrima inercije od 5,000h i 10,000h. Oba pokretanja su dala rezultate kašnjenja od 1,189 sekundi. Do ovakvog zaključka možemo doći i ako analiziramo model zagrijavanja i hlađenja. Kako je glavni dio funkcije $e^{\frac{-vrijeme}{inercija}}$, povećanje inercije eksponent približava nuli, što cijeli izraz približava broju $1$. Navedeni izraz se zbog svojstva eksponencijalne funkcije ne može evaluirati u neki broj $< 1$.

### 5.3.7. Dodatna poboljšanja

Nakon znatnih poboljšanja koja su postignuta implementacijom *air balloon* tehnike, postavlja se pitanje mogu li se postići još bolji rezultati. Naravno, vrlo je vjerojatno da ovo nije donja teoretska granica za ovaj problem te da se dodatnim igranjem parametrima mogu postići poboljšanja.

Postoji vrlo jednostavno i očigledno poboljšanje za koje nam ne trebaju inteligentne tehnike specifične za računarsku znanost. Rješenje je upogonjavanje više aplikacija

koje će paralelno dohvaćati sadržaj. U trenutku pisanja (svibanj 2013.) nije pronađen niti jedan dio Facebookovih pravila koji bi ovu tehniku okarakterizirao zabranjenom.

Također, postoje i neka inteligentnija poboljšanja.

Kako se ovaj model bilježenja aktivnosti izvora (Facebook stranice) pokazao dobrim, moguće je na isti model primijeniti određene heuristike koje će prepoznati karakteristične situacije i pomoći tehnici da brže nauči određene promjene (npr. nagli prekid ili početak aktivnosti).

Dobro je primijetiti i da tehnika ne razlikuje različite dane te da bi se postigla dodatna poboljšanja ako bi razlikovali vikende i radne dane.

# 6. Zaključak

Problem dohvaćanja velike količine sadržaja s Facebook stranica uz dana ograničenja svakodnevni je problem mnogih sustava. Analizom problema i njegovim pojednostavljivanjem dolazi se do zaključka kako je problem koji rad obrađuje analogan problemu obnove baze sadržaja primitivne web tražilice (danas izvori dojavljuju tražilici da je sadržaj promijenjen). Rad uvodi tehniku zračnog balona (*air balloon* tehnika) koja bilježi aktivnosti određene stranice po satima unutar jednog dana i uči na temelju onoga što se već dogodilo postupcima zagrijavanja i hlađenja aktivnosti. Implementacijom i simulacijama pokazano je kako ova tehnika na prikupljenom skupu podataka znatno dominira nad običnom tehnikom jednolikog dohvaćanja (u radu pod imenom *buzzwise*).

# LITERATURA


Matko Bošnjak, Eduardo Oliveira, Jose Martins, Eduarda Mendes Rodrigues, i Luis Sarmento. Twitterecho - a distributed focused crawler to support open research with twitter data. *WWW 2012 – MSND'12 Workshop*, stranice 1233–1239, 2012.

Facebook. *Facebook API*, 2013. URL `http://developers.facebook.com/docs/reference/api/`.

Lawrence S. Husch. *Exponential Functions*, 2001. URL `http://archives.math.utk.edu/visual.calculus/0/exp_log.5/`.

S.M. Ross. *Introduction to Probability Models*. Elsevier Science, 2006. ISBN 9780123756879. URL `http://books.google.hr/books?id=0yDAZf1TfJEC`.

M. SCHRENK. *Webbots, Spiders, And Screen Scrapers: A Guide to Developing Internet Agents With Php/Curl*. No Starch Press Series. NO STARCH PRESS, 2007. ISBN 9781593271206. URL `http://books.google.hr/books?id=QVDCXxagCjwC`.

SpiegelOnline. *'Twitter Revolution': Fearing Uprising, Russia Backs Moldova's Communists*, 2009. URL `http://www.spiegel.de/international/europe/twitter-revolution-fearing-uprising-russia-backs-moldova-s-communis` `html`.

Washington Times. *EDITORIAL: Iran's Twitter revolution*, 2009. URL `http://www.washingtontimes.com/news/2009/jun/16/irans-twitter-revolution/`.

Zeynek Tufekci. *Tunisia, Twitter, Aristotle, Social Media and Final and Efficient Causes*, 2011. URL `http://technosociology.org/?p=263`.

Wikipedia. *Facebook History*, 2011. URL http://en.wikipedia.org/wiki/History_of_Facebook.

Wikipedia. *Six degrees of separation' theory tested on Facebook*, 2013a. URL http://www.telegraph.co.uk/technology/facebook/8704891/Six-degrees-of-separation-theory-tested-on-Facebook.html.

Wikipedia. *Usenet*, 2013b. URL https://en.wikipedia.org/wiki/Usenet.

J.L. Wolf, M.S. Squillante, J. Sethuraman, i L. Ozsen. Optimal crawling strategies for web search engines. *WWW '02 Proceedings of the 11th international conference on World Wide Web*, stranice 136–147, 2002. URL http://dl.acm.org/citation.cfm?id=511465.

Andrija Čajić. Scheduling methods for distributed twitter crawling. Magistarski rad, FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO, June 2012.

# Dodatak A
# Računanje akumulirane aktivnosti

```
1  private void calculateAccumulatedActivity() {
2
3    ExtendedTimestamp targetTime = getCurrentTime();
4    ExtendedTimestamp pointerTime = lastAccumulation.copy();
5
6    double diff =
7      targetTime.getAbsoluteHours()-pointerTime.getAbsoluteHours();
8
9    if (pointerTime.getHours() == targetTime.getHours()) {
10     // starts and ends in same hour
11     int nextHour = (pointerTime.getHours()+1)%24;
12     int hour = pointerTime.getHours();
13     double slope = activity[nextHour]-activity[hour];
14     double startH = activity[hour]
15         + slope*pointerTime.getMinutes()/60.;
16     double endH = activity[hour]
17         + slope*(60-targetTime.getMinutes())/60.;
18     double h = (startH+endH)/2;
19     accumulatedActivity += h*diff;
20   } else {
21
22     while (pointerTime.compareTo(targetTime) < 0) {
23
24       int hour = pointerTime.getHours();
25       int nextHour = (hour+1)%24;
26
27       if (pointerTime.getMinutes() != 0) {    // first part
28
29         double slope = activity[nextHour]-activity[hour];
30         double dx = (60-pointerTime.getMinutes())/60.;
31         double hStart = activity[hour]
```

```
32                  +(1-dx)*slope;
33          accumulatedActivity += dx*(hStart + activity[nextHour])
                /2;
34
35          pointerTime.fillToNextHour();
36          diff -= dx;
37
38        } else if (diff < 1) {            // last part
39
40          double slope = activity[nextHour]-activity[hour];
41          double dx = targetTime.getMinutes()/60.;
42          double hEnd = activity[hour] + dx*slope;
43          accumulatedActivity += dx*(activity[hour] + hEnd)/2;
44
45          pointerTime.fillToNextHour();
46          diff -= dx;
47
48        } else {                  // all other parts
49
50          accumulatedActivity += (activity[nextHour]+activity[hour
                ])/2;
51          pointerTime.addHours(1);
52          diff -= 1;
53
54        }
55
56      }
57    }
58
59    lastAccumulation = new ExtendedTimestamp(AirBalloon.this.
          getCurrentTime());
60  }
```

# Dodatak B
# Implementacija hlađenja

```
1 public void coolDown(Timestamp currentTime) {
2   double lastCoolingHours = lastCooling.getTime()/(1000.*60*60),
3     currHours = currentTime.getTime()/(1000.*60*60),
4     expo = Math.exp((lastCoolingHours-currHours)/inertia);
5
6   for (int i = 0; i < activity.length; ++i)
7     activity[i] = Math.max(minActivity, activity[i]*expo);
8
9   lastCooling = new ExtendedTimestamp(currentTime);
10
11  calculateTotalActivity();
12  notifyCooling();
13 }
```

**Povećanje učinkovitosti dohvaćanja sadržaja putem interneta**

## Sažetak

Rad se bavi razvojem i evaluiranjem tehnika za učinkovito dohvaćanje sadržaja s Facebook stranica.

U početku se ukratko objasni problem i naznače moguće koristi i primjene evoluiranih tehnika. Nakon toga se ukratko opisuje Facebook API (točnije, Facebook Graph API) - API čije ograničenje od 600 poziva u 10 minuta i stvara potrebu za razvijanjem ovakvih tehnika. U 3. poglavlju se detaljnije opisuje problem, predstavlja *buzziwse* tehnika (tehnika implementirana na sustavu `http://www.buzzwise.com`) i njena ograničenja. Uvođenjem određenih zakonitosti postupno se evoluira *air balloon* tehnika inspirirana upravljanjem zračnim balonom. Implementacijom i evaluacijom rezultata se dolazi do zaključka kako je evoluirana tehnika učinkovita za dani problem i kako u svakom pogledu dominira nad *buzzwise* tehnikom ($37\%$ više sadržaja uz $20\%$ manje kašnjenje). Na kraju poglavlja o rezultatima donosi se diskusija o rezultatima i postignućima rada te se navode dodatne ideje koje bi pomaknule granicu i u kombinaciji s tehnikom *air balloon* dale još bolje rezultate.

**Ključne riječi:** facebook, facebook stranice, strojno učenje, buzzwise, crawling, social crawling, dohvaćanje sadržaja, Graph API, Graph API zaobilaženje ograničenja

**Increasing the efficiency of content retrieval via the Internet**

## Abstract

This thesis is based on developing and evaluating techniques for efficient content retrieval from Facebook pages.

The beggining consists of the brief explanation of the problem and indicates possible uses and applications of these evalueted techniques. Furthermore, there is the description of Facebook API (more precisely, Facebook Graph API) – API whose restriction of 600 calls in 10 minutes is the one that causes the need to develop such techniques. Third chapter describes the problem in detail, introduces buzzwise technique (technique implemented in the system `http://www.buzzwise.com`) and its limitations. By introducing certain principles, air baloon technique evolves, inspired by air baloon management. Implementation and evaluation of results reaches the conclusion that this evolved technique is useful for given problem and it dominates over buzzwise technique (37% more content with 20% less delay). The last part of results contains discussion of the results and achievements of this thesis, as well as further ideas for pushing the boundaries and, combined with air baloon technique, for providing even better results.

**Keywords:** facebook, facebook pages, machine learning, buzzwise, crawling, social crawling, scraping, Graph API, breaking Graph API limits