Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miha Hrastel

Obvladovanje realnih povezav med ljudmi v poslovnem svetu

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: doc. dr. Dejan Lavbič

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01958/2013
Datum: 02.09.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MIHA HRASTEL**

Naslov: **OBVLADOVANJE REALNIH POVEZAV MED LJUDMI V POSLOVNEM SVETU**

MANAGING REAL-WORLD CONNECTIONS BETWEEN PEOPLE IN BUSINESS ENVIRONMENT

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V zadnjem času je vedno bolj priljubljena analiza povezav na družabnih omrežjih. Bolj pomembno od virtualnih povezav na družabnih omrežjih pa je obvladovanje dejanskih povezav med ljudmi v realnem življenju. V podjetju so takšne povezave izjemno pomembne, saj pogosto pride na podlagi omenjenih povezav do sklepanja novih poslov. Predstavljajte si sposobnost vodilnega kadra v podjetju, da pred vstopom v pogajanja o novem poslu pridobi vse povezave svojih zaposlenih s podjetjem, s katerim bi želelo skleniti posel in te povezave potem uporabi v prid sklepanja posla. V okviru diplomske naloge bi bilo potrebno pripraviti prototip rešitve, ki na podlagi elektronskih sporočil zaposlenih v podjetju poskuša prikazati bolj realno sliko družabnih povezav sodelavcev v podjetju med sabo in predvsem s poslovnimi partnerji. Pri implementaciji razvijte metrike, ki opredelijo moč povezave med zaposlenim v podjetjem in osebo iz drugega podjetja.

Mentor:

doc. dr. Dejan Lavbič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Miha Hrastel, z vpisno številko **63080027**, sem avtor diplomskega dela z naslovom:

Obvladovanje realnih povezav med ljudmi v poslovnem svetu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 01. septembra 2013 Podpis avtorja:

Diplomsko delo je odraz dosedanjega obdobja nabiranja znanj in zaključek štiriletnega študijskega obdobja. Zato se zahvaljujem vsem, ki so mi pri doseganju tega cilja pomagali.

Zahvaljujem se mentorju, doc. dr. Dejanu Lavbiču, ki me je vodil s svojim strokovnim znanjem, izkušnjami in koristnimi nasveti.

Zaposlenim v podjetju Agilcon, za pomoč ter razumevanje v času izdelave diplomskega dela.

Očetu Iztoku, mami Damjani in bratu Anžetu za vzpodbudo, motivacijo in energijo.

Ter Doroteji, da svoje jeze ob večerih, ki jih zaradi pisanja diplomskega dela nisva preživela skupaj, ni znesla nad računalnikom, na katerem je bilo delo shranjeno.

Kazalo

Seznam uporabljenih kratic in simbolov

CSV	(ang. Comma Separated Value) Datoteka, kjer so vrednosti ločene z vejico
JSON	(ang. JavaScript Object Notation) Tekstovni odprtokodni standard zasnovan za izmenjavo podatkov.
mbox	Generičen izraz za datoteke, ki vsebujejo zbirke e-poštnih sporočil
RFC	(ang. Request for comments) Zahteva za komentar
IMAP	(ang. Internet Message Access Protocol) Protokol za dostop do e-poštnih podatkov.
Gmail	(ang. Google Mail) Googlova e-poštna storitev
MIT	Massachusetts Institute of Technology
JavaScript	Objektni skriptni programski jezik namenjen ustvarjanju interaktivnih spletnih strani
Java	Objektno usmerjeni programski jezik, razvit v podjetju Sun Microsystems
Play	Ogrodje za izdelavo spletnih aplikacij v Javi in Scali
Scala	Objektno-funkcijski skriptni programski jezik namenjen razvoju aplikacij.

MVC	(ang. Model-View-Controller) Popularen arhitekturni pristop uporabljen pri razvoju aplikacij
REST	(ang. Rpresentational state transfer) Tip programske arhitekture namenjen uporabi v distribuiranih sistemih (World Wide Web)
IDE	(ang. Integrated Development Environment) Napredna in celovita programska oprema namenjena programerjem pri razvoju aplikacij.
JVM	(ang. Java Virtual Machine) Virtualni sistem za prevajanje in zagon Javanske kode.
PaaS	(ang. Platform as a service) Poseben model storitve v oblaku.
Ruby	Dinamičen objektni programski jezik
HTML	(ang. Hyper Text Markup Language) Označevalni jezik za izdelavo spletnih strani.
HTML5	HTML verzije 5.
CSS	(ang. Cascading Style Sheets) Slogovni jezik, ki ga uporabimo za vizualni semantični opis nekega dokumenta. Najpogosteje se uporablja skupaj z dokumenti HTML.
CSS3	CSS verzije 3.
SVG	(ang. Scalable Vector Graphics) Označevalni jezik za opis vektorskih grafik.
PDF	(ang. Portable Document Format) Format datoteke, ki se uporablja za prikaz dokumentov. Prikaz je neodvisen od programske ali strojne opreme računalnika, zaradi česar je standard postal zelo popularen za uporabo.

Povzetek

Z bliskovitim razvojem socialnih omrežij, postajajo medosebni stiki med ljudmi vedno manj pomembni. Navidezni prijatelji, s katerimi se socializiramo preko spleta nam dajejo zavajajoč občutek pripadnosti in prijateljstva, ki v realnosti pogosto ne obstaja. V poslovnem svetu, pa se stanje kljub prihodu novih tehnologij ne spreminja tako hitro. Za večino poslovnih dogovorov je še vedno potrebno poznanstvo in osebni stik. Številna podjetja se zavedajo pozitivne vrednosti dodatnih informacij o osebi ali podjetju, s katerim želijo skleniti posel. Zato se poslužujejo uporabe programov za obdelavo spletnih socialnih omrežij, ki jim pomagajo pri njihovem iskanju in obdelavi.

V diplomski nalogi se ukvarjamo s pridobivanjem in analizo takšnih informacij. Razvili smo prototip aplikacije, ki skuša na podlagi analize e-poštnega omrežja zaposlenih v podjetju ugotoviti, kdo izmed zaposlenih v podjetju tudi v realnosti pozna nekoga iz drugega podjetja ter kakšna je njuna moč povezave. Tako lahko direktor podjetja pred sklepanjem posla od zaposlenega pridobi pomembne informacije o stiku ter z njimi uspešneje vodi in zaključi posel. Za lažjo predstavo je prototip dostopen na spletnem naslovu `http://cocoonproject.herokuapp.com/`.

Ključne besede

spletna aplikacija, analiza e-poštnega omrežja, moč povezave, realna poznanstva med osebami

Abstract

Real, personal connections between people are becoming less and less important in everyday world, due to the rapid development of online social networks. People are used to socialize online. It gives them a sense of affiliation and friendship, which doesn't necessarily exist in their lives. Luckily, things in business world move and change slower. Business deals are still discussed in a traditional manner, over a business lunch or a cup of coffee. That suggests, that real human connections are more important in business world than the virtual ones. Many companies try to gather as much information as possible about their prospects. They are aware of the benefits that additional information will bring, when they will try to close the deal. Therefore, they use sophisticated tools to analyze data and gather such information.

Diploma thesis focuses on finding and visualizing such information, in a way that is easy to understand to end-users. We develop a prototype web application, which analyses a company's email system and tries to find information about "who knows whom" in real world. It also determines the strength of the connections, which gives a CEO or employee better understanding of how strong the relationship is and can it be used to find new prospects, close a deal, etc. The prototype is accessible on http://cocoonproject.herokuapp.com/.

Keywords

web application, email network analysis, connection strength, "who knows whom"

Poglavje 1

Uvod

V današnjem digitalnem svetu postaja realnost vedno bolj zapostavljena. Ljudje vedno več časa preživimo na socialnih omrežjih, kjer si gradimo virtualno mrežo "prijateljev", ki jih v realnosti pogosto ne poznamo. Pa vendar so povezave v resničnem življenju tiste, ki v poslovnem svetu odpirajo nove možnosti in poslovnežem omogočajo lažje in bolj uspešno sklepanje novih poslov.

LinkedIn je že nekaj časa sinonim za spletno povezovanje ljudi na podlagi njihove strokovne usposobljenosti, izobrazbe, znanj in veščin. S svojo ogromno mrežo uporabnikov (več kot 225 milijonov [1]), velja za največje spletno poslovno omrežje na svetu. Vodilna vloga mu omogoča uveljavljanje pravil in načinov, s katerimi se v sodobnem poslovnem svetu povezujemo. V kombinaciji z ostalimi velikimi globalnimi socialnimi omrežji, kot sta Twitter in Facebook, pa narekuje tempo in novodobni stil življenja ljudi in poslovnežev.

A kljub poplavi spletnih socialnih storitev so v 21. stoletju osebni stiki med poslovneži ohranili svojo pomembnost. Pogovori o številnih poslih še vedno tečejo v okviru poslovnih kosil ter se v večini primerov uspešno zaključijo s podpisom pogodbe na enem od medosebnih srečanj. Tudi uporaba e-poštnih sporočil, kot glavnega medija za njihovo medsebojno komunikacijo, priča dejstvu, da v poslovnem svetu stvari sledijo utečenim smernicam, ki so

[1] Podatki veljajo za junij 2013. Povzeto po http://en.wikipedia.org/wiki/LinkedIn

se že v preteklosti izkazale kot uspešne.

Omenjena socialna omrežja nudijo množico podatkov o načinu življenja njihovih osnovnih gradnikov — ljudi. Zato je dandanes njihova analiza popularno področje raziskovanj. Številni programi skušajo na podlagi analize podatkov priti do ugotovitev in sklepov o obnašanju in relacijah med posameznimi gradniki omrežja. Med priljubljene teme analiz sodijo iskanje skupnosti v omrežju, iskanje zakonitosti na podlagi atributov oseb, preučevanje povezav med posameznimi osebami ipd. Na spletu obstajajo številne rešitve, ki ponujajo analizo velikih socialnih omrežij. A kljub ogromni množici informacij, ki jih takšna omrežja vsebujejo in širokemu spektru uporabe, imajo vsa socialna omrežja skupen problem. Zaradi želje in težnje ljudi po socializaciji in pripadnosti k skupnosti, relacija povezave med dvema osebama izgubi svoj pomen. Tako ne moremo več enostavno določiti, katere povezave med osebami obstajajo tudi v njihovem *realnem življenju* ter kdo v omrežju je povezan le *po naključju*. Kot že rečeno, pa je ravno *realna* povezava med dvema osebama tista, ki največ šteje v globalnem poslovnem omrežju. Odkrivanje takšnih povezav je mogoče z uporabo kompleksnih analitičnih orodij, a zahteva podrobno semantično analizo komunikacije med osebama. Zaradi problemov s pridobivanjem podatkov in njihovo analizo takšna rešitve ni optimalna.

Socialna omrežja, kot so LinkedIn, Facebook in Twitter niso edina omrežja, ki obstajajo v poslovnem svetu. Pomembna, manjša in bolj zaprta komunikacijska omrežja predstavljajo e-poštni pogovori med zaposlenimi znotraj podjetja ali z njihovimi poslovnimi partnerji. Zaradi privatnosti in pogoste poslovne uporabe ima komunikacija z e-poštnimi sporočili mnogo bolj *"osebен značaj"* kot komunikacija preko socialnih omrežij. Poslano ali prejeto e-poštno sporočilo kaže na profesionalen in resen pristop k rešitvi ali problemu, ki ga naslavlja. Hkrati, je vsebina e-poštnega sporočila last njegovega pisca, kar podzavestno vpliva na njegovo pomembnost. Vsi našteti razlogi utemeljujejo prepričanje, da lahko povezave, ki nastanejo med pošiljateljem in prejemnikom e-poštnega sporočila, označimo kot *realne*.

Čeprav so nekatere spremembe v poslovnem svetu počasnejše, še ne po-

meni, da je uporaba spletnih omrežij v namene *iskanja novih in izboljšanja starih poslovnih priložnosti* brez pomena. Ravno nasprotno. Številna podjetja se zavedajo njihove vrednosti, zato vlagajo čas in denar v spletne oglaševalske storitve, programe za analitično obdelavo podatkov in podobno. Predstavljajte si možnost, da lahko direktor ali zaposleni pred pomembnim srečanjem ali sestankom pridobi informacije o osebi ali o podjetju, s katerim želi skleniti posel. Tako bo lahko hitreje in uspešneje navezal stik s potencialno bodočo stranko ali poslovnim partnerjem, kar se bo odražalo v uspešnosti in poslovni rasti podjetja. Nekatere informacije o stiku lahko pridobi iz spletnih omrežij, a kot že rečeno, več pomenijo realna poznanstva med ljudmi. Obstaja namreč možnost, da s stikom, s katerim se želimo povezati, prijateljuje eden od zaposlenih v podjetju. Mogoče je skupaj z njim nekdo od zaposlenih obiskoval srednjo šolo ali pa je eden od oddelkov v domačem podjetju z njim že sodeloval v preteklosti.

Če privzamemo, da so povezave v e-poštnih komunikacijskih omrežjih realne in upoštevamo dejstvo, da je takšne podatke dosti lažje pridobiti, lahko odgovore na zgornje domneve poiščemo na podlagi *analize vseh poslanih in prejetih e-poštnih sporočil zaposlenih v nekem podjetju*.

V diplomski nalogi bomo zato razvili *prototip aplikacije*, s katero bodo zaposleni *lažje obvladovali povezave*, ki se pojavljajo znotraj omrežja vseh e-poštnih pogovorov med zaposlenimi v podjetju ter njihovimi poslovnimi partnerji. Aplikacija bo zaposlenemu omogočala enostaven pregled vseh oseb, ki se pojavljajo v omrežju in njihovih medsebojnih relacij. Z interaktivno vizualno predstavitvijo omrežja bo njena uporaba intuitivna in preprosta, a kljub temu dovolj informativna. Najpomembneje pa je, da bo lahko zaposleni skozi raziskovanje in njeno uporabo odgovoril na štiri osnovna vprašanja:

- Ali je domače podjetje z neko osebo ali podjetjem *v preteklosti že komuniciralo*?
- Kdo od zaposlenih v domačem podjetju *pozna osebo iz podjetja*, s katero želimo navezati stik in kakšna je *moč njune povezave*?

- S katerimi stiki podjetje *trenutno največ komunicira — so za podjetje najbolj aktualni*?
- Kdo izmed zaposlenih je *najpomembnejši vezni člen* pri komunikaciji podjetja z njegovimi poslovnimi partnerji?

V digitalni dobi, v kateri živimo, pogosto težko razločujemo med navideznimi in realnimi povezavami med osebami. Z razvojem prototipa aplikacije, poskušamo ohraniti in prikazati pomembnost realnih povezav v omrežju poslovnega sveta.

Nadaljnja struktura diplomske naloge je sledeča. Poglavje 2 vsebuje pregled in opis tehnologij, ki smo jih uporabili v razvoju prototipa. Sledi poglavje 3, v katerem raziščemo prednosti in slabosti že obstoječih rešitev, jih primerjamo ter predstavimo predlog rešitve, ki jo bomo implementirali v obliki spletne aplikacije. V poglavju 4 za lažje razumevanje problema najprej predstavimo osnovne pojme in definicije, ki jih uporabimo v nadaljevanju, nato pa opišemo konceptualni razvoj predlagane rešitve. V poglavju 5 prikažemo končen rezultat implementacije prototipa spletne aplikacije. Nalogo zaključimo s poglavjem 6, v katerem povzamemo ugotovitve, sklepe in probleme, na katere smo naleteli med razvojem. Za konec predstavimo še različne možnosti za nadaljnje raziskovanje takšnih omrežij.

Poglavje 2

Opis uporabljene tehnologije

Tehnologije, opisane v poglavju, smo uporabili pri razvoju prototipa spletne aplikacije.

Že iz samega tipa aplikacije je razvidno, da smo pri razvoju potrebovali tehnologije, ki se uporabljajo pri razvoju *spletnih storitev* — storitev, ki za pravilno delovanje potrebujejo spletni brskalnik ter delujočo povezavo z internetom. Takšne aplikacije so v osnovi zgrajene iz skriptnih programskih jezikov, kot so HTML, CSS ter JavaScript. V kombinaciji z različnimi knjižnicami zadoščajo za implementacijo številnih spletnih storitev. Njihova slabost je, da se izvajajo na odjemalcu (spletni brskalnik). Kljub konstantnim izboljšavam in razvoju brskalnikov, le ti še vedno niso namenjeni izvajanju računsko zahtevne in kompleksne programske kode, saj lahko zelo upočasni njihovo delovanje. Zato, smo logiko aplikacije implementirali v objektnem programskem jeziku Java. Problem pri uporabi programskega jezika Java v spletnih aplikacija je, da za izvajanje potrebuje JVM, ki ne deluje na spletnem brskalniku. Na srečo, obstajajo številna ogrodja, ki z različnimi pristopi poenostavijo in omogočajo uporabo Jave, pri razvoju spletnih storitev. Eno izmed novejših ogrodij je ogrodje *Play* [8]. Zaradi njegove enostavnosti in fleksibilnosti smo ga uporabili pri razvoju prototipa.

Podoben pristop, z uporabo ogrodja, smo izbrali tudi pri implementaciji vizualne podobe prototipa, kjer smo si pomagali s spletnim ogrodjem *Bo-*

otstrap [11]. Slednji, z veliko zbirko komponent, ki so zgrajene s pomočjo spletnih tehnologij HTML, CSS in JavaScript, programerjem olajša razvoj vizualne podobe spletne aplikacije.

Dobra stran programskega jezika Java je možnost uporabe številnih dodatnih knjižnic, ki programerjem pomagajo in poenostavijo implementacijo različnih programov. V diplomski nalogi raziskujemo omrežja, zato smo na spletu iskali prosto dostopne knjižnice, ki so specializirane za analizo in interpretacijo različnih tipov omrežij. Ker je področje v zadnjem času osrednja tema raziskovanj, obstaja v programskem jeziku Java nekaj dobrih implementacij. Med seboj se ločijo po tipu omrežij, ki jih podpirajo in metodah, ki jih vsebujejo za njihovo analizo. Ena od bolj popularnih in hkrati zelo kompetentnih je knjižnica *Gephi* [4]. V razvoju prototipa smo jo uporabili pri gradnji ter statistični analizi omrežja.

Čeprav Gephi omogoča vizualizacijo omrežja, smo si zaradi želje po interaktivni predstavitvi omrežja, raje pomagali s spletno knjižnico *D3.js* [2]. Z uporabo knjižnice Gephi lahko prikažemo le statično sliko omrežja v formatu `pdf`, kar za uporabo v spletni aplikaciji ni optimalno, saj bi za vsako predstavitev dela omrežja potrebovali svojo sliko. Tako bi potrebovali ogromno prostora že za shranjevanje slik. Knjižnica D3.js nam omogoča vizualizacijo posameznega dela omrežja direktno na spletnem brskalniku, saj jo poganja jezik JavaScript. S tem izkoristimo vire, ki nam jih ponuja spletni brskalnik, hkrati pa lahko omrežje predstavimo na uporabniku prijazen in interaktiven način. Ker vizualiziramo samo dele omrežja, z izvajanjem kode ne upočasnjujemo brskalnika, tako da lahko le ta normalno deluje.

Delovanje prototipa smo želeli preizkusiti in simulirati še na produkcijskem okolju, zato smo po koncu razvoja spletno aplikacijo prenesli na platformo *Heroku* [6]. Potek prenosa bomo natančneje opisali v podpoglavju 5.3.

V nadaljevanju bomo vsako od omenjenih tehnologij podrobneje predstavili.

2.1 Play Framework

Play je odprtokodno ogrodje za izdelavo spletnih aplikacij, ki omogoča razvoj aplikacij v jezikih Java in Scala. Po svoji arhitekturni zasnovi sledi znanemu MVC (ang. model-view-controller) modelu. Z uporabo ang. "convention over configuration" pristopa, ki stremi k preprostemu a fleksibilnemu načrtovanju aplikacij, skuša optimizirati produktivnost programerjev in pohitriti njihov razvoj. Čeprav omogoča razvoj s pomočjo programskega jezika Java, je v konceptih bolj podoben ogrodjem, kot so "Ruby On Rails" ali "Django". Temu najbolj priča dejstvo, da uporablja REST pristope za prenos podatkov (ang. requests and responses) med strežnikom in odjemalcem. Prav tako v ozadju sam poskrbi za prevajanje kode (ang. compile), ki jo avtomatsko naloži v JVM, brez potrebe, da bi JVM ponovno zagnali.

Glavna prednost Play ogrodja je uporabnost. Za njegovo uporabo in pravilno delovanje ne potrebujemo zapletenih IDE orodij, inštalacije dodatnih javanskih knjižnic ipd. Dovolj je, da ogrodje prenesemo s spleta, odpremo program za urejanje tekstovnih datotek in že smo pripravljeni na razvoj aplikacij. Prav tako nam z uporabo naprednih mehanizmov olajša prevajanje kode, razhroščevanje in detekcijo napak. Ogrodje podpira uporabo izraznih programskih jezikov (ang. expression language), s katerimi poenostavi pisanje programske kode, potrebne za prenos podatkov med podatkovnim modelom in spletno aplikacijo.

Kljub temu da gre za mlado ogrodje, katerega razvoj se je šele dobro začel (prva uradna verzija je sicer izšla leta 2007), Play navdušuje razvijalce po svetu z drugačnim, a vseeno uporabnim in resnim pristopom k razvoju aplikacij v Javi. Temu priča tudi dejstvo, da je podporo zanj leta 2011 naznanil znan ponudnik storitev v oblaku Heroku.

2.2 Gephi

Gephi je odprtokodni program za analizo in vizualizacijo omrežij. V svojih raziskavah ga uporabljajo številne znanstvene in gospodarske ustanove, za-

radi enostavne uporabe pa je popularen tudi med splošnimi uporabniki. Za Windows, Linux in Mac OS X je dostopen v obliki aplikacije, naprednim uporabnikom pa enake funkcionalnosti kot v samostojni aplikaciji omogoča z uporabo ang. "Gephi-toolkit API" Java knjižnice.

Aplikacija uporablja posebni 3D pogon, s katerim lahko omrežja vizualizira v realnem času. To uporabniku omogoča interaktivno uporabniško izkušnjo, s katero lažje in hitreje analizira velika omrežja podatkov. S posameznimi moduli lahko uvozimo, vizualiziramo, filtriramo, manipuliramo in izvozimo vse vrste omrežij. Zaradi dobro zasnovane arhitekture, je aplikacija sposobna v realnem času obdelati in prikazati omrežja z več deset ali sto tisoč vozlišči.

Njegova največja prednost je preprosta razširljivost z uporabo vtičnikov (ang. plugin). To pomeni, da lahko vsakdo razvije nove, hitrejše in boljše algoritme ali metode za obdelavo in vizualizacijo. Na spletu obstaja enotni prostor za objavo prostodostopnih dodatkov. Z uporabo nekaterih lahko analiziramo tudi omrežja z več milijon vozlišči.

2.3 Heroku

Heroku je PaaS (ang. Platform as a service) storitev, ki ponuja platformo za razvoj spletnih aplikacij v oblaku. Platforma omogoča uporabniku razvoj in objavo spletnih aplikacij brez stroškov vzpostavitve strojne in programske opreme, potrebne za delovanje aplikacije. Razvijalcem tako omogoča, da ves svoj čas posvetijo razvijanju aplikacije, medtem ko Heroku skrbi za njeno pravilno delovanje. Poleg osnovne arhitekture uporabnikom ponuja številne dodatne storitve, kot so spletna analitika, uporabo podatkovnih baz, napreden nadzor nad uporabo aplikacije itd. V razvoju je od leta 2007, ko je podpiral le programski jezik Ruby, sedaj pa podpira še številne druge. Zaradi njegovega globalnega uspeha ga je leta 2010 kupil spletni gigant Salesforce.

V osnovi platforma razvijalcem omogoča objavo, upravljanje, zagon ter delovanje aplikacij, napisanih v programskih jezikih Ruby, Java, Node.js,

Scala, Python ali Clojure. Za pravilno delovanje morajo aplikacije vsebovati *programsko kodo* in *datoteko z opisom odvisnosti* (ang. dependency), ki so potrebne za njeno pravilno izvajanje. Zgradba in ime datoteke se razlikujeta glede na programski jezik: V Ruby se imenuje `Gemfile`, v Pythonu `requirements.txt`, v Javi `pom.xml`. Na podlagi datoteke, Heroku sam zazna, kje se nahajajo metode, ki so potrebne za zagon aplikacije. Heroku, za zagon običajno potrebuje še posebno datoteko imenovano `Procfile`. Uporabi jo pri zagonu, kadar želimo ali moramo eksplicitno definirati ukaze, s katerimi poženemo aplikacijo.

Aplikacijo lahko na Heroku prenesemo s pomočjo sistema *Git* [5]. Git je popularen, distribuirani sistem za upravljanje in nadzor programske kode, ki ga zaradi njegove hitrosti in robustnosti uporabljajo številni programerji.

Heroku aplikacije požene s pomočjo posebnega vmesnika, ki ga imenujejo *Dyno*. Za lažje razumevanje jih lahko opišemo kot vire z računsko močjo (procesor in pomnilnik), ki vsebujejo vse potrebno za zagon in pravilno delovanje neke aplikacije. Obstajata dva tipa Dynov: *web*, ki omogoča delovanje spletnih aplikacij in *worker*, ki se uporablja za izvajanje procesov, ki tečejo v ozadju aplikacije (za delovanje ne potrebujejo internetne povezave). Aplikacija za pravilno delovanje potrebuje vsaj 1 web Dyno. Večje aplikacije jih lahko uporabljajo tudi več, odvisno od računske moči in dodatnih storitev. Uporaba enega Dynota je zastonj, morebitne ostale pa je potrebno plačati, po veljavnem ceniku platforme Heroku.

Popularen je postal predvsem zaradi enostavne vključitve dodatnih storitev. Brez posebnih komplikacij in truda lahko namreč povečamo velikost podatkovne baze, računsko moč ter morebitne dodatne aplikacije, ki jih uporabljamo za lažji nadzor in pregled delovanja.

Več o platformi in njeni uporabi bomo opisali v podpoglavju 5.3.

2.4 Spletne knjižnice in ogrodja

2.4.1 D3.js

D3.js je JavaScript knjižnica za vizualno predstavitev podatkov, ki s pomočjo tehnologij HTML, SVG, JavaScript in CSS omogoča ustvarjanje dinamičnih in interaktivnih grafičnih oblik, ki tečejo na strani odjemalca (spletni brskalnik). Z razvojem knjižnice so na stanfordski univerzi leta 2011 začeli trije doktorski študenti.

Knjižnica omogoča manipulacijo HTML dokumenta s pomočjo vnaprej definiranih JavaScript funkcij. Te omogočajo izbiranje HTML elementov, kreiranje SVG objektov, definicijo izgleda, uporabo interaktivnih efektov ipd. Z uporabo CSS lahko posamezne HTML elemente enostavno vizualno uredimo. Knjižnica je zaradi svoje hitrosti in učinkovitosti sposobna obdelati veliko množico podatkov, ki so lahko shranjeni v različnih formatih. Med najbolj pogoste formate, zaradi svoje preprostosti in uporabnosti, sodita JSON in CSV format. Uporaba JavaScript programskega jezika ji omogoča interaktivno obogatitev in manipulacijo vizualnih vsebin.

Svojo popularnost je dosegla zaradi širokega spektra uporabnosti, saj omogoča izdelavo najrazličnejših vrst grafov, diagramov, podatkovnih tokov itd. Njena največja prednost je preprosta vključitev v že obstoječe ali nove spletne vsebine.

2.4.2 Twitter Bootstrap

Bootstrap je ogrodje, ki vsebuje prosto dostopno zbirko orodij za izdelavo spletnih strani in aplikacij. Temelji na skriptnih programskih jezikih HTML5 in CSS3. Kolekcija orodij vsebuje predloge za obliko črk, form, gumbov, navigacijskih elementov in ostalih pomembnih gradnikov spletnih strani. Razvila sta jo Mark Otto in Jacob Thornton iz podjetja Twitter v okviru internega projekta, s katerim so želeli razviti konsistentno ogrodje za uporabo skupnih oblikovalskih standardov znotraj podjetja.

Popularen je postal leta 2011, ko je postal odprtokoden. Njegova prednost je hiter in učinkovit razvoj aplikacij, saj ponuja skoraj vse komponente, ki se danes uporabljajo pri razvoju spletnih rešitev, hkrati pa je združljiv in uporaben v vseh večjih spletnih brskalnikih.

Od verzije 2.0 podpira ang. "responsive design", ki omogoča samostojno, dinamično prilagoditev in uporabo spletne aplikacije na mobilnih napravah. Poleg HTML in CSS gradnikov vključuje tudi JavaScript gradnike, ki skrbijo za interaktivno uporabniško izkušnjo z uporabo efektov, drsnikov, vključitvijo video in avdio elementov v spletne vsebine ipd.

Poglavje 3

Sorodna dela in predlog rešitve

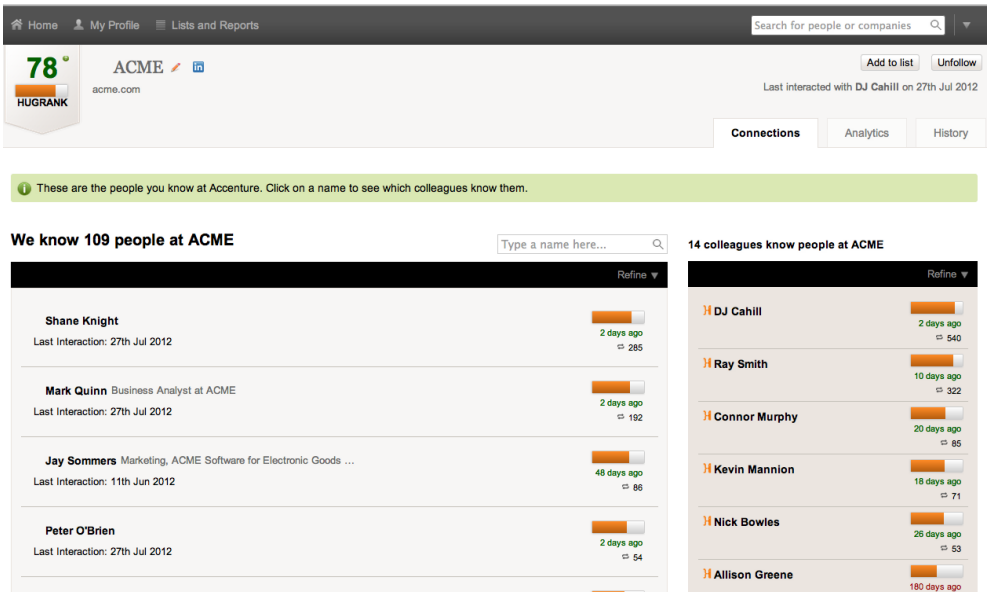
3.1 Pregled sorodnih del

Na spletu so se v zadnjem letu pojavile nekatere zanimive rešitve, ki poskušajo na podlagi podatkov, pridobljenih iz e-poštnih sporočil uporabnika, prikazati in raziskati omrežja. Dve izmed njih, *DataHug* [3] in *Immersion* [7], bomo predstavili v nadaljevanju.

3.1.1 DataHug

DataHug je inovativno irsko start up podjetje, ustanovljeno leta 2010. Razvili so istoimensko spletno aplikacijo, ki preko analize stikov v e-poštnih sporočilih podjetja zgradi dinamično bazo povezav med njimi. Pri tem poskuša poiskati čim več informacij o odkritih povezavah ter tako izračunati vrednost povezave, ki jo sami imenujejo *HugRank*. Obstajata dve verziji aplikacije — zastonjska in plačljiva.

DataHug pridobi nabor podatkov z avtomatskim pregledovanjem in indeksiranjem e-poštnega računa podjetja. Trenutno omogoča povezavo z Gmail in Microsoft e-poštnim računom, preko IMAP protokola pa se lahko poveže tudi z ostalimi e-poštnimi strežniki. Napredni algoritmi so sposobni sami, na podlagi e-poštnih naslovov, določiti, h kateremu podjetju pripada kontakt in osebe kategorizirati. Poleg e-poštnih sporočil preišče tudi koledarske vnose



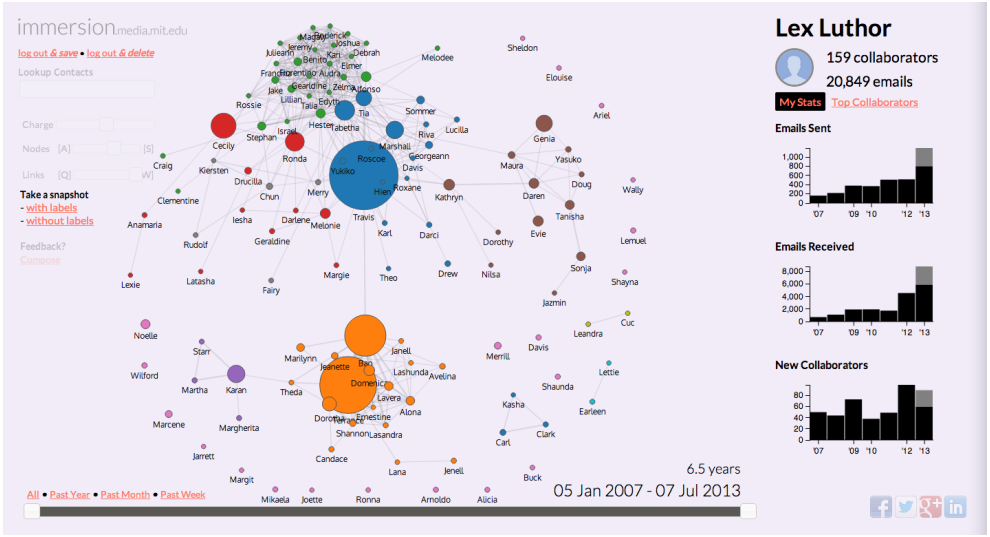
Slika 3.1: Primer zaslonske maske v spletni aplikaciji DataHug

in telefonske klice (Google Talk storitev), ki dodatno vplivajo na vrednost povezave.

Aplikacija omogoča enostavno uporabo in iskanje po analiziranih kontaktih in podjetjih. Na domači strani aplikacije so prikazani podatki o stikih, ki jim "sledimo" (so za podjetje trenutno zelo pomembni), podatki o novih stikih ter tistih, s katerimi že nekaj časa nismo bili v interakciji. Za vsak stik, poleg mere HugRank, izračuna še številne statistične informacije, kot so na primer informacije o številu poslanih ali prejetih e-poštnih sporočil, "zdravje" povezave med osebama, spreminjanje mere HugRank skozi čas, podatki o zadnji interakciji itd. Primer prikaza podatkov o podjetju "ACME" lahko vidimo na sliki 3.1.

3.1.2 Immersion

Trojica doktorskih študentov na priznani ameriški univerzi MIT je pred kratkim predstavila spletno aplikacijo, ki omogoča vizualni vpogled v e-poštno "življenje" posameznika. Na podlagi e-poštnih sporočil uporabnika sestavi



Slika 3.2: Primer zaslonske maske v spletni aplikaciji Immersion

sliko omrežja vseh stikov, s katerimi je uporabnik komuniciral. Vsako povezavo v omrežju opremi z dodatnimi statističnimi informacijami, kot so prva in zadnja pojavitev v e-poštnih sporočilih, prikaz oseb, s katerimi izbrana oseba najbolj sodeluje ipd.

Z algoritmom za detekcijo skupnosti posamezne stike razdeli v skupine, ki jih z različno barvo, vidno ločeno od ostalega omrežja, intuitivno vizualizira. S takšno vizualno sliko si uporabnik lažje predstavlja relacije med stiki in skupnosti, ki se pojavljajo znotraj njegovega e-poštnega omrežja. Uporabnik lahko razvoj omrežja opazuje tudi v določenem času ter tako podrobneje razišče razvoj skupnosti in relacij v omrežju. Slika 3.2 prikazuje zaslonski posnetek domače strani spletne aplikacije.

Aplikacija trenutno omogoča povezavo z Gmail, Microsoft Exchange in Yahoo spletnimi servisi, na voljo pa je tudi “demo” prikaz na izmišljenem naboru podatkov.

3.2 Primerjava in predlog rešitve

DataHug in Immersion poskušata vsak s svojim pristopom uporabniku približati in poenostaviti razumevanje komunikacijskega omrežja, ki se oblikuje iz prejetih in poslanih e-poštnih sporočil v njegovem e-poštnem nabiralniku. DataHug daje prednost bolj "resnim" analitičnim metodam in tabelaričnemu pristopu pri prikazu informacij, medtem ko skuša Immersion podobne rezultate prikazati z intuitivno vizualizacijo omrežja in interaktivno spletno aplikacijo.

Kljub številnim dobrim lastnostim in dobremu prikazu informacij smo pri obeh aplikacijah odkrili naslednje slabosti:

1. Obe aplikaciji sta pri svoji analizi osredotočeni na prikaz podatkov s stališča enega uporabnika storitve. To pomeni, da v primeru podjetja *ni možen pregled nad povezavami vseh zaposlenih* v podjetju in stikov, s katerimi komunicirajo (poslovnih partnerjev podjetja).

2. DataHug s tabelaričnim prikazom informacij *ne omogoča vizualnega vpogleda na celotno e-poštno omrežje* podjetja ali uporabnika.

3. Immersion, zaradi interaktivnega pristopa, ne vsebuje dobrih analitičnih podatkov, hkrati so vse povezave enako "močne", s čimer *ni moč razlikovati med pomembnostjo stikov* v e-poštnem omrežju uporabnika.

V diplomski nalogi bomo izdelali prototip aplikacije, ki združuje dobre lastnosti obeh pristopov, hkrati pa omogoča intuitiven, vizualen in širok pregled nad *celotnim e-poštnim omrežjem nekega podjetja*. Prototip mora omogočati interaktivno uporabo aplikacije in enostaven ter hiter dostop do informacij.

V naslednjem poglavju bomo opisali konceptualni razvoj prototipa, v katerem smo analizirali in podrobneje raziskali e-poštno omrežje, ki smo ga ustvarili iz izbranega nabora podatkov. Z izdelavo in uporabo prototipa želimo odgovoriti na štiri osnovna vprašanja, ki smo jih omenili v uvodu diplomskega dela.

Poglavje 4

Konceptualni razvoj rešitve

V poglavju bomo predstavili potek konceptualnega razvoja rešitve, ki smo jo predlagali v podpoglavju 3.2. Najprej bomo v podpoglavju 4.1 za lažje razumevanje podali osnovne definicije in pojme, ki jih uporabimo v razvoju. Pripravo podatkov in definicijo omrežja zaposlenih bomo predstavili v podpoglavju 4.2. Sledi podpoglavje 4.3 o preliminarni analizi, s katero smo poskušali poiskati optimalne algoritme ter mere za rešitev ključnih vprašanj, na katera poskušamo odgovoriti z razvojem prototipa. V podpoglavju 4.4 bomo razložili uporabo raziskanih mer in algoritmov na omrežju zaposlenih ter zaključili z opisom načinov in problemov pri vizualizaciji omrežja.

4.1 Osnovni pojmi in definicije

Omrežja predstavljajo izjemno močno ogrodje za analizo in vizualizacijo kompleksnih sistemov v realnem ali navideznem svetu, saj omogočajo enostavno predstavitev velike količine podatkov v obliki, zelo naravni človeškemu umu in dojemanju sveta. Zaradi širokega spektra uporabe jih znanstveniki za lažje razumevanje klasificirajo v kategorije, ki se med sabo ločijo v različnih lastnostih omrežij.

4.1.1 Realna omrežja

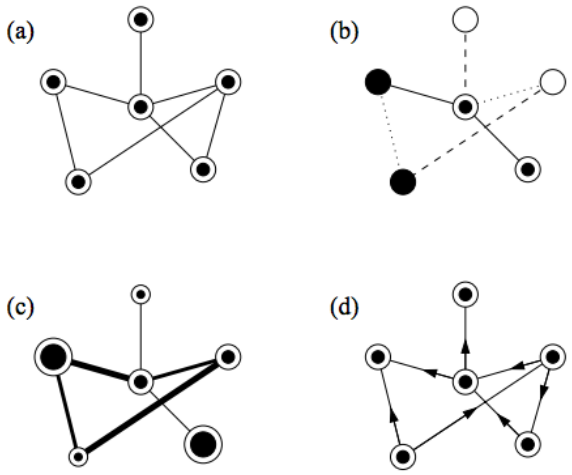
Sisteme, sestavljene iz velikega števila med seboj povezanih komponent, pogosto ponazorimo s pomočjo omrežij. Primeri nekaterih preučevanih omrežij so svetovni splet (World Wide Web), omrežja prijateljstev, organizacijska omrežja, nevronske mreže, omrežja citiranj med znanstvenimi prispevki, komunikacijska omrežja itd. V zadnjih letih se je področje preučevanja premaknilo iz analize posameznih, majhnih delov omrežij na preučevanje in iskanje statističnih zakonitosti v celotnem omrežju. Nove pristope je omogočil predvsem razvoj računalniške opreme, ki nam omogoča obravnavo velikih naborov podatkov v realnem času [26].

Preprosta omrežja so sestavljena iz enega tipa vozlišč in povezav (podrobnejša razlaga v poglavju 4.1.3). Poznamo pa tudi bolj kompleksna omrežja, kjer imajo vozlišča in povezave številne lastnosti. Če kot primer raziščemo socialno omrežje ljudi, ugotovimo, da lahko vozlišča predstavljajo moške ali ženske, različnih starosti, narodnosti ipd. Povezave med njimi v večini primerov ponazarjajo poznanstva med osebami na osebni ali profesionalni ravni. Lahko so utežene ali neutežene, s čimer določajo, kako dobro se dve osebi poznata ter usmerjene ali neusmerjene, kar v realnosti prikazuje, ali je poznanstvo samo enostransko ali obojestransko [26]. Primere različnih tipov omrežji vidimo na sliki 4.1.

Realna omrežja lahko v grobem razdelimo v štiri kategorije: socialna, informacijska, biološka in tehnološka [24].

- V *socialnih omrežjih* množica vozlišč običajno predstavlja ljudi (ali skupine ljudi), ki se ločijo po spolu, starosti, državljanstvu itd. Povezave med njimi so predstavljene kot interakcije v obliki prijateljstva, poslovnih odnosov, sodelovanja ipd.

- Klasičen primer *informacijskega omrežja* je omrežje citiranj med akademskimi prispevki. Ime izhaja iz strukture in delovanja omrežja. Informacije so shranjene v vozliščih in se prenašajo preko povezav, kar ustreza toku informacij v informacijskem sistemu. V to kategorijo so-



Slika 4.1: Primeri različnih tipov omrežij: (a) neusmerjeno omrežje sestavljeno iz vozlišč in povezav enega tipa; (b) diskretno omrežje; (c) omrežje z različnimi utežmi na vozliščih in povezavah; (d) usmerjeno omrežje. [26]

dijo komunikacijska omrežja, čeprav so po svoji naravi (povezave med osebami) večkrat kategorizirana tudi kot socialna.

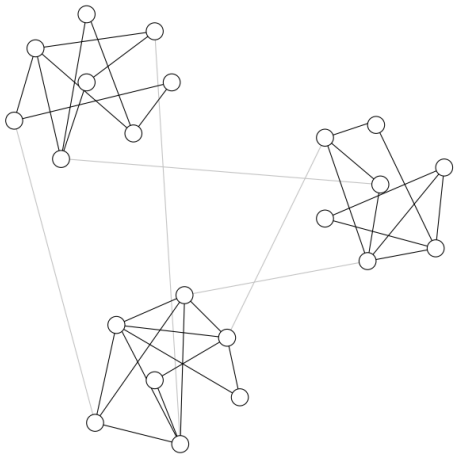
- *Biološka omrežja* ponazarjajo metabolične reakcije med celicami, geni, proteini, sem spadajo omrežja nevronskih mrež ipd. Z biološkim omrežjem lahko ponazorimo tudi prehrambeno verigo.

- Umetno ustvarjena *tehnološka omrežja* so običajno narejena za distribucijo neke storitve ali medija, na primer elektrike ali interneta. V to kategorijo spada internetno omrežje, električno in cestno omrežje ipd.

4.1.2 Lastnosti omrežij

Kljub svoji raznolikosti pa imajo realna omrežja številne skupne lastnosti. V nadaljevanju bomo izpostavili najbolj zanimive in za naš problem bistvene lastnosti realnih omrežij [32].

V 60. letih je Stanley Milgram izvedel poskus, s katerim je dokazal obstoj ti. *fenomena majhnega sveta* [22]. Posameznikom je naročil, naj s poslanim



Slika 4.2: Shematični prikaz omrežja s tremi skupnostmi (gosto povezana vozlišča), z manjšo gostoto povezav med njimi [18].

pismom preko več oseb, dosežejo končnega naslovnika. Izkazalo se je, da so poslana pisma končni cilj dosegla v zelo majhnem številu korakov — okoli 6 v zgoraj navedenem prispevku (ang. *six degrees of separation*). Fenomen dokazuje, da so pari vozlišč v realnih omrežjih med seboj povezani preko zelo kratkih poti. Kot zanimivost naj omenim, da je povprečna razdalja uporabnikov Facebook omrežja v letu 2011 znašala le $l = 4.7$.

Barabasi et al. v članku [17] opisujejo obstoj ang. *scale-free* omrežij. To so realna omrežja, kjer porazdelitev stopnje vozlišč navadno sledi potenčnemu zakonu $P(k) \propto k^{-\gamma}$, $\gamma > 1$ (ang. power law). V enačbi spremenljivka k predstavlja povprečno stopnjo vozlišča, ki je definirana kot število povezav, ki imajo en konec v obravnavanem vozlišču. Ena od ključnih lastnosti takšnih omrežij je pojav velikega števila vozlišč z majhno stopnjo in majhnega števila vozlišč z izjemno veliko stopnjo (ang. hubs). V socialnih omrežjih so to osebe z velikim številom prijateljev (k). Takšne osebe so ključne pri pretoku informacij skozi omrežje, saj omogočajo hitro razširjanje informacije po omrežju.

Za naš primer najbolj pomembna lastnost realnih omrežij je obstoj *komun ali skupnosti* [18] (ang. communities). Za lažje razumevanje bomo njihov obstoj razložili na primeru socialnega omrežja — omrežja prijateljstva ali

drugačnega poznanstva med posamezniki. Splošno znano je, da se v takšnih omrežjih pojavljajo skupnosti: *večje ali manjše podmnožice vozlišč, v katerih je gostota povezav med vozlišči velika, podmnožice med sabo pa so šibko povezane* (slika 4.2). V socialnih omrežjih lahko predstavljajo dejanske skupine ljudi s podobnimi interesi ali znanji; v omrežju citiranj ali v spletnih omrežjih lahko ponazarjajo članke oziroma strani s podobno vsebino itd. Zmožnost odkrivanja takšnih skupnosti ima praktično uporabnost in nam pomaga pri razumevanju in analizi realnih omrežij. Poleg komun pa zadnje študije realnih omrežij kažejo na obstoj drugih značilnih skupin vozlišč ali strukturnih modulov, imenovanih *funkcijski moduli [31]* (ang. functional modules). Ti navadno vežejo komponente nekega sistema, ki imajo podobno funkcijo ali vlogo.

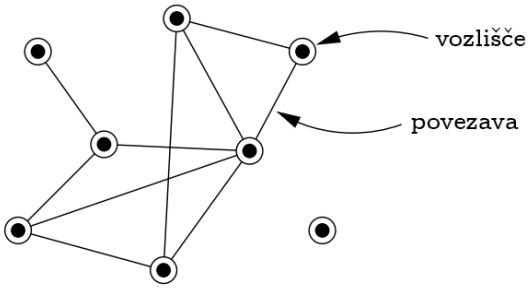
Več o gručenju, njegovi detekciji in uporabnosti, bomo predstavili v poglavju 4.3.2.

4.1.3 Metrike in mere

Če poznamo strukturo omrežja, ki ga raziskujemo, lahko z uporabo metrik in mer podrobneje analiziramo in preučimo omrežje ali njegove posamezne dele. V tem poglavju bomo podali osnovne definicije in pojme, ki se uporabljajo pri razlagi strukture omrežja. Kasneje jih v poglavjih 4.3 in 4.4 uporabimo za odkrivanje statističnih zakonitosti.

Omrežje je v svoji najpreprostejši obliki množica med seboj povezanih vozlišč [32]. Navadno je predstavljeno z diskretnim matematičnim objektom, ki mu pravimo graf. Graf $G(V, E)$ je sestavljen iz množice vozlišč $V = \{v_1, v_2, \ldots, v_n\}$, $n = |V|$ ter množice povezav med njimi $E = \{\{v_1, v_2\} \mid v_i, v_j \in V,\ m = |E|$. Primer preprostega omrežja vidimo na sliki 4.3.

Vozlišče (ang. vertex) je osnovni gradnik omrežja, ki ga najlažje ponazorimo z osebo — v primeru socialnega omrežja, molekulo — biološko omrežje ipd. Vozlišča lahko imajo številne atribute in lastnosti, ki dodatno opisujejo njihovo strukturo.



Slika 4.3: Preprosto omrežje, sestavljeno iz vozlišč istega tipa in neusmerjenih povezav [26].

Povezava (ang. edge) je črta, ki povezuje dve vozlišči med sabo. V primeru socialnega omrežja predstavlja relacijo prijateljstva med dvema osebama (vozliščema). Povezave so lahko:

- *Usmerjene / neusmerjene*: Usmerjene povezave tečejo v eno smer (npr. enosmerna cesta) in so običajno predstavljene s puščicami, ki nakazujejo njihovo usmerjenost. Graf je usmerjen, če so vse povezave v grafu usmerjene. Neusmerjene so običajno predstavljene s črto, čeprav lahko neusmerjen graf ponazorimo tudi z vozlišči, kjer sta med vsakim parom vozlišč dve povezavi, vsaka v svojo smer.

- *Utežene / neutežene*: Utež na povezavi določa raven moči povezave. V socialnih omrežjih bi utež predstavljala na primer moč poznanstva med dvema osebama.

Stopnja vozlišča (ang. degree) je število povezav, ki imajo en konec v opazovanem vozlišču. Stopnja ni nujno enaka številu sosednjih vozlišč (tistih, ki so povezani z vozliščem), saj lahko med dvema vozliščema teče poljubno število povezav. V usmerjenem grafu ima vozlišče dve stopnji — eno za vhodne in eno za izhodne povezave.

Najkrajša pot (ang. shortest path) med dvema vozliščema je definirana kot najmanjše število povezav na poti med opazovanima vozliščema [23]. Obstaja lahko več najkrajših poti, ki si lahko delijo vozlišča.

Osrednjost vozlišča (ang. betweenness) je definirana kot število najkrajših poti med ostalimi vozlišči, ki tečejo skozi opazovano vozlišče [23]. Je mera pomembnosti in vpliva nekega vozlišča na tok informacij med ostalimi vozlišči in se lahko nanaša tudi na povezavo. Uporabna je pri iskanju gruč in skupnosti v omrežju [18], kot pomembno mero pa smo jo uporabili tudi v razvoju prototipa (poglavje 4.4.2).

Modularnost omrežja [25] (ang. modularity) je kvantitativna mera, ki jo znanstveniki pri razvoju algoritmov za detekcijo skupnosti pogosto uporabljajo kot mero za kakovost razvitega ali uporabljenega algoritma, čeprav temu v osnovi ni namenjena. Po definiciji, določa "kvaliteto" posamezne skupnosti. V algoritmu za detekcijo skupin jo Leung et al. [19] uporabijo kot prag, ki določa, kdaj je neka skupnost postala dovolj "dobra", da lahko samostojno obstaja, sicer bi se lahko zgodilo, da bi algoritem vsa vozlišča združil v isto skupnost. Modularnost označimo s spremenljivko $Q \in [0, 1]$.

4.2 Pridobivanje podatkov za analizo

Znanstveniki in raziskovalci kakovost novih algoritmov in metod za analizo omrežij pogosto testirajo na znanih, realnih naborih podatkov. To jim omogoča dobro primerjavo z obstoječimi rešitvami, ki jih poskušajo z novimi pristopi izboljšati. Med primere pogosto uporabljenih socialnih omrežij sodijo *Zacharyev karate klub* [29], omrežje razporeda tekem prve divizije ameriške univerzitetne lige v ameriškem nogometu, Facebook, Twitter, Google+ omrežja prijateljstev in drugo. Na spletu je razpršena kopica naborov, velik in lepo urejen arhiv pa na stanfordski univerzi ureja slovenski raziskovalec dr. Jure Leskovec [10].

Zaradi specifike problema za podrobnejšo analizo potrebujemo nabore podatkov, ki opisujejo *omrežje e-poštnih sporočil več uporabnikov iz istega podjetja in njihove komunikacije s poslovnimi partnerji podjetja*. Glede na delitev v podpoglavju 4.1.1 omrežja e-poštnih sporočil sodijo med komunikacijska omrežja.

Eno takšnih, že obstoječih, je svetovno znani *Enron* [20]. Enron je e-poštno omrežje, objavljeno leta 2003, ki vsebuje podatke 158 zaposlenih podjetja "Enron Corporation". Zaradi bogate vsebine in ogromnega števila e-poštnih sporočil (skoraj 1.6 milijona) je izjemno priljubljeno omrežje med raziskovalci. Manjši nabor štiridesetih e-poštnih pogovorov (okoli 3300 sporočil) vsebuje *The BC3: British Columbia Conversation Corpus* [1]. V Juretovem spletnem arhivu pa so dostopni podatki o omrežju *raziskovalnih ustanov Evropske unije*, zbrani med letoma 2003 in 2005.

Vsa v prejšnjem odstavku navedena omrežja bi predstavljala dobro osnovo za analizo. Zmotilo nas je dejstvo, da o sami sestavi omrežja ni nobenih dodatnih informacij. Tako bi med analizo težko ovrednotili kakovost in natančnost algoritmov in rezultatov. Še največ dodatnih informacij ima omrežje Enron. Njegova pomanjkljivost je, da so podatki razpršeni po številnih datotekah in mapah, kar oteži njihovo pridobivanje.

V želji po bolj reprezentativnih podatkih smo uspeli pridobiti e-poštne podatke manjšega podjetja z desetimi zaposlenimi iz Slovenije. Nabor podat-

kov predstavlja e-poštno omrežje osmih zaposlenih v podjetju. V obdobju treh let (podatki so bili arhivirani 7. 8. 2013) so si med sabo ali s poslovnimi partnerji poslali okoli 163.240 e-poštnih sporočil. Trije zaposleni imajo znotraj podjetja vodstveno vlogo in v podjetju sodelujejo celotno časovno obdobje. Ostali so se kot razvijalci priključili naknadno. Na zahtevo podjetja so bili podatki anonimizirani. Nabor smo uporabili v analizi in pri razvoju prototipa, njihovo sestavo in uporabo pa bomo podrobneje opisali v nadaljevanju.

4.3 Preliminarna analiza

Analiza omrežij je kot že rečeno v zadnjih letih priljubljeno področje raziskovanj, čemur pričajo številni znanstveni prispevki na to temo. Posledično obstaja veliko različnih algoritmov, metod in mer, s katerimi lahko podrobneje preučimo strukturo in značilnost omrežja. V tem poglavju bomo opisali preliminarno analizo, ki smo jo izvedli na izbranem naboru podatkov, v želji, da bi podrobneje raziskali lastnosti izbranega omrežja. Natančneje, želeli smo preveriti, ali lahko iz podatkov s pomočjo algoritmov za detekcijo skupnosti poiščemo obstoj različnih skupin znotraj omrežja.

4.3.1 Priprava podatkov za analizo in definicija omrežja

E-poštni podatki, ki smo jih dobili od podjetja, so bili shranjeni v `mbox` formatu. Mbox je generično ime za družino formatov, ki jih pogosto uporabljamo za hrambo zbirk e-poštnih sporočil (v nadaljevanju sporočil) v tekstovni datoteki. Posamezna sporočila so ločena s prazno vrstico in shranjena v skladu z originalnim formatom, ki definira strukturo sporočila (`RFC 2822` [9]). Primer strukture je prikazan v kodi 4.1.

Vsako sporočilo sestoji iz *glave* in *telesa* sporočila. Indikator začetka glave sporočila je rezervirana beseda `From`, ki definira e-poštni naslov pošiljatelja. V nedefiniranem vrstnem redu ji sledijo `To` (e-poštni naslov prejemnika), `Cc` (ang. carbon copy), `Date` (datum in čas prejetega ali poslanega sporočila)

in `Message-ID` (enoličen identifikator sporočila). Ostala polja (`Reply-To`, `References`, `In-Reply-To`) se uporabljajo pri odgovoru (`Re`) in posredovanju (`Fwd`) sporočila, saj vsebujejo enolične identifikatorje predhodnih sporočil, ki se nanašajo na isti pogovor. Glava lahko vsebuje tudi druge rezervirane besede, zato je natančna struktura odvisna od posameznega e-poštnega odjemalca. Telo sporočila je običajno sestavljeno iz predmeta ali naslova (`Subject`) in telesa (ang. body), ki vsebuje vsebino sporočila.

Koda 4.1: Primer strukture e-poštnega sporočila [9]

```
From: Mary Smith <mary@example.net>
To: John Doe <jdoe@machine.example>
Cc: Gary Webber <gwebber@example.net>
Reply-To: "Mary Smith: Personal Account" <smith@home.example>
Date: Fri, 21 Nov 1997 10:01:10 -0600
Message-ID: <3456@example.net>
In-Reply-To: <1234@local.machine.example>
References: <1234@local.machine.example>
Subject: Re: Saying Hello

This is the body of the message.
```

Informacije, ki jih lahko razberemo iz glave sporočila, zadostujejo za gradnjo *komunikacijskega omrežja e-poštnih sporočil med zaposlenimi*, ki ga lahko definiramo kot:

Definicija 4.1 *Usmerjeno omrežje, kjer relacijo povezave predstavlja vsaka instanca e-poštnega sporočila, vozlišča pa množica vseh pošiljateljev in prejemnikov v glavi e-poštnega sporočila. Povezava je usmerjena od pošiljatelja (polje* `From`*) k vsem prejemnikom sporočila (polji* `To` *in* `Cc`*).*

V usmerjenem omrežju lahko med vsakim pošiljateljem in prejemnikom obstajata *največ dve usmerjeni uteženi ali neuteženi* povezavi. V neusmerjenem omrežju je *povezava le ena*, pri čemer se v primeru uteženega omrežja uteži na obeh povezavah seštejeta. V primeru, da sta polji `To` ali `Cc` vsebovali

več oseb, povezav med njimi nismo upoštevali, saj ni jasno ali med takšnima osebama obstaja neka realna povezava. Enako velja za povezavo med osebami v polju `To` in osebami v polju `Cc`. Definicija 4.1 predstavlja osnovo pri nadaljnji analizi omrežja in iskanju njegovih zakonitosti.

Vsako povezavo v omrežju smo dodatno opremili še z datumom in časom poslanega ali prejetega sporočila (polje `Date`). Ostalih polj v glavi sporočila nismo upoštevali, možnosti njihove uporabe pa smo podrobneje opisal v podpoglavju 6.2. Na podlagi e-poštnega naslova smo določili tudi *podjetje, ki mu neka oseba pripada*. To lahko naredimo z upoštevanjem domene e-poštnega sporočila, ki je v večini primerov enaka domeni spletne strani podjetja. Seveda lahko pri takšni preprosti definiciji pride do napak — na primer uporaba `gmail.com` naslovov za službene namene — vendar takšnih mejnih primerov za potrebe razvoja prototipa nismo upoštevali.

Anonimizirane podatke za analizo smo si za lažjo obdelavo pripravili v `csv` datoteko, katere primer je viden spodaj. (koda 4.2). Datoteko smo uporabili za računanje in izvajanje vseh v nadaljevanju opisanih mer in algoritmov.

Koda 4.2: Primer uporabljene (anonimizirane) .csv datoteke

```
To,From,Cc,Bcc,Date
1024017854,779628868,,,"Thu, 8 Nov 2012 09:57:01 +0100"
1024017854,1100499497,,,"Thu, 8 Nov 2012 09:57:46 +0100"
-1753224299,779628868,1024017854,,"Thu, 8 Nov 2012 09:59:21 +0100"
1024017854,779628868,,,"Thu, 8 Nov 2012 10:50:16 +0100"
...
```

4.3.2 Odkrivanje skupnosti v omrežju zaposlenih

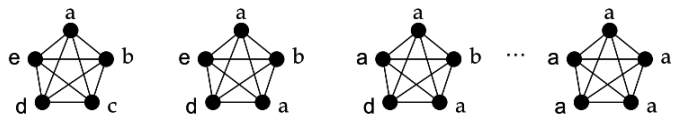
Z odkrivanjem skupnosti v omrežju zaposlenih smo poskušali *pridobiti dodatne informacije* o morebitnih skupinah ljudi, ki sodelujejo na istih projektih, spadajo v isto podjetje ali pa so tako ali drugače preko e-poštnih sporočil zelo povezani.

Ker je odkrivanje skupnosti v omrežju nedeterminističen problem, ne obstaja "najboljši" algoritem, ki bi za vsak nabor vhodnih podatkov vrnil enak optimalen rezultat. Na fakulteti za računalništvo in informatiko se v okviru laboratorija za podatkovne tehnologije z omenjeno tematiko ukvarja as. dr. Lovro Šubelj, ki ima na svoji spletni strani prosto dostopno implementacijo nekaterih algoritmov. Uporabo le teh na preučevanem naboru podatkov, bomo opisali v nadaljevanju.

Cilj algoritmov za detekcijo komun ali skupnosti je iskanje skupin znotraj nekega omrežja, ki so neločljivo povezana zaradi nekega skupnega, zunanjega pojma (podobni interesi, isti tipi molekul ipd.). Število skupnosti v omrežju določi algoritem, saj ni poznano v naprej.

V preteklosti, so bili za detekcijo skupnosti razviti številni algoritmi [18], ki omrežja delijo glede na tradicionalne metrike in mere. Med takšne sodi hierarhično razvrščanje v skupine (ang. hierarchical clustering), delitev na podlagi osrednjosti vozlišča (ang. edge betweenness) ipd. Njihova slabost je časovna kompleksnost, ki je v večini primerov $O(n^3)$ — kubična časovna kompleksnost.

Raghavan et al. [27] v svojem članku predlagajo algoritem, ki problem detekcije skupnosti reši v skoraj linearnem času $O(n + m)$, kjer je n število vozlišč, m pa število povezav v grafu. Poimenujejo ga *algoritem z izmenjavo oznak* (ang. label propagation algorithm). Osnovna ideja algoritma je sledeča: privzemimo, da ima vozlišče x sosede $x_1, x_2, ..., x_k$ in da ima vsak izmed njih oznako, ki določa, kateri skupnosti pripada (na začetku algoritma imajo vsa vozlišča različne oznake). Vozlišče x bo svojo oznako izbralo na podlagi oznak svojih sosedov in sicer tisto, ki ji pripada največje število sosedov. Gosto povezana vozlišča, s širjenjem oznak skozi omrežje, hitro dosežejo



Slika 4.4: Prikaz delovanja algoritma z izmenjavo oznak. Zaradi visoke gostote povezav, vsa vozlišča privzamejo isto oznako [27].

konsenz glede izbire enolične oznake (primer je lepo viden na sliki 4.4). Vozlišča z enako oznako po koncu algoritma predstavljajo isto skupnost.

Leung et al. [19] algoritem izboljšajo z uporabo metode *dinamičnega slabljenja oznak* (ang. label hop attenuation) in upoštevanjem *preference vozlišča* (ang. node preference). Z uporabo navedenih metod odpravijo glavno slabost osnovnega algoritma - nenadzorovano epidemično širjenje neke oznake po omrežju, kar povzroči tvorjenje izjemno velikih skupnosti.

Šubelj et al. [30] predlagajo algoritem z *napredno izmenjavo oznak* (ang. diffusion and propagation algorithm), ki združuje dve novi strategiji za detekcijo skupnosti, imenovani *zadržana in napadalna izmenjava oznak* (ang. defensive preservation and offensive expansion). Omenjeni algoritem za detekcijo skupnosti smo z dovoljenjem dr. Šublja uporabili na našem naboru podatkov. Algoritem je prilagojen za *neusmerjena neuteženа omrežja*. V osnovi deluje tako, da poskuša najprej s strategijo zadržane izmenjave oznak natančno določiti jedra skupnosti. Nato pa s strategijo napadalne izmenjave oznak, še izboljša jedra in poišče vozlišča, ki predstavljajo obrobje posamezne skupnosti. V nadaljevanju bomo predstavili rezultate algoritma, ki smo ga pognali na preučevanem naboru podatkov.

Nabor podatkov smo zaradi hitrejše obdelave prilagodili tako, da smo upoštevali vsa poslana ali prejeta sporočila *štirih zaposlenih v podjetju*. Primer kode 4.3 prikazuje izpis `log` datoteke algoritma, s katerim lahko ocenimo, kako dobro je algoritem v neusmerjenem neuteženem omrežju zaposlenih poiskal skupnosti. V podpoglavju 4.1.3 definirano modularnost uporabimo kot mero za kakovost algoritma. V izpisu sta poleg povprečne modularnosti (`Q`) prikazani minimalna (`min Q`) in maksimalna modularnost (`max Q`), število

vseh odkritih skupnosti (`#`), povprečno število vozlišč v skupnosti (`C`) ter število vozlišč v najmanjši (`min C`) in največji skupnosti (`max C`).

Koda 4.3: Izpis log datoteke algoritma (DPA*) za neusmerjeno neuteženo omrežje

```
Filename: Emails of 4 users in the company - unweighted
Graph: n = 5578, m = 17740, k =    6.36 (2742)
Algorithm: DPA(*)
Runs: 100
Time: 6 s 587 ms (0 s 35 ms)

   Q      min Q    max Q      #        C      min C   max C      Time

 0.380   0.363    0.382   170.67   38.39      2      3602      0 s 64 ms
```

Po izpisu sodeč je algoritem odkril približno 170 različnih skupnosti s povprečno 38 vozlišči (n). Več kot 3600 vozlišč je klasificiral v eno veliko skupnost, medtem ko je najmanjša skupnost sestavljena iz 2 vozlišč. Modularnost je v primerjavi z nekaterimi drugimi omrežji, ki so bila v članku dr. Šublja [30] testirana z istim algoritmom, zelo nizka. Čeprav nizka modularnost ni strogo odločujoč kriterij za kvaliteto najdenih skupnosti in algoritma, nam takšna klasifikacija v eno veliko skupnost *ne da dovolj informacij*, s katerimi bi lahko ugotovili, kaj v realnosti predstavljajo skupine, ki jih je odkril algoritem.

Predpostavili bi lahko, da je rezultat algoritma posledica uporabe neuteženega omrežja. V takšnem omrežju imajo vse povezave enako težo ($w = 1$), kar pomeni, da je vsaka povezava med dvema osebama enako pomembna. Slednje pa v poslovnem omrežju nekega podjetja zagotovo ne drži, saj obstajajo poslovni partnerji podjetja, s katerimi je podjetje ali oseba v podjetju v preteklosti ali sedanjosti sodelovala "močneje" kot z drugimi. Da bi preverili predpostavko, moramo omrežje zaposlenih nekako utežiti.

Koda 4.4: Izpis log datoteke algoritma (DPA*) za neusmerjeno uteženo omrežje

```
Filename: Emails of 4 users in the company - weighted
Graph: n = 5578, m = 17740, k =   6.36 (2742)
Algorithm: DPA(*)
Runs: 100
Time: 6 s 791 ms (0 s 38 ms)

   Q     min Q    max Q      #       C     min C   max C     Time

0.278   0.251    0.282   163.87   34.13     2      3474     0 s 67 ms
```

Kot utež, ki določa moč povezave, smo uporabili v podpoglavju 4.4.1 definirano mero *InteractionsRank*. Rezultati algoritma, ki smo ga pognali na neusmerjenem uteženem omrežju, so vidni v spodnjem izpisu (koda 4.4). Ugotovimo lahko, da predpostavka ni bila pravilna, saj *utežitev omrežja ni prinesla želenih izboljšav* pri klasifikaciji vozlišč v skupnosti. Še več, modularnost je v primeru uporabe uteženega omrežja celo nižja kot v neuteženem.

Iz klasifikacije velikega števila vozlišč v eno veliko skupnost bi lahko sklepali, da je podjetje močno povezano s svojimi poslovnimi partnerji. Zelo verjetno je, da podjetje zaradi svoje majhnosti ni razdeljeno na oddelke ali področja. Zaradi tega v vzorcu ne obstajajo skupine zaposlenih, ki v podjetju tesno sodelujejo med sabo. Da bi lahko svoje ugotovitve testirali, bi potrebovali nov nabor podatkov, ki pa ga nismo uspeli pridobiti. Ker nam uporaba algoritma za detekcijo skupnosti ni prinesla želenih dodatnih informacij o morebitnih "bolj povezanih" skupinah zaposlenih znotraj podjetja ali med posameznimi podjetji, je v nadaljnjem razvoju prototipa nismo uporabil.

4.4 Obvladovanje povezav v omrežju

Zaradi slabih rezultatov algoritmov za detekcijo skupnosti smo poskušali relacije med osebami v omrežju raziskati z uporabo kvantitativnih metrik in mer.

Po definiciji 4.1 je omrežje e-poštnih sporočil definirano kot usmerjeno omrežje, kjer je povezava usmerjena od pošiljatelja k prejemniku sporočila. Takšna definicija nam omogoča, da za vsako osebo natančno določimo *moč povezave med osebo in vsemi stiki*, s katerimi je preko e-poštnih sporočil komunicirala. V primeru kasnejše pretvorbe usmerjenega v neusmerjeno omrežje, skupna moč povezave predstavlja preprost seštevek obeh povezav. Po zgledu DataHug aplikacije, opisane v poglavju 3.1, smo želeli poiskati mero, s katero bi nazorno in enolično utežili vsako povezavo v omrežju. Poimenovali smo jo *"InteractionsRank"*. Poleg mere za moč povezave smo poskušali poiskati še druge statistične značilnosti, ki bi jih lahko pridobili iz omrežja. Razvoj in detekcijo bomo opisali v nadaljevanju.

4.4.1 Moč povezave

Če želimo ugotoviti, kakšne so relacije med osebami v omrežju, potrebujemo mero, ki bo določala moč povezave med vsakim parom vozlišč. *Moč povezave* v socialnih omrežjih po definiciji določa, kako dobro oseba A "pozna" osebo B. V primeru e-poštnega omrežja podjetja pa lahko z njeno uporabo določimo stopnjo e-poštne interakcije med dvema osebama (vozlišče) ali podjetjema (skupina vozlišč).

Pri računanju moči povezave (stopnjo interakcije) med vsakim parom vozlišč moramo upoštevati:

1. *Pogostost pošiljanja sporočil*: Večje število izmenjanih sporočil predstavlja višjo stopnjo interakcije.

2. *Aktualnost poslanega ali prejetega sporočila*: Vrednost poslanega ali prejetega sporočila s časom upada, kar pomeni, da imajo povezave med

pari oseb, ki trenutno aktivno sodelujejo, višjo stopnjo interakcije kot druge.

3. *Utež pomembnosti povezave*: Povezava med pošiljateljem in prejemnikom v `To` polju e-poštnega sporočila ima večjo utež kot povezava med pošiljateljem v prejemnikom v `Cc` polju.

Da bi zadovoljil navedene kriterije, smo po zgledu Googla [28] vpeljali mero *InteractionsRank* (v nadaljevanju $\mathcal{IR}$). Mero lahko izračunamo kot vsoto poslanih ali prejetih sporočil med osebama, pri čemer vsako e-poštno sporočilo utežimo z utežjo pomembnosti povezave in funkcijo njegove aktualnosti. Slednja je definirana kot funkcija *eksponentnega razpada* [12] v času in določa vrednost e-poštnega sporočila. $\mathcal{IR}$ izračunamo nad celotno množico povezav $I \in \{I_{FromTo}, I_{FromCc}\}$ po enačbi 4.1:

$$\mathcal{IR} \leftarrow \sum_{i \in I_{FromTo}} w_{FromTo} \left(\frac{1}{2}\right)^{\frac{t_{now} - t(i)}{\lambda}} + \sum_{i \in I_{FromCc}} w_{FromCc} \left(\frac{1}{2}\right)^{\frac{t_{now} - t(i)}{\lambda}} \tag{4.1}$$

kjer I_{FromTo} predstavlja množico vseh poslanih sporočil med pošiljateljem (polje `From`) in prejemniki v polju `To`, I_{FromCC} množico vseh povezav med pošiljateljem in prejemniki v polju `Cc`, w_{FromTo} in w_{FromCc} ustrezno utež pomembnosti povezave, t_{now} trenutni čas, $t(i)$ čas poslanega ali prejetega e-poštnega sporočila, ter λ nastavljiv parameter, ki določa, po kolikšnem času e-poštno sporočilo doseže polovico svoje vrednosti.

S parametrom λ lahko uravnavamo hitrost eksponentnega razpada. Sodeč po enačbi, ima neka interakcija v trenutnem času vrednost 1. Pri implementaciji prototipa smo vrednost nastavili na $\lambda = 2$, kar poenostavljeno pomeni, da je po dveh dneh e-poštno sporočilo vredno polovico prvotne vrednosti. Parametra, ki določata utež pomembnosti povezave, smo nastavili na $w_{FromTo} = 1$ in $w_{FromCc} = 0.5$. Po bontonu pisanja e-poštnih sporočil v polje `Cc` dodamo le prejemnike, na katere se vsebina e-poštnega sporočila neposredno ne nanaša, vseeno pa jih o poslanem sporočilu želimo "obvestiti". Tako

imajo povezave v e-poštnih sporočilih, kjer je naslovnik definiran v polju `To`, za polovico višjo utež.

Glavni problem mere $\mathcal{IR}$ je hiter, eksponentni padec vrednosti e-poštnega sporočila. Na primeru našega e-poštnega omrežja to pomeni, da imajo poslana in prejeta sporočila po nekaj dneh ali tednih že skoraj ničelno vrednost. V praksi lahko s takšnim obnašanjem mere sicer natančno določimo, s katerimi osebami neka oseba *trenutno* najbolj sodeluje. Zanemarimo oziroma ne upoštevamo pa oseb, s katerimi je močno sodelovala v preteklosti. To nam ne da najbolj realne slike o moči povezav med osebami ali podjetji v e-poštnem omrežju podjetja. V želji, da bi problem rešili in moč povezav določili neodvisno od časa, smo vpeljali mero *"ImportanceRank"*. Za njeno razlago, moramo najprej predstaviti in definirati mero *"Top Collaborators"*, zato jo bomo natančneje predstavili v naslednjem podpoglavju.

4.4.2 Ostale statistične informacije

Moč povezave med dvema osebama v omrežju, pa kljub pomembnosti, ni edina mera, ki jo lahko izračunamo na podlagi izbranega nabora podatkov. Iz podatkov, ki smo si jih pripravili, je moč razbrati še številne statistične informacije o pošiljateljih in prejemnikih, ki se pojavljajo v sporočilih ter njihovem obnašanju. Večina v nadaljevanju predstavljenih mer ali informacij na moč povezave sicer nima neposrednega vpliva. Ponujajo pa dodaten in bolj podroben vpogled v obravnavano omrežje.

V poglavju 3.1 o sorodnih delih smo kot glavno slabost obstoječih aplikacij izpostavili nezmožnost analize poslovnega omrežja več uporabnikov. To pomeni, da ne moremo enostavno ugotoviti, kdo od zaposlenih je v interakciji s katerim od stikov podjetja, kdaj sta bila nazadnje v interakciji ipd. Takšne informacije pa so za podjetje zelo dragocene. Zaradi dobro pripravljenega nabora podatkov (poglavje 4.3.1) je takšna analiza v našem primeru enostavna. Že samo z dodatnim upoštevanjem polja `Date` v e-poštnih sporočilih lahko za vsak stik podjetja sklepamo: kdo od zaposlenih je imel s stikom prvi kontakt in kdaj, kdo od zaposlenih je imel s stikom zadnji kontakt in kdo od

zaposlenih s stikom najbolj sodeluje.

Zastavljene sklepe smo v prototip aplikacije prenesli z vpeljavo naslednjih mer:

- *Introduced By*: množica oseb iz našega podjetja, ki je stik predstavila podjetju (prvi kontakt). Poiščemo jo lahko s primerjavo časov poslanih ali prejetih e-poštnih sporočil, v katerih se pojavi povezava med vsakim parom oseb. Iščemo najstarejše e-poštno sporočilo.

- *Last Contact By*: množica oseb iz našega podjetja, ki je bila s stikom nazadnje v interakciji. Poiščemo jo podobno kot mero "Introduced By", le da iskanje osredotočimo na najmlajša e-poštna sporočila.

- *First Email*: datum in čas prvega kontakta s stikom.

- *Last Email*: datum in čas zadnjega kontakta s stikom.

- *Top Collaborators*: zaposleni v našem podjetju, ki s stikom najbolj "sodelujejo". To pomeni, da imajo s stikom največji $\mathcal{IR}$.

Za vsako povezavo med dvema osebama smo poiskali statistične podatke o frekvenci interakcije med njima (število poslanih sporočil) v zadnjih 12 mesecih. Ločili smo jih po mesecih in glede na smer pošiljanja, pri čemer *poslana* sporočila pomenijo vsa e-poštna sporočila pošiljateljevm zaposlenih v domačem podjetju, *prejeta* sporočila pa tista, ki jih je poslal eden od poslovnih partnerjev podjetja. Z njimi smo zgradili *histogram poslanih in prejetih sporočil* v opazovanem obdobju.

V omrežju obstajajo osebe, ki imajo pri pretoku informacij skozi omrežje pomembno vlogo. Običajno so to vozlišča z veliko stopnjo, saj skozi njih tečejo številne najkrajše poti. V podpoglavju 4.1.3 smo takšna vozlišča poimenovali kot *centralna*. Takšne osebe so v omrežju pomembne, saj se obnašajo kot "most" med posameznimi deli omrežja (ang. hubs). To pomeni, da bi njihova morebitna odstranitev omrežje razdelila na več, potencialno neodvisnih komponent. Če poskušamo takšna vozlišča primerjati z realnimi

osebami, so takšne osebe običajno koordinatorji projektov ali različnih oddelkov. V podjetju imajo tako zaradi svoje lege pomembno in odgovorno vlogo. Na podlagi izračuna centralnosti vozlišča smo v razvoju prototipa aplikacije, za iskanje in določitev takšnih oseb uporabili mero *CollaborationRank* (v nadaljevanju $\mathcal{CR}$).

Omenili smo, da mera $\mathcal{IR}$ ne predstavlja optimalne mere za določitev moči povezave. Zaradi eksponentnega padca vrednosti e-poštnega sporočila se moč povezave v času hitro približuje ničelni vrednosti. Da bi lahko izračunali moč povezave neodvisno od časa poslanega ali prejetega sporočila, smo uvedli še mero *ImportanceRank* (v nadaljevanju $\mathcal{IM}$), v kateri smo upoštevali:

1. *število poslanih ali prejetih sporočil*: Večje število izmenjanih sporočil predstavlja višjo $\mathcal{IM}$;

2. *število najbolj povezanih oseb (Top Collaborators)*: Število zaposlenih iz domačega podjetja, s katerimi je oseba že sodelovala ter imajo najvišji $\mathcal{IR}$;

3. *trajanje interakcije*: Število dni med prvim (First Email) in zadnjim (Last Email) poslanim ali prejetim e-poštnim sporočilom.

Za končni izračun mere $\mathcal{IM}$ smo posamezne vrednosti utežili glede na njihovo pomembnost. Iz navedenega lahko izpeljemo enačbo 4.2:

$$\mathcal{IM} \leftarrow w_1 \cdot \texttt{NoOfEmails} + w_2 \cdot \texttt{NoOfTopCol} + w_3 \cdot \texttt{DaysBetween} \qquad (4.2)$$

kjer vrednosti $w_1 = 0.6$, $w_2 = 0.2$ in $w_3 = 0.2$ predstavljajo uteži, vrednosti `NoOfEmails`, `NoOfTopCol` in `DaysBetween` pa na prejšnji strani navedene kriterije.

Uporabo vseh opisanih mer bomo na primeru prototipa aplikacije predstavili v poglavju 5.

4.5 Metode in algoritmi za vizualno predstavitev omrežij

Pravijo, da slika pove več kot tisoč besed. Ljudje smo po naravi zelo vizualna bitja, saj naše oči predstavljajo eno najpomembnejših čutil. Človeško oko je izjemno učinkovito in dobro analitično orodje, zato je bilo analiziranje omrežij na podlagi njihove vizualne predstavitve dolgo časa ena primarnih metod raziskovanja. Vendar pa z naraščanjem števila povezav in vozlišč v omrežju narašča tudi zahtevnost njihove vizualizacije. Čeprav se zdi preprosto narisati omrežje z nekaj deset ali sto vozlišči, je uspešna in reprezentativna vizualizacija omrežja z milijon vozlišči, tudi z uporabo naprednih 3D računalnikov in algoritmov, skorajda nemogoča. Zato je sočasna uporaba analitičnih mer in pristopov, opisanih v prejšnjem poglavju, pri analizi zelo pomembna.

Glavni problem pri vizualni predstavitvi nekega omrežja je, *kako in kam v omejenem prostoru postaviti vozlišča in povezave*, da bo vizualna slika omrežja še vedno *uporabna, razumljiva in pregledna*. To pomeni, da lahko samo na podlagi slike sklepamo o lastnostih in zakonitostih relacij v predstavljenem omrežju.

V podpoglavju 4.1.3 smo omrežje definirali kot graf $G(V, E)$, sestavljen iz množice vozlišč V in množice povezav E med njimi. Tako definirana omrežja najpogosteje vizualiziramo v obliki ang. *node-link diagramov*, kjer s krogi predstavimo vozlišča, z ravnimi ali krivimi črtami pa povezave med njimi (slika 4.3). Usmerjene povezave pogosto prikažemo s puščicami, ki nakazujejo smer usmeritve povezave.

S preprostimi hierarhičnimi diagramskimi tehnikami (dendrogrami in drevesa) lahko pregledno predstavimo omrežja z do nekaj deset ali sto vozlišči, odvisno od velikosti omejenega prostora. Značilnost takšnih omrežij je v naprej poznana struktura omrežja. Obstajajo pa številna omrežja s predhodno nedefinirano strukturo, ki jim pravimo arbitrarna omrežja (ang. arbitrary graphs/networks). Zaradi pojava takšnih omrežij in naraščanja njihove kom-

pleksnosti in velikosti so se razvili algoritmi in pristopi, s katerimi lahko (do neke mere) predstavimo omrežja z več tisoč ali milijon vozlišči. Takšno omrežje obravnavamo v diplomski nalogi.

Cilj vseh metod in algoritmov za vizualizacijo je *razumljiva in pregledna predstavitev omrežja*. Med sabo se ločijo po načinu razporeditve in postavitve vozlišč ter povezav v omejeni prostor (ang. layout). Eden izmed najbolj popularnih pristopov je uporaba ang. *Force-directed* metod, ki delujejo tako, da lego vozlišča spreminjajo neprekinjeno, v skladu s privlačnimi in raztežnimi silami [15]. Po načinu postavitve jih najlažje primerjamo s fizikalnimi silami vzmeti ali pa silami, ki delujejo med molekulami. Zaradi uporabe v prototipu aplikacije bomo nekaj metod za izračun sil podrobneje opisali.

Za lažje razumevanje si pri Force-directed algoritmih vozlišča predstavljamo kot fizikalne delce, ki so na začetku naključno razporejeni v prostoru. Zaradi efekta sil se njihova lega v času spreminja, dokler ne zasedejo končne pozicije v grafu [21]. Izbira metod za izračun sil je odvisna od posameznega algoritma. Privlačne sile navadno delujejo med vsemi pari sosednjih vozlišč. Algoritmi za njihov izračun pogosto sledijo Hookovemu zakonu [16]. Njihova naloga je, da poskušajo gosto povezana vozlišča obdržati skupaj, a vidno ločeno od preostalega omrežja. Hkrati na vsa vozlišča v grafu, delujejo raztezne sile. Po zgledu delcev, na katere deluje električna sila, algoritmi za izračun temeljijo na Coulombovem zakonu [14]. Raztezne sile skrbijo za širitev vozlišč po definiranem prostoru, s čimer dosežejo, da se vozlišča med sabo ne prekrivajo. Algoritmi navadno tečejo v več korakih, kjer se v vsakem koraku za vsako vozlišče, glede na prejšnjo lego in lego ostalih vozlišč, izračuna nova lega vozlišča. Po koncu algoritma so povezave med sosednjimi vozlišči navadno enako dolge, medtem ko vozlišča, ki niso povezana z drugimi, zaradi odbojnih sil, ležijo na obrobju grafa. Primer uporabe takšnega algoritma je viden na sliki 4.5.

V bolj naprednih algoritmih metode za izračun privlačnih in odbojnih sil ne sledijo fizikalnemu obnašanju delcev, ali pa so definirane na logaritemski in ne linearni skali. Prav tako vsebujejo algoritme za izračun dodatnih sil, ki

so po delovanju podobne gravitacijski sili, magnetni sili ipd.

Kljub številnim različnim pristopom in izvedbam, imajo algoritmi sledeče *dobre* lastnosti:

1. *Dober in razumljiv prikaz rezultatov*: grafi z manjšim številom vozlišč (nekaj sto).
2. *Fleksibilnost*: so prilagodljivi specifičnemu problemu.
3. *Preprostost in intuitivnost*: algoritme lahko hitro in učinkovito implementiramo.
4. *Trdna teoretična podlaga*: kljub preprosti osnovni implementaciji imajo zaradi teoretičnega ozadja široke možnosti za nadaljnji razvoj.

Seveda pa vsebujejo tudi *slabe* lastnosti. Ena glavnih je velika časovna kompleksnost algoritmov. Tipično imajo kubično $O(n^3)$ časovno kompleksnost, kjer n predstavlja število vozlišč v grafu. Obstajajo nekatere izboljšave, kot je uporaba *Barnes-Hut simulation* algoritma [13] za optimizacijo časovne kompleksnosti. Kljub temu je njihova uporaba pogosta, sposobni pa so uspešno vizualizirati omrežja z milijoni vozlišč.

Obstajajo tudi nekateri alternativni pristopi k vizualizaciji [21], kot so ang. *arc diagrami*, v katerih so vozlišča nanizana v ravni navpični črti, *sosednostne matrike* (ang. adjacency matrix), kjer so vozlišča in povezave predstavljene kot celice in stolpci v matriki, *krožna predstavitev* (ang. circural layout), kjer so vozlišča razporejena po v naprej določeni krožnici ipd.

Vizualizacija omrežij je kompleksen pojem, katerega razvoj in uporaba presega okvire tega diplomskega dela. Bralec, ki ga vsebina zanima, lahko več informacij najde v navedenih referencah, ogromno informacij pa ponuja tudi svetovni splet.

Slika 4.5: Primer omrežja, kjer je način razporeditve vozlišč po omejenem prostoru rezultat Force-directed algoritma

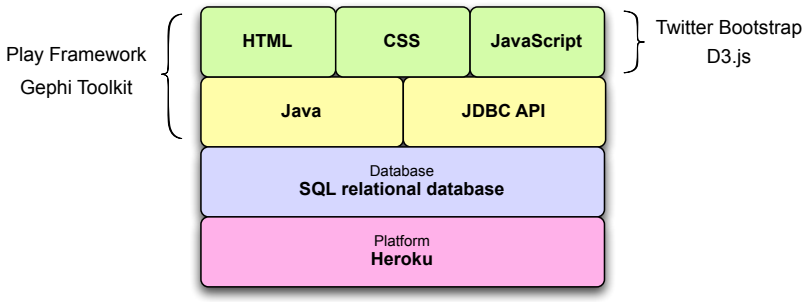
Poglavje 5

Prototip aplikacije

V prejšnjem poglavju smo predstavili konceptualni razvoj prototipa, v katerem smo podali osnovne pojme in definicije ter razložili razvoj uporabljenih mer. Cilj diplomskega dela je razvoj prototipa aplikacije, ki smo jo predlagali v podpoglavju 3.2. V tem poglavju bomo zato najprej predstavili koncept implementacije, nato prikazali uporabo in zaslonske maske končne verzije prototipa spletne aplikacije ter poglavje zaključili z opisom prenosa aplikacije na platformo Heroku.

5.1 Koncept implementacije

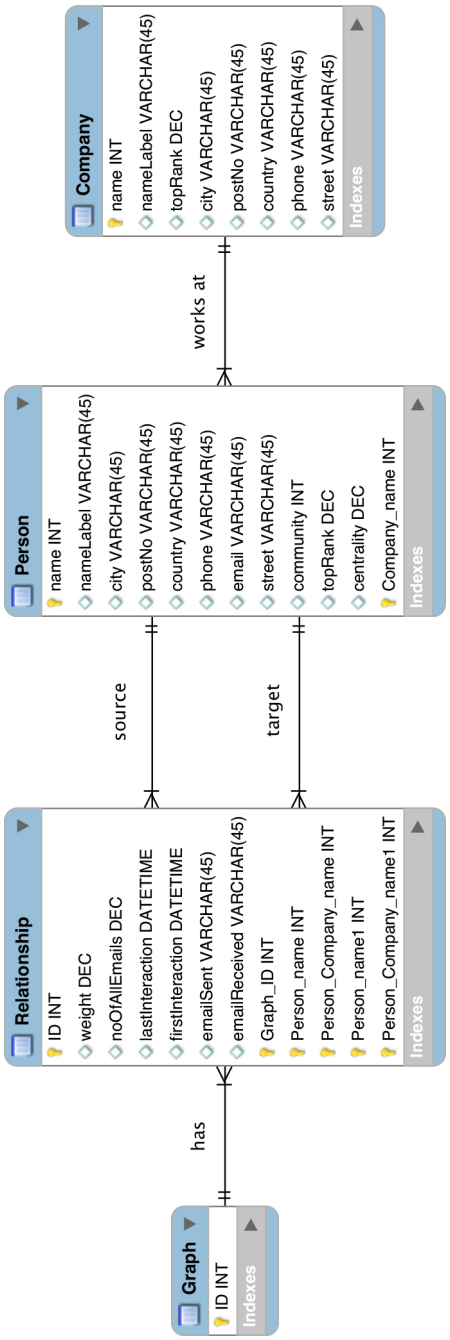
Spletno aplikacijo poganja ogrodje *Play*. Zgrajena je po MVC (ang. Model-View-Controller) arhitekturnem pristopu, ki jo razdeli na tri posamezne, med seboj na videz neodvisne enote. Podatkovni model (ang. model) je sestavljen iz podatkovnega nabora, logike in funkcij, ki v ozadju skrbijo za pravilno in nemoteno delovanje aplikacije. Uporabniški vmesnik (ang. view) predstavlja zaslonske maske aplikacije, ki so vidne končnemu uporabniku, medtem ko kontroler (ang. controller) skrbi za povezavo med zahtevami uporabnika in podatki, ki jih vrača model. Takšen pristop zahteva tudi ogrodje Play, kar nakazuje že z v naprej definirana datotečna struktura projekta, ki vsebuje aplikacijo.



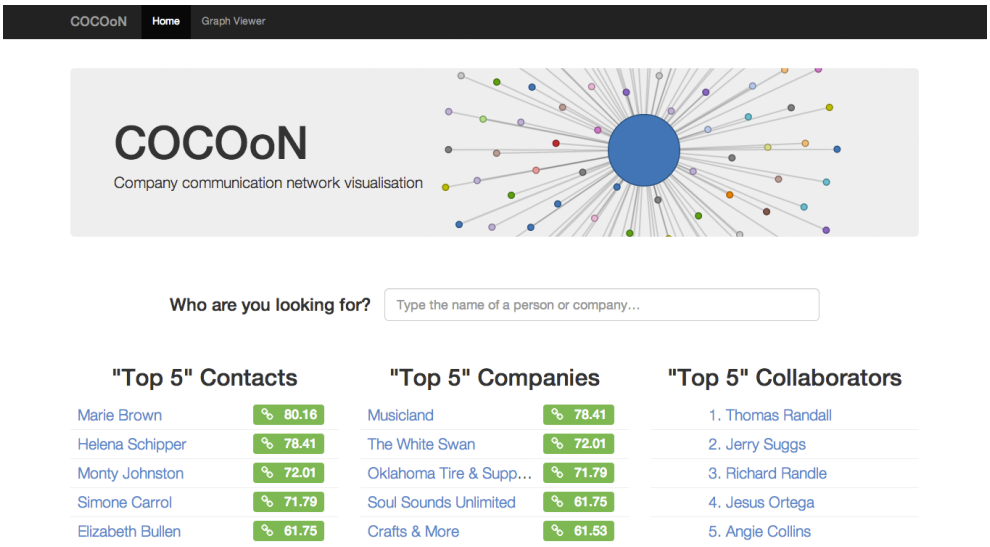
Slika 5.1: Shema arhitekturne zasnove prototipa spletne aplikacije

Aplikacijo smo razvili z uporabo programskih jezikov *Java, HTML, CSS in JavaScript*. S pomočjo Jave smo implementirali jedro aplikacije (logiko), ki skrbi za uvoz pripravljenega nabora podatkov, izgradnjo omrežja ter izračun mer in metrik, ki smo jih predstavili v podpoglavju 4.4. S pomočjo skriptnih jezikov HTML, CSS in JavaScript smo izdelali uporabniški vmesnik, ki ga bomo natančneje predstavili v naslednjem poglavju. Izdelani uporabniški vmesnik temelji na zbirki orodij, ki jih ponuja ogrodje *Bootstrap*.

V podrobnejšo implementacijo se v diplomski nalogi ne bomo spuščali, saj smo delovanje aplikacije predstavili predvsem na konceptualnem nivoju. Za boljše razumevanje je na sliki 5.1 viden koncept arhitekturne zasnove aplikacije, na sliki 5.2 pa konceptualni podatkovni model, ki smo ga uporabili za shranjevanje podatkov.



Slika 5.2: Konceptualni podatkovni model, uporabljen v prototipu spletne aplikacije



Slika 5.3: Primer zaslonske maske osnovne (domače) strani spletne aplikacije

5.2 Prikaz in uporaba

Na spletu obstaja kopica spletnih strani in aplikacij, ki se po izgledu, uporabnosti in vsebini zelo razlikujejo. S pojavom tehnologij, kot sta HTML5 in CSS3, vse več razvijalcev pozablja na osnovni namen spletnih aplikacij — *uporabnost*. Številni svoj trud raje vlagajo v izdelavo estetske in vizualno dovršene podobe. Takšne spletne aplikacije so na prvi pogled sicer "popolne", vendar končna uporabniška izkušnja večkrat kaže drugače. Navigacija po strani je težka, pogosto so prenapolnjene z vsebino ipd.

V želji, da bi zaposlenim v domačem podjetju (v nadaljevanju uporabniku) omogočili uporabno, razumljivo in pregledno rešitev, smo prototip spletne aplikacije zasnovali po konceptu "uporabniku prijazne" aplikacije. Ta sledi nekaterim osnovnim konceptom, kot so *razumljiva in hitra navigacija, berljivost in hiter odzivni čas*. Z uporabo ogrodja *Bootstrap* poseben trud pri implementaciji ni potreben, saj je večina osnovnih konceptov implementirana v samo rešitev.

Spletna aplikacija je zgrajena iz dveh podstrani (v nadaljevanju strani):

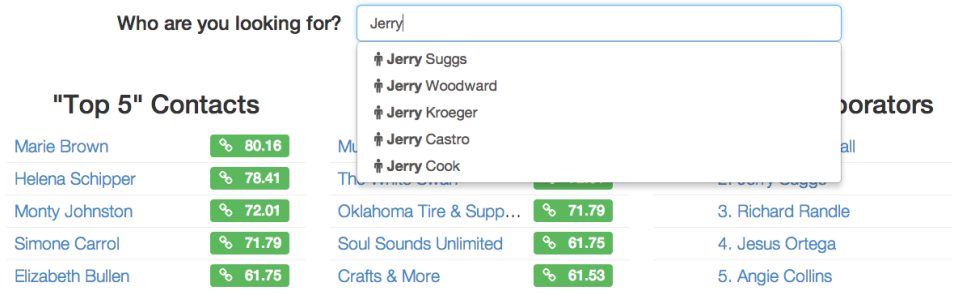
1. *Osnovna (domača) stran*: Namenjena iskanju in hitremu pregledu trenutno najbolj pomembnih stikov podjetja (slika 5.3).

2. *Stran za interaktivni pregled omrežja in analitiko*: Namenjena vizualni interakciji z omrežjem ter prikazu analitičnih vsebin (slika 5.6).

Na sliki 5.3 je prikazana zaslonska maska osnovne (domače) strani spletne aplikacije (v nadaljevanju aplikacija). S svojo zasnovo in zgradbo že na prvi pogled deluje pregledna, urejena in enostavna za uporabo. Najprej bomo v naslednjem odstavku predstavili njeno zgradbo s strani razvijalca, nato pa možne načine uporabe s strani uporabnika aplikacije.

Z razvijalskega stališča je osnovna stran zgrajena iz dveh delov: *navigacijske vrstice* in *vsebine*. V navigacijski vrstici so za boljšo predstavo in lažjo navigacijo definirane vse podstrani, do katerih lahko dostopamo. Vsebino osnovne strani lahko razdelimo na tri neodvisne komponente. Verjetno najbolj prepoznaven je ogromen *logotip*, ki s semantiko in vizualno podobo vsebinsko zaokroži njen pomen. Pod logotipom se nahaja *polje za iskanje stikov podjetja*, ki služi kot osnovna komponenta za prehod med osnovno stranjo in stranjo namenjeno prikazu omrežja in analitičnih vsebin. Za lažje iskanje stikov smo implementirali funkcijo ang. autocomplete, ki na podlagi delnega niza črk samodejno vrne vse delno ujemajoče se rezultate (slika 5.4). Sledi še zadnja komponenta, s katero smo želeli obogatiti vsebino osnovne strani ter uporabniku na enostaven način prikazati trenutno najbolj pomembne in aktivne stike podjetja.

S takšno postavitvijo komponent smo želeli uporabniku približati in poenostaviti uporabo aplikacije. Če se vrnemo na osnovna vprašanja, ki smo si jih zastavili v uvodu diplomskega dela, vidimo, da vsebina osnovne strani omogoča enostavno iskanje želenih odgovorov. Veliko vnosno polje vizualno nakazuje na njeno najpomembnejšo funkcionalnost — *iskanje stikov podjetja, ki jih želimo podrobneje preučiti*. Hkrati pa zgoščen tabelarični pregled trenutno najbolj aktualnih stikov podjetja in najpomembnejših zaposlenih,

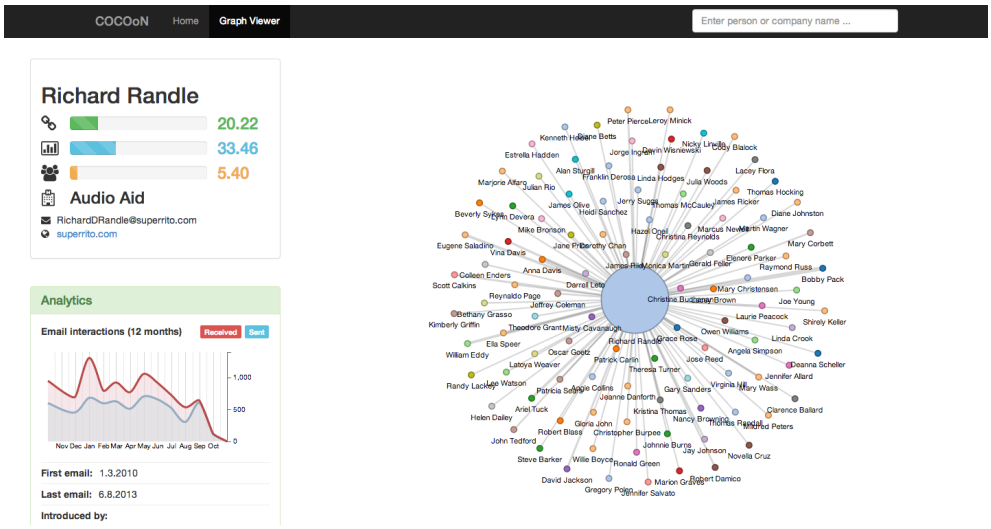


Slika 5.4: Primer funkcije ang. autocomplete, za samodejno dokončevanje iskalnih nizov

Slika 5.5: Zaslonska maska prikaza trenutno najbolj pomembnih stikov podjetja ("Top 5" Contacts, "Top 5" Companies) in najbolj aktivnih zaposlenih ("Top 5" Collaborators). Mera $\mathcal{IR}$ je prikazana z zeleno obrobo.

omogoča hiter dostop do informacij, ki so v tem momentu za podjetje bistvenega pomena.

Na sliki 5.5 si lahko podrobneje ogledamo vsebino omenjenih tabel. Tabela *"Top 5" Contacts* prikazuje osebe, ki imajo s podjetjem največji $\mathcal{IR}$ (zelena obroba). Vrednost $\mathcal{IR}$ predstavlja aritmetično sredino vseh vrednosti, torej vseh povezav neke osebe (stika) z domačim podjetjem. Po svoji definiciji (enačba 4.1) so to osebe, ki imajo trenutno največjo moč povezave, torej interakcijo z zaposlenimi v podjetju. Ker mera temelji na času poslanih ali prejetih sporočil, so v tabeli prikazane *najbolj aktualne osebe* - tiste, ki so z enim od zaposlenih komunicirale zadnje. Po enakem načelu so razvrščeni stiki v tabeli *"Top 5" Companies*, le da slednja prikazuje podjetja, s katerimi je domače podjetje komuniciralo nazadnje. Vrednost $\mathcal{IR}$ je torej



Slika 5.6: Zaslonska maska, ki prikazuje vizualni pregled omrežja ter analitiko izbrane osebe ali podjetja

izračunana kot aritmetična sredina vrednosti vseh stikov, ki so zaposleni v podjetju, ki ga opazujemo ter vseh stikov zaposlenih v domačem podjetju. Tabela *"Top 5" Collaborators* pa prikazuje zaposlene v domačem podjetju, ki imajo v podjetju najbolj centralno vlogo (mera $\mathcal{CR}$). S podatkom dobi uporabnik aplikacije pomembne informacije o zaposlenih, preko katerih bo lahko verjetno najhitreje prišel v kontakt s stikom v drugem podjetju, po katerem poizveduje.

V primeru da uporabnik izbere eno izmed iskanih oseb ali podjetij, ga aplikacija samodejno preusmeri na interaktivni vizualni prikaz omrežja ter analitike izbrane osebe ali podjetja (slika 5.6). V grobem stran sestoji iz dveh delov. Leva paleta vsebuje splošne podatke o iskani osebi ali podjetju, vse tri najpomembnejše mere ter splošno analitiko osebe, na desni pa je vidna interaktivna slika omrežja. Stran vsebuje še navigacijsko vrstico, v kateri je poleg osnovne navigacije dodatno iskalno polje za hitro iskanje po stikih podjetja.

Iz uporabniškega stališča je stran na sliki 5.6 jedro aplikacije. V enem oknu mu omogoči pregled nad vsemi najpomembnejšimi podatki in informa-

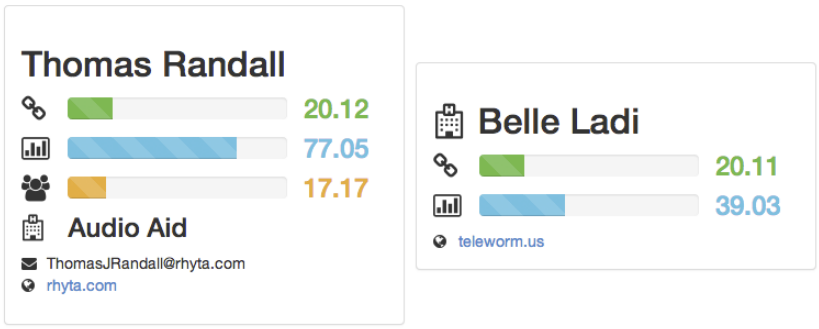
cijami o stiku, skupaj z vizualno predstavitvijo omrežja. Podatki so prikazani barvito in pregledno, kar še izboljša uporabniško izkušnjo.

Slika 5.7 prikazuje celici z osnovnimi informacijami o stiku, kjer vse tri najpomembnejše mere na videz izstopajo. Zeleni drsnik določa trenutno moč povezave ($\mathcal{IR}$), modri drsnik določa pomembnost osebe za podjetje ($\mathcal{IM}$), oranžni drsnik pa centralnost osebe v celotnem omrežju podjetja in vseh njegovih poslovnih partnerjev ($\mathcal{CR}$). Poleg drsnikov so mere predstavljene še v številski obliki. $\mathcal{IR}$ in $\mathcal{IM}$ lahko zavzameta vrednosti od $20 - 100$, kjer 20 predstavlja najnižjo vrednost 100 pa najvišjo. Za takšno domeno smo se odločili zato, ker menimo, da je vsako poslano ali prejeto e-poštno sporočilo pomembno. V kolikor bi povezave z najmanjšim $\mathcal{IR}$ lahko zavzele vrednost 1, bi jih marsikdo podzavestno podcenjeval. Tako pa z zamikom spodnje meje vrednosti povezave, njeno "pomembnost" ohranimo. $\mathcal{CR}$ lahko vseeno zavzame "normalne" vrednosti od $0 - 100$, kjer 0 predstavlja najnižjo vrednost 100 pa najvišjo, saj obstajajo povezave, skozi katere ne vodi nobena najkrajša pot. Takšne povezave morajo zavzeti vrednost 0, saj so ostale predstavitve ne realne. Poleg mer vsebuje paleta še osnovne informacije, ki jih lahko razberemo iz e-poštnega naslova stika: podjetje, e-poštni naslov stika in spletno stran podjetja.

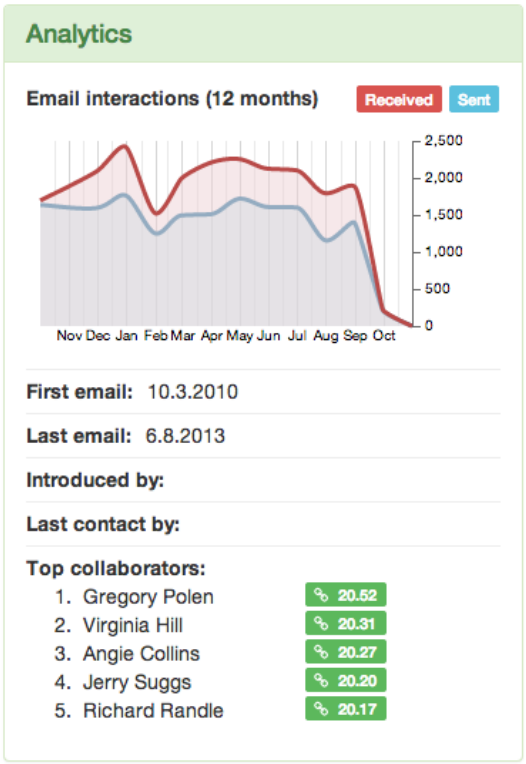
Pod celicami z osnovnimi podatki o stiku se v levi paleti nahajajo analitski podatki (slika 5.8). Gre za mere, ki smo jih opisali v poglavju 4.4.2. Od zgoraj navzdol si sledijo:

- *Email interactions*: histogram poslanih in prejetih sporočil, ki smo ga podrobneje že opisali v zgoraj omenjenem poglavju. Na sliki lahko vidimo končno realizacijo rešitve. Za vsak mesec, prikazuje informacijo o številu poslanih in prejetih sporočil med domačim podjetjem in stikom, za katerim poizvedujemo. V kolikor je stik podjetje, prikazuje vsoto vseh poslanih sporočil med vsemi zaposlenimi v obeh podjetjih.

- *First email*: datum in čas prvega e-poštnega sporočila med stikom in domačim podjetjem.



Slika 5.7: Celici, ki prikazujeta osnovne informacije in najpomembnejše mere. Leva paleta prikazuje informacije za primer osebe, desna pa podjetja. Najpomembnejše mere $\mathcal{IR}$, $\mathcal{IM}$ in $\mathcal{CR}$, so z različnimi barvami označene od zgoraj navzdol.



Slika 5.8: Paleta, ki prikazuje analitične podatke o stiku

- *Last email*: datum in čas zadnjega e-poštnega sporočila med stikom in domačim podjetjem.
- *Introduced by*: imena zaposlenih, ki so stik predstavila podjetju (prazno v primeru izbire stika iz domačega podjetja).
- *Last contact by*: imena zaposlenih, ki so si s stikom zadnja pošiljala e-poštna sporočila (prazno v primeru izbire stika iz domačega podjetja).
- *Top collaborators*: imena zaposlenih, ki imajo z stikom največji $\mathcal{IR}$ (zelena obroba).

Če primerjamo osnovno stran aplikacije in stran z analitiko, lahko opazimo uporabo skupnih konceptov pri predstavitvi podatkov. Iste mere so označene na enak način in z istimi barvami in ikonami, njihova velikost, navadno odraža njihovo pomembnost ipd. Vse rešitve sledijo konceptu uporabniku prijazne aplikacije, ki smo ga omenjali na začetku poglavja, hkrati pa z vizualno semantiko podzavestno pomagajo uporabniku pri dojemanju informacij.

Za konec ostane še opis rešitve za vizualizacijo omrežja. Sliki 5.9 in 5.10 prikazujeta vizualno podobo omrežja, ki smo ga izdelali s pomočjo JavaScript knjižnice *D3.js*. Omrežje je *neusmerjeno in uteženo*. Vozlišča v omrežju prikazujejo osebe, ki se pojavljajo v e-poštnih sporočilih. V primeru, da stik ki ga preučujemo, predstavlja osebo, je omrežje sestavljeno iz vseh oseb, s katerimi je povezan. To pomeni, da je z njimi komuniciral prek e-poštnega sporočila. Primer je lepo viden na sliki 5.9. Kadar stik predstavlja podjetje, je omrežje sestavljeno iz vseh oseb iz domačega podjetja, ki so z iskanim podjetjem komunicirala. Takšno omrežje predstavlja slika 5.10. V vizualni predstavitvi omrežja je zaradi zmogljivosti knjižnice D3.js predstavljenih največ 100 povezav, ki imajo z iskano osebo najvišji $\mathcal{IR}$. Estetske značilnosti omrežja lahko opišemo z naslednjimi atributi:

- *Barva vozlišč* prikazuje pripadnost oseb k različnim podjetjem, pri čemer so osebe iz istega podjetje označene z isto barvo. To nekako

nakazuje na skupnosti, ki se pojavljajo v omrežju, čeprav pripadnost podjetjem ni rezultat algoritmov za detekcijo skupnosti.

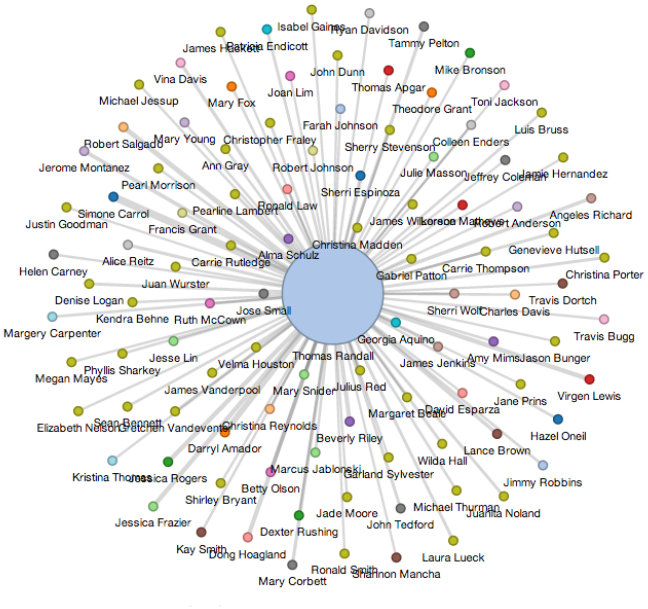
- *Velikost vozlišč* je sorazmerna s številom povezav, ki imajo en konec v opazovanem vozlišču. Na takšen način lažje razločimo pomembna vozlišča z velikim številom povezav (ang. hubs).

- *Barva povezav* v omrežju nima bistvenega pomena. Svetlo sivo barvo smo izbrali, ker sem želeli, da povezave vizualno ne izstopajo iz omrežja.

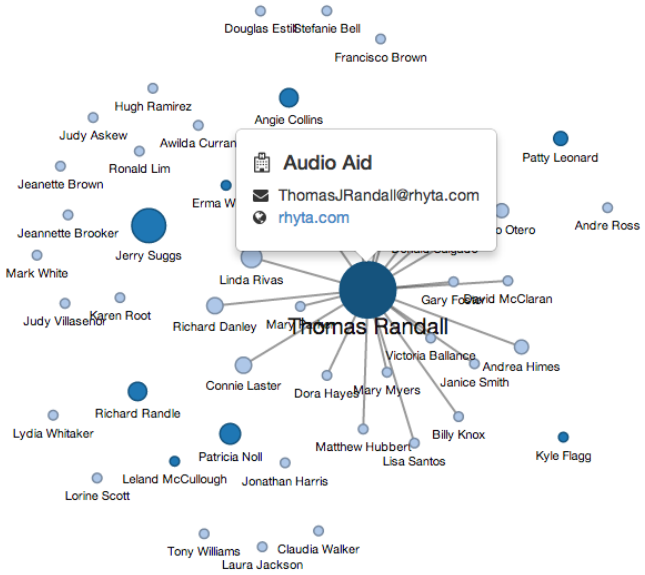
- *Širina povezav* je odvisna od velikosti $\mathcal{IR}$. Med pari vozlišč (oseb) z višjim $\mathcal{IR}$ tečejo širše povezave. Tako kot z velikostjo vozlišč, tudi s širino povezav vizualno ponazorimo njihovo pomembnost.

- *Oznake vozlišč* predstavljajo imena oseb, ki so prikazana v omrežju. Pomembni so pri njegovi semantiki, saj drugače ne bi vedeli katere osebe nastopajo v omrežju.

Velika prednost vizualizacije s knjižnico D3.js je enostavna implementacija interaktivnosti. Na sliki 5.10 je prikazan način in rezultat interakcije z omrežjem. Uporabnik lahko s preprostim gibanjem kazalca miške za vsako vozlišče poišče dodatne informacije, ki ga utegnejo zanimati. Prav tako se samodejno odstranijo povezave, ki se vozlišča ne dotikajo, kar uporabniku še olajša iskanje stika ali povezave. S klikom na vozlišče v omrežju aplikacija samodejno posodobi analitične podatke in izriše nov graf, ki sedaj opisuje pravkar izbrano osebo. Takšen interaktivni pristop k uporabi spletne aplikacije omogoča enostavno uporabo, poizvedovanje in navigacijo med vsemi stiki in poslovnimi parterji domačega podjetja.



Slika 5.9: Primer vizualne ponazoritve omrežja, ki predstavlja omrežje osebe “Thomas Randall” (veliko svetlo modro vozlišče)



Slika 5.10: Primer vizualne podobe omrežja povezav med zaposlenimi v dveh podjetjih. Slika prikazuje tudi način interakcije z omrežjem.

5.3 Prenos na platformo Heroku

V želji, da bi delovanje aplikacije preizkusili še v produkcijskem okolju, smo aplikacijo prenesli na platformo Heroku. Heroku smo predstavili že v poglavju 3, sedaj pa bomo natančneje opisali prenos in interakcijo s storitvijo.

Ena izmed dobrih strani uporabe ogrodja Play je enostavna integracija in povezava s platformo Heroku. Heroku namreč samodejno podpira Play aplikacije. To pomeni, da jih sam avtomatsko zazna in požene, saj so vse potrebne odvisnosti definirane v Play datotekah `application.conf` in `Build.scala`. Tako projektu ni potrebno dodati datoteke `Procfile`, v kateri bi sami definirali ukaze za zagon aplikacije.

Aplikacijo lahko na platformo prenesemo s pomočjo storitve *Git*. Git v osnovi deluje tako, da v posebnem repozitoriju hrani vse spremembe programske kode, ki jih je programer ali skupina programerjev naredila med razvojem aplikacije. To pomeni, da lahko v vsakem trenutku dostopamo do vsake verzije programske kode, ki smo jo do sedaj shranili v repozitorij. Za pravilno delovanje repozitorija je potrebno Git najprej inicializirati. To naredimo s posebnim ukazom `git init`, ki datotečno mapo, v kateri je shranjen naš projekt, pripravi na povezavo z Gitom (koda 5.1).

Koda 5.1: Inicializacija Gita v datotečni mapi, kjer imamo shranjen projekt

```
$ cd /path/to/app
$ git init
Initialized empty Git repository in .git/
```

Po inicializaciji moramo določiti, katere datoteke želimo povezati z Gitom. Z rezerviranim ukazom `add` jih dodamo, z rezerviranim ukazom `commit` pa Git opozorimo, naj ob prenosu vse spremembe v datotekah shrani v svoj repozitorij. Primer prikazuje koda 5.2. Git je s tem pripravljen na prenos programske kode na platformo Heroku.

Koda 5.2: Povezovanje datotek z Git repozitorijem

```
$ git add .
$ git commit -m "Initial commit"
[master 87f94f2] Commit before production push
46 files changed, 6 insertions(+), 5584 deletions(-)
 create mode 100755 .DS_Store
 mode change 100644 => 100755 .gitignore
 mode change 100644 => 100755 .target/ApplicationTest.class
 mode change 100644 => 100755 .target/IntegrationTest$1.class
...
```

Če želimo uporabljati Heroku, se moramo na platformo registrirati. Po uspešni registraciji lahko ustvarimo novo aplikacijo. To naredimo z ukazom `heroku create`. V izpisu (koda 5.3) vidimo, da Heroku samodejno ustvari aplikacijo ter ji dodeli naključno ime, ki ga lahko kasneje v nastavitvah spremenimo. V izpisu je *HTTP povezava* do strežnika, ki vsebuje aplikacijo ter *Git povezava*, potrebna za prenos programske kode na platformo. Če smo predhodno pravilno inicializirali Git, se povezava do Heroku samodejno doda v seznam oddaljenih strežnikov (ang. Git remote).

Koda 5.3: Izpis uspešno ustvarjene aplikacije na platformi Heroku

```
$ heroku create
Creating infinite-beyond-3159... done, stack is cedar
http://infinite-beyond-3159.herokuapp.com/ |
git@heroku.com:infinite-beyond-3159.git
Git remote heroku added
```

S tem smo pripravili vse potrebno za prenos aplikacije na platformo Heroku. Za začetek prenosa moramo izvesti rezerviran Git ukaz `git push`, ki mu dodamo še ciljno oznako, definirano v seznamu oddaljenih strežnikov. Z oznako povemo, v kateri oddaljeni repozitorij želimo prenesti programsko kodo. Ker želimo prenesti na Heroku, uporabimo privzeto oznako `heroku master`. V izpisu (koda 5.4) vidimo, da Heroku prepozna Play aplikacijo, jo zažene z

ukazom `sbt clean compile stage`, samodejno pridobi vse potrebne odvisnosti ter ji v primeru uspešnega prenosa dodeli web Dyno in HTTP naslov, na katerem teče.

Koda 5.4: Prenos aplikacije in njene programske kode na platformo Heroku

```
$ git push heroku master
Counting objects: 594, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (526/526), done.
Writing objects: 100% (594/594), 33.92 MiB | 552 KiB/s, done.
Total 594 (delta 231), reused 0 (delta 0)

-----> Removing .DS_Store files
-----> Play 2.x - Java app detected
-----> Installing OpenJDK 1.6...done
-----> Building app with sbt
-----> Running: sbt clean compile stage
       Getting net.java.dev.jna jna 3.2.3 ...
       downloading ...
       [SUCCESSFUL ] net.java.dev.jna#jna;3.2.3!jna.jar (388ms)
       :: retrieving :: org.scala-sbt#boot-jna
       confs: [default]
       1 artifacts copied, 0 already retrieved (838kB/18ms)
       Getting org.scala-sbt sbt 0.12.2 ...
       ...
       ...
       ...
-----> Dropping ivy cache from the slug
-----> Discovering process types
       Procfile declares types          -> (none)
       Default types for Play 2.x - Java -> web

-----> Compiled slug size: 212.2MB
-----> Launching... done, v6
       http://cocoondemo.herokuapp.com deployed to Heroku
```

S tem je prenos na platformo zaključen, prototip spletne aplikacije pa je dostopen na spletnem naslovu `http://cocoonproject.herokuapp.com`. Pri tem naj omenimo, da spletna aplikacija, ki smo jo prenesli na Heroku, zaradi omejitev, ne vsebuje celotne baze e-poštnih pogovorov.

Poglavje 6

Zaključek

6.1 Ugotovitve in sklepi

V diplomski nalogi smo poskušali z analizo e-poštnega omrežja podjetja raziskati značilnosti omrežja ter poiskati morebitne informacije, ki jih ne moremo najti s preprostim branjem in pregledovanjem e-poštnih sporočil. Razvili smo prototip spletne aplikacije, ki zaposlenim v podjetju omogoča lažji pregled nad vsemi stiki, ki se pojavljajo v prejetih ali poslanih e-poštnih sporočilih podjetja.

Z analizo nabora podatkov o e-poštnem omrežju manjšega slovenskega podjetja smo poiskali statistične informacije o naslovnikih in prejemnikih sporočil ter izračunali kvantitativne mere, s katerimi smo določili moč povezave med dvema osebama v omrežju. Moč povezave smo uporabili kot glavno mero za določitev stopnje povezanosti dveh oseb v omrežju. Raziskali in izmerili smo različne možnosti za določitev moči povezave. V eni od mer smo poleg števila e-poštnih sporočil upoštevali še časovno komponento poslanih ali prejetih e-poštnih sporočil, s čimer smo pridobili trenutno najbolj aktualne povezave v podjetju. Ker takšne povezave, zaradi padca pomembnosti sporočila, v času hitro izgubijo vrednost, smo v drugi meri čas zanemarili ter tako izračunali moč povezave v celotnem obdobju podjetja.

Z različnimi pristopi smo pridobili širok spekter informacij, ki smo jih

prikazali v spletni aplikaciji na prijazen in intuitiven način. S kombinacijo tabelaričnega prikaza statističnih informacij in interaktivnega vizualnega prikaza omrežja lahko vsak zaposleni hitro poišče podatke o stiku, po katerem poizveduje. Najpomembneje pa je, da lahko z uporabo aplikacije uspešno odgovori na vsa štiri osnovna vprašanja, ki smo si jih zastavili v uvodu diplomske naloge.

Čeprav smo v okviru diplomske naloge razvili prototip, ugotavljamo, da lahko uporaba le tega pomembno pripomore k uspešnosti sklepanja poslov v nekem podjetju. Tega se zavedajo tudi številna računalniška podjetja, ki v analizi kompleksnih omrežij iščejo nove tržne niše, v katerih bi lahko uspeli. Takšna podjetja vedo, da področje analize omrežij vsebuje ogromno možnosti za nadaljnje raziskave in razvoj. Nekatere izmed njih bomo na primeru razvitega prototipa predstavili v podpoglavju 6.2.

6.2 Problemi in možnosti nadaljnjega razvoja

Največji problem, ki smo ga imeli med analizo e-poštnega omrežja zaposlenih, je bil predhodno anonimiziran nabor podatkov. Čeprav nam v postopku implementacije to ni predstavljalo posebnih težav, nas je anonimizacija podatkov močno ovirala pri pregledu rezultatov, ki smo jih prikazali v prototipu aplikacije. Problem smo delno rešili z uporabo izmišljenega nabora imen oseb in podjetij. Kljub temu pa dobljene podatke in rezultate nismo mogli enostavno primerjati z morebitnimi povezavami ljudi v realnem življenju. Če bi bilo to mogoče, bi lahko bolj realno ocenil moč povezave.

Zaradi anonimizacije podatkov prav tako nismo mogli razločevati med e-poštnimi naslovi, ki dejansko pripadajo neki osebi in e-poštnimi naslovi, ki jih uporabljajo programi za avtomatsko pošiljanje e-poštnih sporočil. Takšna sporočila v komunikacijskem omrežju nimajo vrednosti, v prototipu pa se vseeno pojavljajo. Tukaj govorimo predvsem o sporočilih, katerih glavni namen je oglaševanje, o sporočilih, ki jih pošiljajo avtomatski odzivniki ali pa tistih, ki jih v e-poštnem žargonu označimo kot vsiljeno pošto (ang. spam).

V nadaljnjem razvoju prototipa bi jih morali izločiti, če bi želeli pridobiti bolj reprezentativne rezultate. Hkrati je res, da bi takšne povezave lahko izločili tudi vizualno znotraj aplikacije, oz. jih podzavestno ne bi upoštevali. Vendar to v primeru prototipa ni mogoče, saj se zaradi anonimizacije zgubi semantična sled za njimi.

Enega od problemov je predstavljala tudi uporaba knjižnice D3.js za vizualizacijo omrežja. Ker se programska koda izvaja na brskalniku, ni mogoče dobro vizualizirati omrežja z več kot 100 ali 150 vozlišči, saj izgubijo pregled in posledično uporabnost. Seveda, pa znotraj omrežja obstajajo številni zaposleni, ki imajo več kot 150 povezav z drugimi osebami. Problem smo rešili tako, da smo pri takšnih vozliščih prikazal le najpomembnejše povezave — tiste, ki imajo najvišji $\mathcal{IR}$. Na ta način smo pri vizualizaciji izgubili nekatere za podjetje potencialno pomembne stike. Problem bi lahko enostavno rešili tako, da bi uporabniku omogočili interaktivno izločanje povezav iz omrežja, pri čemer bi ostale vidne le povezave, ki jih ima iskana oseba z našim podjetjem. Tudi to je ena izmed možnih nadaljnjih izboljšav.

Če se vrnemo na podpoglavje 4.3.1 o pripravi podatkov za analizo, lahko na podlagi primera `csv` datoteke (koda 4.2) in primera zgradbe e-poštnega sporočila (koda 4.1) ugotovimo, da se v poljih `To` in `Cc` pojavlja tudi večje število prejemnikov. Na podlagi te informacije, bi lahko poiskali skupine stikov, ki se pogosto skupaj pojavljajo v e-poštnih sporočilih, oziroma natančneje, v omenjenih dveh poljih. Takšne skupine pogosto predstavljajo zaposlene in njihove poslovne partnerje, ki sodelujejo v istih projektih. V vizualizaciji omrežja bi jih lahko prikazali z isto barvo, ločeno od drugih skupin ter tako dobili nov, zelo informativen pogled nad omrežjem podjetja.

Dodatna izboljšava je možna tudi pri računanju moči povezave med dvema osebama. Vemo, da so prejeta ali poslana sporočila pogosto odgovor na neko začetno, predhodno poslano ali prejeto sporočilo. Takšna sporočila imajo v naslovu sporočila oznako `Re` ali `Fwd`. Od ostalih sporočil se v `mbox` formatu ločijo po tem, da vsebujejo rezervirani besedi `In-Reply-To` in `References`. Slednji vsebujeta kazalce na enolični identifikator sporočila `Message-ID`, ki

ga naslavljajo. V e-poštnem žargonu jih označimo kot *ugnezdena sporočila*. V trenutnem izračunu imajo vsa e-poštna sporočila enako težo. To pomeni, da vsako e-poštno sporočilo upoštevamo kot novo, samostojno instanco. Moč povezave tako ni odvisna od tega, ali je neka oseba samo odgovorila na prejeto sporočilo ali pa je z novim sporočilom sama izrazila pobudo o začetku komunikacije. Z upoštevanjem ugnezdenih sporočil lahko dodelimo večjo težo sporočilom, ki predstavljajo pobudo za začetek komunikacije ter s tem povezave bolj relevantno ovrednotimo. S tem bi v omrežju še lažje razločili zaposlene, ki sodelujejo z veliko različnimi poslovnimi partnerji, saj bi imele večjo stopnjo povezanosti ($\mathcal{CR}$) glede na ostale.

Pri implementaciji in razvoju aplikacije so se pojavili še številni manjši problemi, ki smo jih sproti rešili in za končni prototip spletne aplikacije niso bistvenega pomena.

Za konec naj omenimo, da je spletna aplikacija, ki smo jo razvili le prototip. To pomeni, da kljub številnim testom, verjetno obstajajo manjše napake v delovanju. Te bi lahko v nadaljnjem razvoju odpravili. Kljub temu predstavlja dobro osnovo za nadgradnje in izboljšave. Povsem realno je pričakovati in sklepati, da bi lahko v prihodnosti končno aplikacijo, v želji po izboljšanju poslovnih rezultatov, uporabljala podjetja širom sveta.

Literatura

[1] BC3 - British Columbia Email Corpus. http://www.cs.ubc.ca/labs/lci/bc3.html. Accessed: 03.08.2013.

[2] D3js: Data-Driven Documents. http://d3js.org/. Accessed: 01.08.2013.

[3] DataHug - solving "Who knows who?". http://www.datahug.com/. Accessed: 25.07.2013.

[4] Gephi: An open-sorce graph visualisation and manipulation software. https://gephi.org/. Accessed: 27.07.2013.

[5] Git: Distributed version control system. http://git-scm.com/. Accessed: 01.08.2013.

[6] Heroku: Cloud Application Platform. https://www.heroku.com/. Accessed: 01.08.2013.

[7] Immersion - A people-centric view of your email life. https://immersion.media.mit.edu/. Accessed: 25.07.2013.

[8] Play Framework: The High Velocity Web Framework For Java and Scala. http://www.playframework.com/. Accessed: 03.08.2013.

[9] RFC 2822 - Internet Message Format. http://tools.ietf.org/html/rfc2822. Accessed: 04.08.2013.

[10] SNAP - Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/index.html. Accessed: 03.08.2013.

[11] Twitter Bootstrap: Web Application Framework. http://getbootstrap.com/. Accessed: 03.08.2013.

[12] Wikipedia - Exponential Decay. http://en.wikipedia.org/wiki/Exponential_decay. Accessed: 04.08.2013.

[13] Wikipedia: Barnes-Hut simulation. http://en.wikipedia.org/wiki/Barnes%E2%80%93Hut_simulation. Accessed: 16.08.2013.

[14] Wikipedia: Coulomb' s law. http://en.wikipedia.org/wiki/Coulomb%27s_law. Accessed: 15.08.2013.

[15] Wikipedia: Force-directed graph drawing. http://en.wikipedia.org/wiki/Force-based_layout. Accessed: 15.08.2013.

[16] Wikipedia: Hooke' s law. http://en.wikipedia.org/wiki/Hooke%27s_law. Accessed: 15.08.2013.

[17] A. L. Barabasi and R. Albert. The structure and function of complex networks. e-print: arxiv:cond-mat/9910332.

[18] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. e-print: arxiv:cond-mat/0112110v1.

[19] Pietro Lio Jon Crowcroft Ian X.Y. Leung, Pan Hui. Towards real-time community detection in large networks. *PHYSICAL REVIEW*, 79, 2009. DOI 10.1103/PhysRevE.79.066107.

[20] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *CEAS*, 2004.

[21] Michael J. McGuffin. Simple algorithms for network visualization: A tutorial. *Tsinghua Science and Technology*, 17(4):383–398, 2012.

[22] S. Milgram. The small world problem. *Psychology Today*, pages 60–67, 1967.

[23] M. E. J. Newman. Scientific collaboration networks. II. shortest paths, weighted networks, and centrality. *PHYSICAL REVIEW*, 64, 2001. DOI 10.1103/PhysRevE.64.016132.

[24] M. E. J. Newman. The structure and function of complex networks. 2003. e-print: arxiv:cond-mat/0303516.

[25] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. 2003. e-print: cond-mat/0308217.

[26] Mark E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[27] Usha N. Raghavan, Reka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks, September 2007. e-print: abs/0709.2938.

[28] Maayan Roth, Assaf Ben-David, David Deutscher, Guy Flysher, Ilan Horn, Ari Leichtberg, Naty Leiser, Yossi Matias, and Ron Merom. Suggesting friends using the implicit social graph. In *KDD*, pages 233–242. ACM, 2010.

[29] Wayne Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[30] Lovro Šubelj and Marko Bajec. Unfolding communities in large complex networks: Combining defensive and offensive label propagation for core extraction. *Phys. Rev. E*, 83, Mar 2011.

[31] Lovro Šubelj and Marko Bajec. Clustering assortativity, communities and functional modules ineal-world networks. 2012. e-print: arxiv:cond-mat/1202.3188v1.

[32] Lovro Šubelj, Slavko Žitnik, Aljaž Zrnec, Aleš Kumer, Bojan Klemenc, Dejan Lavbič, and Marko Bajec. Lastnosti velikih realnih omrežij in primeri uporabe. In *Proceedings of DSI '12*, pages 3–6, 2012.

Seznam slik

Seznam kod