Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Primož Kerin

Identifikacija profilov istih oseb na različnih socialnih omrežjih

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Marko Bajec

Ljubljana 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Večina ljudi je vključenih v različna socialna omrežja. Pri registraciji ne podajo vedno istih podatkov (uporabniški profil), zato je težko ugotavljati, kdaj se profili nanašajo na isto osebo. V diplomskem delu preučite to problematiko ter razvijte algoritem, ki bo znal ugotoviti, ali se dva profila na različnih socialnih omrežjih nanašata na isto osebo.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Primož Kerin sem avtor diplomskega dela z naslovom:

Identifikacija profilov istih oseb na različnih socialnih omrežjih

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Bajca,
- 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 na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 13. marca 2016 Podpis avtorja:

Na tem mestu bi se rad zahvalil mentorju prof. dr. Marku Bajcu, svoji družini, prijateljem, vsem, ki so me podpirali pri izdelavi diplomske naloge, predvsem pa prijateljem iz Datafy.it, ki so mi omogočili prostor, opremo za implementacijo in testiranje algoritmov ter s svojim znanjem in nasveti prispevali, da sem od izdelave diplomske naloge odnesel čim več.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
SVM	support vector machine	metoda podpornih vektorjev
URL	uniform resource locator	enolični krajevnik vira
LCS	longest common subsequence	razdalja najdaljšega skupnega podniza
ID	identifier	identifikator
CSV	file with comma separated values	datoteka z vejico ločenimi vrednostmi

Povzetek

V diplomskem delu sem kombiniral več različnih tehnik za prepoznavo podvojenih oseb, kot so metode, ki temeljijo na pravilih določanja podobnih oseb, klasifikacija parov, sestavljanje gruč in strojnega učenja. Duplicirane osebe sem dodatno prepoznaval s primerjavo profilnih fotografij, saj je njihov vizualni učinek tudi pri človeški prepoznavi v veliko pomoč. Čeprav so podvojeni zapisi shranjeni v eni podatkovni bazi, sem vseeno uporabil tehnike, ki se drugače izkoriščajo tudi za uparjanje več baz z enakimi entitetami. Na koncu sem ocenil časovno zahtevnost in pravilnost deduplikacije.

Ključne besede: duplikati, socialno omrežje, spajanje, čiščenje podatkov.

Abstract

In my thesis I combined multiple duplicate person recognition techniques like rule-based methods to determine similar persons, pair classification, cluster building and machine learning. Extra comparison of profile pictures was used for recognizing person duplicates, because this comparison is the one of the first things that humans use when comparing profiles. I also used techniques for entity resolution on multiple databases. In the end I measured time complexity and success of the deduplication.

Keywords: deduplication, social network, merge, data cleaning, record linkage.

Poglavje 1

Uvod

Družabna omrežja predstavljajo mrežo uporabnikov s celega sveta. Povezave med raznimi socialnimi omrežji pa so precej slabše določene. Izbira socialnega omrežja, v katerega se bo uporabnik registriral, je odvisna predvsem od popularnosti na tistem območju, kjer uporabnik živi oz. območja, kjer so registrirani uporabniki, s katerimi se želi povezati. Tako na različnih območjih prevladujejo različna socialna omrežja. Izbira je odvisna tudi od popularnosti določene industrije in položajev v podjetjih.

V primeru profesionalnih socialnih omrežjih je svetovno najbolj znano omrežje LinkedIn. V nemško govorečih državah še vedno prevladuje Xing (v Nemčiji je registriranih 7,3 milijona uporabnikov [2], medtem, ko na LinkedIn-u 3.6 milijona [1]) in v francosko govorečih Viadeo. Profesionalna socialna omrežja predstavljajo omrežja, v katerih se povezujejo ljudje zaradi poslovnih namenov, npr. iskanja službe, novih zaposlenih in ostalih priložnosti.

Zaradi različnih razlogov se nekateri uporabniki registrirajo na več različnih profesionalnih socialnih omrežjih. Tako lahko iskalci novih poslovnih stikov naletijo na iste osebe iz različnih socialnih omrežij. Ti načeloma lahko prepoznajo, kdaj gre za isto osebo, jaz pa sem ta problem želel rešiti programsko.

Ker je izdelava profilov na socialnih omrežjih v veliki večini prepuščena uporabnikom, prihaja do različnih interpretacij navodil in razlik pri vnosu

podatkov v zahtevana polja in obrazce, ki jih morajo izpolniti.

Vzorci dogajanj in razlik, ki sem jih sam opazil pri pregledovanju in raziskovanju profilov uporabnikov že znotraj enega od socialnih omrežij:

Zatakne se že pri osebnih imenih, ki so lahko zapisana v različnih oblikah:

- J. Novak,
- Jože Novak,
- Jože N.,
- Jože P. Novak

Nato so tu še različne oblike istih imen oz. okrajšave in izpeljanke:

- Robert,
- Bob

Vse to predstavlja problem pri prepoznavanju istih oseb.

S pomočjo konteksta oz. dodatnih podatkov je ljudem veliko lažje prepoznati, ali gre dejansko za eno in isto osebo, kot kateremu koli algoritmu, saj hitro najdejo povezave med njimi. Največji izziv je najti/razviti algoritem, ki bo deloval 100% zanesljivo in bo obenem še dovolj hiter. Med raziskovanjem dupliciranih profilov sem ugotovil tudi, da si ljudje najprej izdelajo en profil, ki ga nato iz različnih razlogov (npr. pozabljeno geslo ali zbrisan poštni račun) zanemarijo in ustvarijo novega, kjer je večina podatkov enakih, nekaj vrednosti pa dodanih oz. posodobljenih. Največkrat se spremembe opazijo pri profilnih slikah in dodatnih delovnih mestih. Manj sprememb je pri osebnih imenih, čeprav do teh prihaja med različnimi socialnimi omrežji. V teh primerih je največkrat dodan oz. manjkajoč naziv (Dr., prof., itd).

Za reševanje problema je bilo predlaganih že veliko rešitev, npr. Newcombe in drugi [3], ki so predstavili problem združevanja zapisov v bazah. Večina predlaganih rešitev pa število primerjav zmanjša tako, da objekte s podobnimi vrednostmi atributov zberejo v skupine, v katerih se potem primerjajo samo objekti znotraj vsake skupine.

Poglavje 2

Podatki so “umazani”

2.1 Predstavitev problema

Problem, ki sem ga s to diplomsko nalogo želel rešiti, je identifikacija profilov, ki se nanašajo na iste osebe, tako znotraj profesionalnih družabnih omrežij, kot tudi med različnimi profesionalnimi družabnimi omrežji. Na osnovi tega bi lahko razvil storitev, ki bi omogočala dopolnjevanje profilov.

V mojem primeru sem podatke o osebah, ki so na voljo brez registracije, pridobil iz omrežij Linkedin, Xing in Viadeo.

Podvojeni podatki negativno vplivajo na podatkovno shrambo in pridobivanje informacij iz nje, kvaliteto podatkov, zmogljivost celotnega podatkovnega sistema in na splošno uporabnost podatkov. Duplicirani podatki povzročajo nepotrebno smetenje podatkovne zbirke, zasedanje virov tudi pri poizvedbah, še posebno pa povzročajo probleme pri posodabljanju zapisov v bazi. Poleg tega pa so duplikati pogosto nepopolni, z manjkajočimi podatki, pri katerih le unija vseh prekrivajočih podatkov zagotavlja popolno razumevanje podatkovnega elementa.

Pri reševanju teh težav sem se srečeval z različnimi izzivi, kot sta implementacija metod, s katerimi ocenim enakost oz. podobnost oseb, njihova zanesljivost in pa seveda časovna zahtevnost oz. hitrost primerjanja velikega števila oseb med seboj, saj so v podatkovni zbirki javni podatki o več kot

milijon oseb.

Ponavadi se pri prepoznavi duplikatov srečujemo z naslednjimi izzivi:

- Dvoumnost imen oz. atributov. Naprimer, ko imata dve ali več oseb enako ali podobno ime ali ostale lastnosti.
- Napake v zapisih, kot so napačna črkovanja, napačno vneseni podatki, neskladnost vpisanih podatkov, itd.
- Manjkajoče vrednosti: Če primerjamo dve osebi in ena od njih za določen atribut nima vnešene vrednosti, to še ne pomeni, da sta to dve različni osebi. Razlogov za manjkajoče vrednosti je več, npr. uporabniki jih niso vnesli, ali pa se niso pravilno shranili v podatkovno bazo.
- Atributi so se po določenem časovnem obdobju spremenili, uporabniki, ki so prisotni na več socialnih omrežijih, ne posodabljajo vedno vseh svojih profilov hkrati, zato lahko pride do neposodobljenih vrednosti na enem profilu in aktualnih na drugem.
- Oblika podatkov, npr. datumov
- Jezik vpisanih podatkov
- Okrajšave, kratice...

2.2 Hipoteza

Moja hipoteza je bila, da se poleg hitrejše računalniške prepoznave duplikatov pri tem tudi približam zanesljivosti prepoznave oz. vsaj identificam resnične pare duplikatov in tiste, ki to definitivno niso. Če rezultate merim v kontekstu natančnosti in priklica, sem želel doseči čimvečjo natančnost. Problem, ki sem ga tu pričakoval, je bila dejanska možnost meritve teh dveh rezultatov, saj v konkretni podatkovni zbirki zapisi niso bili označeni.

2.3 Pregled obstoječih rešitev

2.3.1 Verjetnostno razreševanje duplikatov

Ker je podvojenost podatkov že dolgo časa problem, so ga poskušali rešiti že leta 1959, ko so se Newcombe in drugi [3] srečevali s podvojenimi bolniškimi zapisi. Ena izmed njihovih idej je bila, da imajo manj pogosta imena večjo razločevalno moč od pogostih in, če se dva zapisa združita v novega, se njuna rezultata podobnosti kombinirata. Fellagi in Sunter [4] sta nato formalizirala njihovo intuicijo in uradno predstavila matematični model, ki je zagotavljal teoretično ogrodje za računalniško reševanje problema dupliciranih zapisov. Pari, ki se ujemajo z rezultatom podobnosti nad določenim pragom, so določeni kot ujemanja, pari med najnižjim in najvišjim pragom, so označeni za možne duplikate in pari pod najnižjo mejo, se obravnavajo kot različni. Ker niso imeli učne množice, je bilo določanje pragov za doseg visoke kvalitete zelo težko.

Verjetnostni model je uporabljal Bayesov pristop, ki je klasificiral pare v dva razreda, *ujemanja* in *neujemanja*. Sistemi za nadzorovano učenje so se zanašali na obstoj učne množice v obliki parov zapisov, ki so bili prej označeni kot duplikati ali ne. Problem pri tehnikah nadzorovanega učenja je zahteva po velikem številu označenih podatkov. Medtem, ko je ustvarjanje velikega števila učnih parov, ki so zagotovo ujemanja in neujemanja dokaj enostavno, je generiranje primerov, ki bi pomagali ustvariti visoko natančen klasifikator, precej težje. Zato so Atlas in drugi [5] uporabili tehniko aktivnega učenja, pri kateri se aktivno izbira podmnožico neoznačenih podatkov in ti podatki, po procesu označevanja, zagotavljajo najbolj kvalitetno informacijo klasifikatorju.

2.3.2 Pristopi, ki temeljijo na pravilih

To so pristopi, ki so poseben primer pristopov baziranih na razdaljah, kjer z uporabo pravil definiramo, ali sta dva zapisa enaka ali ne. Wang in Ma-

dnick [6] sta, za primer zaznave podvojenih zapisov, predlagala uporabo pravil, razvitih s strani ekspertov, za izpeljavo množice atributov, ki služijo kot ključ za vsak zapis. Z uporabo teh pravil sta generirala unikatne ključe, ki naj bi več zapisov, ki predstavljajo isti objekt, spravili v skupine. Lim in drugi [7] so poleg uporabe tega pristopa dodali pogoj, da mora biti rezultat teh pravil vedno pravilen. Hernández and Stolfo [8] sta to idejo še dodatno razvila in izpeljala teorijo, ki specificira sklepanje o podobnosti zapisov. Na primer, če imata osebi podobno ime in enako delovno mesto v podjetju, lahko sklepamo, da je to ista oseba.

2.3.3 Konkretne rešitve zaznave duplikatov na omrežjih

Dosti raziskav se je ukvarjalo tudi s konkretnimi problemi zaznave duplikatov na različnih socialnih omrežjih, med njimi Bilgic in drugi [10], ki so razvili D-Dupe, orodje za razločevanje entitet na družabnih omrežjih. Orodje od uporabnikov, ki se morajo odločiti ali gre za iste osebe ali ne, zahteva odločitvi *Da* ali *Ne*, ne omogoča pa neodločenosti. Poleg tega zahteva podatke o relacijah med osebami. Naj omenim še diplomsko delo Tomaža Kuralta [9], ki je razvil avtonomen sistem za združevanje poljubnega števila omrežij. Pri svoji konkretni rešitvi problema zaznave duplikatov na različnih profesionalnih družabnih omrežjih, si nisem mogel pomagati z medsebojnimi povezavami uporabnikov, saj ti podatki na obravnavanih omrežjih niso bili dostopni.

2.3.4 Moje delo

Preveč generične rešitve problema deduplikacije so me spodbudile, da sem za svoj primer potreboval bolj konkretno rešitev deduplikacije oseb na različnih profesionalnih družabnih omrežjih, torej algoritem, s katerim bi ugotavljal ali se dva profila nanašata na isto osebo. V svojem diplomskem delu sem poizkušal izboljšati natančnost prepoznave podvojenih oseb s kombinacijo aktivnega strojnega učenja za pridobitev začetnega nabora označenih duplikatov in nato z dodatnimi metodami preverjati rezultate prepoznanih dupli-

katov tudi s pomočjo primerjanja profilnih fotografij. Moje delo je kombinacija aktivnega učenja za nabor in učenje na podatkih, blokiranja podobnih oseb v skupine in na koncu zagotavljanje pravilnosti deduplikacije z uporabo veriženja pravil.

2.4 Čiščenje podatkov

Preden se lotimo razpoznave entitet, je potrebno oz. priporočljivo pripraviti in očistiti podatke. To je proces zaznavanja in popravljanja ali brisanja poškodovanih oz. nepravilnih podatkov iz podatkovne zbirke. Zaradi velikosti moje podatkovne zbirke ročno čiščenje ne pride v poštev. Očiščeni podatki zvišujejo kakovost zaznavanja dupliciranih entitet. Na voljo imamo možnost različnih metod čiščenja podatkov za vsako vrsto podatka posebej, nekaj pa je tudi splošnih, npr. vse nize pri primerjanju pretvorimo v male črke in manjkajoče podatke označimo s praznim nizom. V splošnem je treba vsako vrsto podatka o osebi pretvoriti v čim bolj enotno obliko, da po nepotrebnem ne znižujemo kakovosti primerjanja teh podatkov.

2.5 Lastnosti deduplikacije

Enostavni in praktični pogoji v funkcijah, ki določajo ujemanje in združevanje oseb so sledeči:

Vsak zapis entitete Oseba je predstavljena z oznako p. $\langle\rangle$ označujeta objekt, ki nastane po združenju zapisov.

- Komutativnost: $\forall p_1, p_2$: $p_1 \Rightarrow p_2$, če $p_2 \Rightarrow p_1$ in, če sta p_1 in p_2 duplikata, potem $\langle p_1, p_2\rangle$ in $\langle p_2, p_1\rangle$ rezultirata v enak zapis
- Idempotentnost: $\forall p$, $p \approx p$ in $\langle p, p\rangle = p$. Vsak zapis p je enak samemu sebi.
- Asociativnost pri združevanju: $\forall p_1, p_2, p_3$, kjer obstajata oba $\langle p_1, \langle p_2, p_3\rangle\rangle$ in $\langle\langle p_1, p_2\rangle, p_3\rangle$, velja: $\langle p_1, \langle p_2, p_3\rangle\rangle = \langle\langle p_1, p_2\rangle, p_3\rangle$

Tranzitivnost

Če ugotovimo, da sta osebi p_1 in p_2 podobni in sta v istem trenutku tudi osebi p_2 in p_3 označeni, kot podobni, potem lahko po pravilu tranzitivnosti osebi p_1 in p_3 označimo, kot podobni.

Recimo, da imamo tri zapise, p_1, p_2, in p_3. Če je p_1 dovolj podoben p_2, ampak različen od p_3, se lahko zgodi, da ob združitvi zapisov p_1 in p_2 v enega, ob primerjavi s p_3, novi zapis doseže mejo, ki določa, ali se osebi ujemata. Torej je potrebno vsak novi zapis rekurzivno primerjati z vsemi ostalimi.

Kako se združita p_1 in p_2, lahko določimo tako, da tistega, ki vsebuje več informacij izberemo kot dominantnega. Če vsebujeta enako informacij, izberemo najaktualnejšega. Če kot dominantnega izberemo p_1, potem dopolnimo vse manjkajoče informacije s tistimi, ki jih morda vsebuje p_2, nato pa lahko p_2 zbrišemo.

Kompleksnejši primer bi bil ta, da bi ustvarili popolnoma nov zapis, ki bi bil nadrejen zapisoma p_1 in p_2.

2.6 Zapuščeni profili

Obstajajo primeri, ko si uporabnik socialnega omrežja ustvari profil, ki ga nato ne posodablja redno in nato raje ustvari nov profil z veliko podobnimi podatki, nekaj podatkov posodobi in ostale doda. V tem primeru pride do situacije, ko ima isti uporabnik več profilov. Če sistem za deduplikacijo ni implementiran, pride do dveh ločenih zapisov v podatkovni zbirki.

Problem se pojavi tudi v primerih, ko si uporabnik izdela profil zgolj s podatki, ki so nujno zahtevani za ustvarjanje profila in ga nato zapusti. Profil ostane shranjen, a se ne uporablja in vsebuje zelo malo podatkov. V profesionalnih družabnih omrežjih se ponavadi zahteva le ime, priimek in lokacija uporabnika. Skupina teh podatkov največkrat ne predstavlja dovoljšnje zanesljivosti, če jih primerjamo z drugim profilom, saj je znotraj ene države oziroma regije ponavadi veliko oseb s podobnim imenom in priimkom. Take profile je zato težje povezati z drugimi profili, ker ne moremo biti prepričani,

da gre za profile istih oseb.

2.7 Zastareli podatki o profilih

Težava pri primerjanju profilov je tudi starost podatkov o osebah. Če primerjamo star profil v bazi z novejšim, se lahko zgodi, da so podatki že tako različni, da ju algoritem ne bi zaznal kot duplikat, čeprav v resnici je. V primeru, da je skupnih podatkov še vedno dovolj, ju lahko vseeno uspešno dedupliciramo. Ponavadi uporabniki redko spreminjajo svoja imena, profilne slike in obstoječa delovna mesta. Delovna mesta običajno uporabniki samo dodajajo in tako njihova prejšnja vseeno ostanejo vpisana na profilu. Tako osebe še vedno lahko zanesljivo primerjamo po delovnih mestih, kakovost pa zvišujemo z ujemanji dodatnih skupnih atributov.

2.8 Podatki v različnih jezikih

Ista oseba lahko na družabnem omrežju opiše svoj profil v različnih jezikih, to je pogosto v tistih poljih, v katera lahko prosto vnašajo podatke.

Nekatera socialna omrežja podatke v poljih, kjer je nabor vrednosti vnaprej določen in predložen uporabniku, ki si nato izbere eno ali več vrednosti, osebi, ki si profil ogleduje, prevaja glede na lokacijo, kjer se ta oseba nahaja. Tako ni nujno, da dve osebi locirani v različnih jezikovnih območjih vidita iste podatke. Brez znanja več jezikov je zato z navadnim primerjanjem nizov težje opaziti, ali gre za isti profil.

Prevodi jezikov, ki jih je lastnik profila vpisal, so si še nekoliko podobni, tako bi se jih z algoritmi za primeranje nizov še dalo primerjati, ostale podatke, kot so na primer položaji v podjetjih, izobrazba, hobiji, pa bi morali prevajati.

Poglavje 3

Oseba

Podatke iz različnih profesionalnih omrežij (Linkedin, Xing in Viadeo) sem združil in vpisoval v svojo podatkovno bazo. V njej je tabela Oseba, v kateri vsak zapis predstavlja vse javne podatke pridobljene o osebi, registrirane na nekem profesionalnem družabnem omrežju. Vsaka oseba v podatkovni zbirki ima unikatno povezavo do socialnega omrežja, je pa možno, da dve razični povezavi vodita do profilov iste osebe. Zato so novejši podatki lahko le posodobljeni profili oseb, starejši profil pa je zato vseeno ostal v podatkovni bazi.

3.1 Predstavitev entitete Oseba

Oseba v podatkovni zbirki je predstavljena z naslednjimi atributi:

- polno ime,
- lokacija,
- URL do profila na družabnem omrežju
- profilna fotografija,
- osebne spletne strani,
- zanimanja (hobiji),

- izobrazba,
- jeziki, ki jih oseba zna,
- položaji v podjetjih, kjer je oseba bila ali je trenutno zaposlena

Vsi atributi so predstavljeni kot nizi znakov, razen profilne fotografije, kjer gre za sliko shranjeno v *.png* ali *.jpg* formatu. Čeprav so vsi drugi atributi predstavljeni z nizi, je za njihovo primerjanje potrebno izbrati več različnih pristopov, saj zgolj en pristop ne bi enako zagotavljal pravilnosti primerjanja.

3.2 Ime in priimek

Pričakovali bi, da sta to podatka, ki že z dovolj veliko verjetnostjo nakazujeta iste oz. različne osebe. Pa se izkaže, da sta sama po sebi neuporabna, saj tudi, ko resnično določata isti osebi, ni nujno, da sta zapisana v dveh enakih oblikah, kar je težava za programsko določanje duplikatov.

Če primerjamo dve osebi, ki jima je ime Miha, ne moremo z zagotovostjo trditi, da gre za isto osebo. Če pa imata osebi še enak priimek, se ta stopnja zaupanja hitro poveča, a še vedno nimamo dovolj podatkov, da bi z gotovostjo trdili, da smo našli duplikat. Pri primerjanju polnega imena osebe naletimo še na razne druge težave. Na primer tu ponavadi uporabniki na profesionalnih socialnih omrežjih večkrat dodajo svoje znanstvene nazive. Če je na enem profilu ta naziv znan, na drugem pa ne, se verjetnost, da gre za isto osebo, zmanjša. Težava je tudi v različnih oblikah istega imena, okrajšav srednjih imen in priimkov in seveda priimkov, ki se dodajo oz. spremenijo ob poroki.

Za primerjanje osebnih imen so bile izvedene raziskave [11], ki so ugotavljale, kateri algoritmi se najbolje obnesejo za to nalogo. Raziskovalci so se srečevali z istimi težavami, kot sem jih imel jaz, torej različnimi variacijami imen, vzdevkov, naknadnih sprememb osebnih imen. Zato to predstavlja bolj kompleksen primer v primerjavi z navadnim primerjanjem nizov. Pojavljajo se tudi napake, ki jih naredijo uporabniki, ko vpisujejo svoja imena v zahtevana polja. Po nesreči spremenijo vrstni red črk, dodajo ali izpustijo črke,

jih zamenjujejo, itd.

Dva prevladujoča pristopa k tem problemom sta primerjava fonetičnih kodiranj in primerjanje vzorcev v nizih. Fonetična kodiranja pretvarjajo nize v kodo glede na to, kako se ime izgovori. Ta proces je odvisen od jezika. Večina tehnik je bila primarno razvita za angleško govoreče. Najstarejši in najbolj znan algoritem fonetičnega kodiranja je **Soundex** [15]. Obdrži prvo črko niza ostale črke pa pretvori v številke glede na kodirno tabelo. Vse črke, ki so glede na tabelo zakodirane v 0, so nato odstranjene in zaporedne enake številke so okrajšane v eno številko. Končni rezultat je prvotna prva črka in tri številke (daljše kode so odrezane, prekratkim pa se dodajo ničle). Primer kodiranja za ime "peter"je "p360".

Ker pa imam v svoji zbirki osebe z imeni iz različnih svetovnih območij, sem se odločil, da primerjanje fonetičnih kodiranj ni primerno, saj fonetična kodiranja niso enotna po vsem svetu.

Zato sem primerjal podobnosti nizov oz. imen. Ponavadi se izračuna neka normalizirana podobnost, ki je predstavljena z vrednostmi od 0.0, kar pomeni, da sta si niza zelo različna, do 1.0, ki predstavlja enakost nizov. Ker sem v svojem primeru hotel to vrednost predstaviti kot razdaljo med dvema nizema, sem vrednosti obrnil, torej 0.0 pomeni enakost nizov, 1.0 pa nasprotje. Za računanje podobnosti nizov je na izbiro veliko tehnik in pristopov, kot so Levenshteinova razdalja, razdalja najdaljšega skupnega podniza (LCS) [11], algoritem Smith-Waterman [11], Jaro-Winklerjeva razdalja [11].

Na osnovi analize raziskave [11] sem se odločil, da je najbolj primerna Jaro-Winklerjeva razdalja, ki je najbolj optimizirana za primerjanje krajših nizov in pod te spadajo tudi osebna imena. Rezultat, ki ga vrne algoritem Jaro-Winkler, je predstavljen z vrednostmi od 0.0 (različna niza) do 1.0 (identična niza). Za potrebe svoje implementacije sem razdaljo izračunal po formuli:

$$d' = 1 - d$$

Manjša, kot je nova razdalja d', bolj sta si niza podobna.

Normalizacija nizev

Da so nizi že pred primerjanjem v čimbolj enotni obliki, jih je priporočljivo normalizirati in s tem zagotoviti boljše rezultate ujemanj.

Nize sem zato pri svojih algoritmih normaliziral z operacijami:

- pretvorba v male črke,
- ločevanje po praznih znakih,
- brisanje ločil,
- brisanje številk

Normalizirane nize sortiram po abecedi, da rešim problem, ko sta ime in priimek v dveh nizih zamenjana med sabo in s tem poskrbim za njuno medsebojno neodvisnost.

3.3 Zanimanja, izobrazba, lokacija, jeziki

Te vrednosti so predstavljene z naštetimi besedami, torej gre lahko za več vnosov splošnega besedila, ki je lahko v različnih jezikih, z napakami, ki jih stori uporabnik pri kreiranju oz. posodabljanju profila. Če jih ločimo, lahko nato za vsak atribut primerjamo množice elementov in se glede na število ujemanj odločimo, ali gre za dovolj veliko podobnost množic in s tem verjetnost, da gre za duplikat. Ti atributi lahko vsebujejo nič ali več vrednosti in poljubno besedilo, zato se zgolj z njimi ne da dovolj zanesljivo odločiti, ali gre za ujemanje ali ne. V kombinacijah z drugimi atributi pa zvišujejo verjetnost, da gre za isto osebo.

Na primer uporabniki imajo lahko jezike naštete tako:

profil 1	english, spanish, german
profil 2	german, spanish, englisch

Tabela 3.1: Primer naštetih jezikov uporabnika

Nize normaliziramo in ločimo tako, kot v prejšnjem primeru in naredimo presek množic iz obeh profilov. Elemente obeh množic primerjamo med sabo vsakega z vsakim le enkrat. Če algoritem Jaro-Winkler, ki sem ga izbral tudi za primerjanje teh nizov, vrne dovolj veliko verjetnost, da se dva jezika ujemata, za vsako ujemanje zvišamo verjetnost, da so vrednosti v poljih enake. Nato rezultat povprečimo glede na število ujemanj in številom elementov, ki smo jih primerjali.

3.4 URL povezava do profesionalnega socialnega omrežja

Povezava do profesionalnega družabnega omrežja pri primerjanju oseb med različnimi omrežji ni toliko pomembna, saj se URL povezave med njimi precej razlikujejo in zato ne moremo primerjati podobnosti nizov. Lahko pa jih primerjamo v primeru, ko gre za osebe znotraj istih omrežij.

Za dedupliciranje znotraj omrežja lahko ignoriramo shemo URL-ja (http/https) in poddomene, saj je glavna informacija, ki razlikuje dve osebi v URL poti. Zato lahko to informacijo izvlečemo v nov atribut entitete Oseba, ki ga nato definiramo in opišemo v podatkovnem modelu, ki ga uporabljamo za deduplikacijo. Če sta URL poti dveh oseb enaki, nam to že zagotavlja, da naj bosta tudi osebi isti objekt. Če sta si URL poti zgolj podobni, lahko ti osebi vseeno pogrupiramo in nato dodatno preverimo vrednosti drugih atributov in, če se dovolj vrednosti ujema, označimo kot duplikat.

Protokol	Ime gostitelja	Pot
http	www.linkedin.com	profile/janez-novak-687594
https	si.linkedin.com	profile/janez-novak-687594

Iz primera 3.4 vidimo, da je pot obeh povezav do profesionalnega družabnega omrežja enaka in zato zagotovimo uspešno zaznan duplikat.

3.5 Osebne spletne strani

Čeprav so tudi spletne strani povezave URL, tako kot URL-ji profesionalnih omrežjih, jih za razliko od njih lahko uporabimo za primerjanje oseb med različnimi omrežji. Pri povezavah lahko upoštevamo samo glavno domeno, saj je dovolj velika verjetnost, da poleg ostalih ujemanj, tudi ujemanja domen nakazujejo isto osebo. Ker ima lahko oseba na družabnem omrežju vpisanih več spletnih strani, primerjamo podmnožici spletnih strani. Velikost preseka množic spletnih strani določa podobnost vrednosti tega atributa.

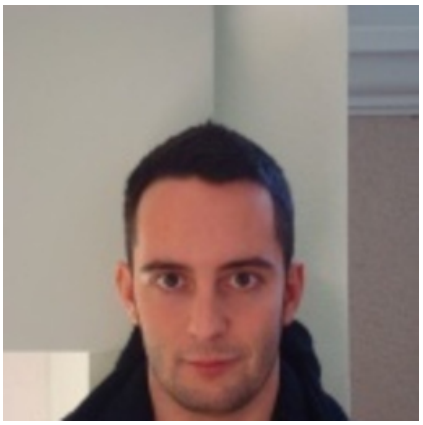
3.6 Profilne fotografije

Na družabnih omrežjih, ki sem jih obravnaval, si uporabnik lahko nastavi največ eno profilno sliko. Fotografija ni zahtevan podatek na nobenem izmed obravnavanih socialnih omrežij. Največkrat je to osebna fotografija, kjer se vidi obraz človeka. Človek za razliko od računalnika iz različnih fotografij veliko bolj zanesljivo razbere, ali je na njih ista oseba. Predvsem lahko hitreje prepozna isto osebo, fotografirano iz različnih kotov, v različnih okoljih, tako svetlobnih, scenskih in barvnih. Ker pa so fotografije pomemben podatek, s katerim lahko določimo, ali gre za iste osebe, jim v primeru ujemanja določimo visoko utež.

Normalizacija profilnih fotografij

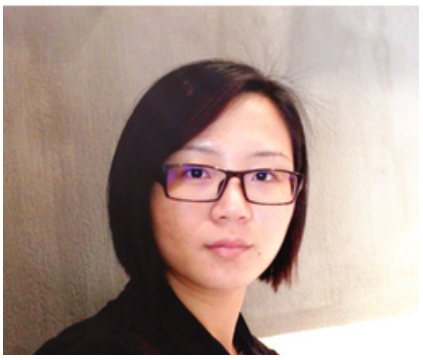
Ker socialna omrežja naložene fotografije uporabnikov obdelajo (velikost, kodiranje, resolucija), spremenijo imena datotek, njihove velikosti, kontrolne vsote, barve, si lahko pomagamo le s primerjanjem samih karakteristik slike. Zaradi ignoriranja barvnih razlik, ki nastanjejo zaradi morebitnih različnih barvnih profilov, fotografije pred primerjanjem najprej pretvorimo v sivinske slike. V mojem primeru je na vsakem izmed treh družabnih omrežjih širina slike enaka višini. Tako lahko slikam spreminjamo velikost na enotne vrednosti. S tem olajšamo primerjanje matrik pikslov.

Slika 3.1: Mikloš 1



Slika 3.2: Mikloš 2

Slika 3.3: Rainbow 1



Slika 3.4: Rainbow 2

V naslednjih podpoglavjih so predstavljeni algoritmi, s katerimi lahko zanesljivost in hitrost primerjanja fotografij približamo človekovim.

3.6.1 Zgoščevalna funkcija

Najlažja primerjava dveh profilnih fotografij iz omrežij bi bila, če bi bili profilni fotografiji na omrežjih enaki, saj bi si lahko pomagal z zgoščevalno funkcijo. Slike normaliziramo na enotno velikost. S tem odstranimo vse

visoke frekvence (vrednosti sosednjih pikslov). Ker bo velikost enotna, skaliranje slike ne bo vplivalo na zgoščevalno vrednost. Za tem primerjamo sosednje piksle v vsaki vrstici slike in na podlagi njihovih ujemanj v nov seznam zapišemo logične vrednosti, ki predstavljajo ujemanje ali neujemanje vrednosti pikslov. Da bi bila zgoščevalna funkcija preprosta za uporabo in morebitno shranjevanje v bazo, njen rezultat pretvorimo v šestnajstiški zapis. Po končanem postopku lahko primerjamo zapise slik in v primeru enakih zgoščevalnih vrednosti ugotovimo, ali gre za duplicirane slike.

Prednosti tega algoritma sta predvsem neodvisnost od objektov na slikah in enostavna implementacija.

Slabosti je v našem konkretnem primerjanju profilnih fotografij več, ker algoritem deluje le v primeru, ko slike niso transformirane, saj že rotacija ene slike ali njeno zrcaljenje rezultira v drugačni zgoščevalni vrednosti. Potrebovali bi tudi prostor, kamor bi te vrednosti shranjevali, saj bi bilo računanje zgoščevalne vrednosti za vse slike, s katerimi želimo določeno sliko primerjati, časovno neučinkovito. V primeru, da ima ista oseba na dveh omrežjih različni fotografiji, ta algoritem odpove.

3.6.2 Prepoznavanje obrazov

Detekcija obrazov

Informacije, kot so barve, svetloba in okolje okoli osebe pri odkrivanju obrazov na fotografiji, niso pomembne. Zato fotografije pretvorimo v sivinske slike in vsem nastavimo enako velikost.

Prepoznavanje obrazov s pomočjo algoritma SIFT

V primeru, kjer je na fotografijah največkrat obraz, lahko uporabimo algoritme za prepoznavanje enakih obrazov. Metodo za detekcijo obrazov lahko uporabimo tudi pri računanju podobnosti z algoritmom SIFT [12], saj se s tem že pred primerjanjem znebimo nepomembnih informacij iz okolice obraza in se osredotočimo le na glavni subjekt na sliki, ki je obraz.

Algorithm 1 Zgoščevalna funkcija

```
1: function DHASH(image)
2:     image ← GRAYSCALE(image)
3:     image ← RESIZE(image)
4:     width, height ← SIZE(image)
5:     image ← RESIZE(image)
6:     difference ← ARRAY()
7:     for row in RANGE(width) do
8:         for col in RANGE(height) do
9:             pixelLeft ← GETPIXEL(col, row)
10:            pixelRight ← GETPIXEL(col + 1, row)
11:            difference ← APPEND(pixelLeft > pixelRight)
12:        end for
13:    end for
14:    hash ← HEXADECIMAL(difference)
15:    return hash
16: end function
```

Da bi bili ti algoritmi uporabni, potrebujemo zbirko obraznih slik, a je v tem primeru nimamo. Nato se za neznano obrazno sliko vprašamo, kateri osebi ta obrazna slika pripada? Za reševanje tega problema je bilo predlaganih veliko algoritmov npr. množica lastnih vektorjev za prepoznavanje obraza (Eigenfaces), Fischerfaces, SIFT. SIFT je v primerjavi z drugima učinkovitejši oz. izboljšan [12].

SIFT značilke so značilke izvlečene iz slik z namenom, da pomagajo pri zanesljivem ujemanju različnih pogledov istega objekta, v tem primeru obraza. Z njimi rešimo prejšnjo odpoved algoritma zgoščevalnih funkcij pri različnih velikostih in orientacijah slik, saj so neodvisne od teh transformacij in se med različnimi slikami zelo razlikujejo. Izvlečene so v štirih korakih: Najprej izračunamo lokacije točk zanimanja z odkrivanjem največjih in najmanjših vrednosti množice Difference of Gaussian (DoG) filtrov, uporabljenih pri različnih velikostih po vseh sliki. Nato izpilimo lokacije z neupoštevanjem točk na nizkih kontrastih. Smer je nato določena vsaki ključni točki na podlagi lokalnih značilk slike. Na koncu izračunamo deskriptor lokalnih značilk za vsako ključno točko. Vsaka značilka je vektor dimenzije 128, ki identificira okolico vsake ključne točke.

V konkretnem primeru se SIFT značilke izvlečejo iz vseh obrazov na fotografijah v bazi. Nato lahko značilke vsake slike primerjamo z značilkami ostalih slik. Značilki se ujemata, če je razdalja do druge značilke manjša od podanega praga razdalje do druge najbližje značilke. To zagotavlja zmanjšanje števila negativnih ujemanj, kjer bi prepoznali duplikat, kjer ga v resnici ni.

Na sliki 3.5 so prikazane ključne točke, njihova velikost in smer orientacije.

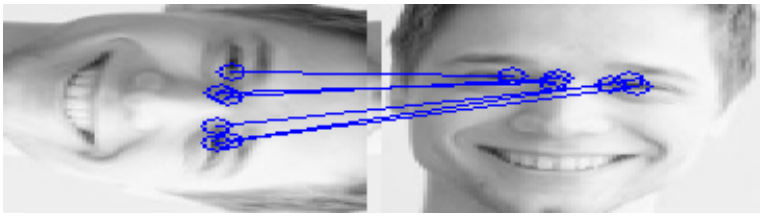
V teoriji bi torej morali za enaka obraza, kjer sta velikosti in/ali orientaciji slik obraza različni, dobiti 100% ujemanje. Izmed šestih ujemanj, ki so predstavljene z modrimi daljicami na sliki 3.6, so bila vse razdalje dovolj majhne, da so bila vsa ujemanja dobra, torej 100% ujemanje.

Slika 3.7 prikazuje najboljših 10 ujemanj med dvema obrazema. Izmed 24 zaznanih ujemanj, je bilo 11 takšnih, ki so bila dobra, kar pomeni, da je

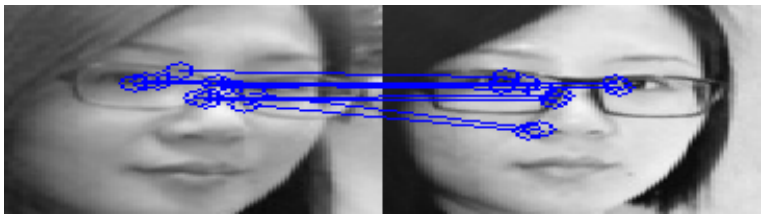
Algorithm 2 Primerjava slik s pomočjo SIFT

```
1: function COMPARE(firstImage, secondImage)
2:     firstImage ← GRAYSCALE(firstImage)
3:     secondImage ← GRAYSCALE(secondImage)
4:     sift ← SIFT()
5:     keypoints1, descriptors1 ← SIFTDETECTANDCOMPUTE(firstImage)
6:     keypoints2, descriptors2 ← SIFTDETECTANDCOMPUTE(secondImage)
7:     bf ← BFMATCHER()
8:     matches ← BFKNNMATCH(descriptors1, descriptors2)
9:     goodMatches ← ARRAY()
10:    for first, second in matches do
11:        if DISTANCE(first) ≤ 0.75 * DISTANCE(second) then
12:            goodMatches ← APPEND(first)
13:        end if
14:    end for
15:    return goodMatches
16: end function
```

Slika 3.5: SIFT ključne točke.



Slika 3.6: Primer ujemanj tranformiranih slik.



Slika 3.7: Primer ujemanj dveh slik osebe.

Slika 1	Slika 2	Razdalja
Miklošˇ 1	Miklošˇ 2	0.0
Miklošˇ 1	Rainbow 1	0.777
Miklošˇ 1	Rainbow 2	0.714
Miklošˇ 2	Rainbow 1	0.0
Miklošˇ 2	Rainbow 2	0.5
Rainbow 1	Rainbow 2	0.542

Tabela 3.2: Razdalje izračunane s SIFT

bila njihova razdalja manj kot 0.75.

Iz tabele 3.2 lahko razberemo, da je algoritem pravilno zaznal osebo Miklošˇ pri obeh njegovih fotografijah, a je narobe zaznal duplikat slik Miklošˇ 2 in Rainbow 1 ter slabše ocenil ujemanje Rainbow 1 in Rainbow 2, ker je razdalja precej večja.

3.6.3 Openface

Ker z rezultati algoritma SIFT, prikazanimi v tabeli 3.2 nisem bil zadovoljen, saj so bile razdalje med istimi osebami prevelike in med različnimi osebami premajhne, sem za prepoznavanje istih oseb iz profilnih slik, uporabil knjižnico Openface [16], ki temelji na implementaciji sistema FaceNet [17] - prepoznavanja obrazov z globokimi nevronskimi mrežami, ki se neposredno uči povezovanja obraznih slik v kompaktni evklidski prostor, kjer razdalje neposredno pripadajo meritvam podobnosti obrazov. Ko je tak prostor iz-

delan, se naloge, kot so prepoznavanje obrazov, preverjanje in razvrščanje v skupine lahko implementirajo z uporabo standardnih tehnik FaceNet-ovih vdelovanj kot značilk.

Rezultati primerjanja obrazov so predstavljeni tako, da manjše vrednosti kvadratnih evklidskih razdalj predstavljajo bolj podobne medsebojne obraze v primerjavi z večjimi razdaljami. Tako lahko z nastavljanjem pragu te razdalje določimo ali rezultati razdalj pod tem pragom predstavljajo enake osebe ali ne.

Openface na vhodni fotografiji zazna obraz, ki ga nato transformira za namen nevronskih mrež tako, da poskuša oči in spodnjo ustnico poravnati na enak položaj na vsaki sliki.

Z uporabo globokih nevronskih mrež predstavimo obraz na 128-dimenzijski hipersferi. Predstavitev je generična za katerikoli obraz. Daljša razdalja med dvema reprezentacijama pomeni obraze različnih oseb. Ta lastnost olajša gručanje, prepoznavanje podobnosti in klasifikacijske naloge v primerjavi z drugimi tehnikami prepoznav obrazov, kjer evklidska razdalja med značilkami nima pomena.

Algorithm 3 Primerjava slik s pomočjo Openface

```
1: function OPENFACE(image)
2:     image ← DETECTFACE(image)
3:     image ← TRANSFORMFACE(image)
4:     image ← CROPTOFACE(image)
5:     PUTINDEEPNEURALNETWORK(image)
6:     result ← REPRESENTATE(image)
7: end function
```

Nato lahko primerjamo različne obrazne slike in izvemo razdaljo med njimi. Tiste slike, ki imajo med seboj razdaljo, manjšo od nastavljenega praga, so prepoznane kot ista oseba.

Kot lahko vidimo v tabeli 3.3 je Openface nalogo primerjanja obrazov opravil precej bolj zanesljivo, saj je pravilno ocenil majhno razdaljo v primeru različnih slik iste osebe, drugim kombinacijam pa pripisal večjo razdaljo. Iz

Slika 1	Slika 2	Razdalja
Rainbow 1	Rainbow 2	0.135
Rainbow 1	Miklo$\check{s}$ 2	1.043
Rainbow 1	Miklo$\check{s}$ 1	0.820
Rainbow 2	Miklo$\check{s}$ 2	0.970
Rainbow 2	Miklo$\check{s}$ 1	1.003
Miklo$\check{s}$ 2	Miklo$\check{s}$ 1	0.232

Tabela 3.3: Razdalje izračunane z Openface

tega vzorca fotografij bi lahko prag, ki določa isti osebi, nastavili na okoli 0.3. Čeprav je algoritem nekoliko počasnejši v primerjavi s SIFT-ovim, mi je veliko bolj pomembna pravilnost računanja razdalj med istimi osebami. Če primerjamo isti fotografiji, je izračunana razdalja enaka 0.0.

3.6.4 Čiščenje fotografij

Privzetih fotografij, ki jih socialno omrežje nastavi v primeru, ko si uporabnik ne naloži profilne fotografije, ne želim upoštevati, zato jih tudi ne vključim v izbor fotografij, ki jih primerjam, saj bi zaznani duplikati slik zagotovili veliko verjetnost oz. utež, da gre za isto osebo.

Če v obeh slikah prepoznamo obraz človeka, lahko za njuno primerjavo uporabim primerjanje enakih obrazov.

V primeru, da vsaj na eni sliki obraza ni zaznanega, še vedno lahko preverim, če sta si sliki podobni vsaj po zgoščevalni funkciji.

3.7 Položaji v podjetjih

Na vseh profesionalnih družabnih omrežjih lastnik profila oz. uporabnik, opiše svoja delovna mesta s položajem in podjetjem, v katerem je bil oziroma je zaposlen. Ko primerjamo delovna mesta dveh oseb, vsako delovno mesto v istem podjetju, ki se ujema, zvišuje verjetnost, da gre dejansko za isto osebo.

Med primerjanjem dveh profilov, lahko pride do primera, ko ima en profil manjkajoča delovna mesta, pa to ne pomeni, da sta profila zagotovo od dveh različnih oseb. Lahko je prišlo do napake pri pridobivanju podatkov, ali pa ima en profil bolj aktualne podatke oziroma so bili pri enem od profilov mogoče podatki izbrisani.

Vrednosti vsakega položaja in podjetja sta neposredno povezani in ju moramo pri primerjanju upoštevati skupaj. Če sta obe osebi, ki ju primerjamo, direktorja, s tem težko določimo enakost. Če pa sta dva zapisa direktorja v istem podjetju, se verjetnost enakosti poveča.

Ker se ujemajoča delovna mesta pomemben del, ki z večjo verjetnostjo pove, da gre za isto osebo, je tej primerjavi potrebno posvetiti več oz. največ pozornosti.

Rezultat podobnosti izračunamo kot razmerje ujemajočih se delovnih mest z vsemi delovnimi mesti obeh oseb, ki jih primerjamo. Ujemanje je boljše, če se ujema več delovnih mest glede na vsoto vseh različnih delovnih mest.

Poglavje 4

Optimizacija in izboljšava primerjanja

Glede na atribut, ki ga želimo primerjati med obema osebama, je potrebno izbrati čim boljše algoritme, s katerimi bomo izmerili podobnosti med vrednostmi v atributih.

4.1 Kombiniranje primerjalnih metod

Lahko bi računali podobnosti enakih atributov Osebe in vrednosti vstavili v vektor dimenzije n, kjer n predstavlja število atributov, ki jih primerjamo. Vsako podobnost atributa lahko predstavimo z logično vrednostjo, kjer 1 predstavlja ujemanje, 0 pa različni vrednosti. Še točneje pa je, da logične vrednosti nadomestimo z rezultati, ki jih vrnejo funkcije, ki primerjajo vrednosti atributov. Vrednosti so predstavljene s številom med 0.0 in 1.0, kjer 0.0 predstavlja isto vrednost (najmanjšo razdaljo), 1.0 pa dve čisto različni vrednosti. Če katera funkcija vrača podobnost na drugačen način, jo je potrebno normalizirati, da so vse podobnosti predstavljene enako. Vendar vsaka podrobnost ni enako pomembna. Na primer enako ime in priimek predstavljata večjo verjetnost enakih oseb, kot npr. enaka lokacija. Zato podobnosti v vektorju utežimo in tako, nastavljamo pomembnosti vsake podobnosti atri-

butov. Večja utež predstavlja večjo pomembnost pri primerjanju atributov. Če sta si vrednosti atributov podobni in imata veliko utež, pomeni, da je verjetnost za isto osebo večja, kot pa v primeru, ki sta si dva atributa podobna in imata majhno utež. Največja težava pri tej metodi je pravilna izbira in kombinacija uteži. Za določitev te meje je potrebno znanje, ki ga pridobimo s poskušanjem oz. eksperimentiranjem in sprotnim učenjem, kjer se naučimo, katera je tista prava meja. Nepravilno izbrana meja vodi do napačno določenih ujemajočih oseb ali k zmanjšanju števila duplikatov, ki bi morali biti združeni v eno osebo. Na koncu izračunamo vsoto uteženih enotskih vektorjev, ki dajo vektor razdalje med dvema osebama. Na podlagi te vsote se nato odločimo, ali je velikost tega vektorja dovolj majhna. Manjša, kot je, višja je verjetnost enakih oseb, ki jih nato označimo za dupliciran par. Tudi ta prag vrednosti je težko ovrednotiti. Lahko pa kombiniramo pravila, ki so se v konkretnem primeru dobro obnesla, saj lahko sami določamo odvisnosti podobnosti atributov in s tem tudi vrstni red primerjanja atributov, saj jih lahko verižimo. Najprej sem dal največjo težo imenom oseb, nato pa delovnim mestom v podjetjih. Z vsakim nadaljnjim korakom smo bolj prepričani, da gre za isti osebi.

4.2 Sestavljanje blokov

Ker bi bilo računanje podobnosti za vsak par oseb preveč zahtevno, se postopki, ki temeljijo na primerjanju podobnosti v glavnem osredotočajo na zmanjšanje števila parov za vrednotenje.

Tako je bolje že pred dejanskim začetkom primerjanja podobne zapise združiti v skupine. Za njihovo izdelavo, v katerih bi bili zapisi, ki so si med seboj najbolj podobni je na izbiro več možnosti. Na voljo je veliko pogojev, ki bi določali, v katero skupino bi se uvrstil določen zapis v bazi. Če definiramo skupine podatkov, ki si delijo neke skupne vrednosti in primerjamo samo zapise znotraj te skupine, potem lahko občutno zmanjšamo število primerjav, ki jih je potrebno narediti. Če so te skupine dobro definirane, potem lahko

zmanjšamo število primerjav in obdržimo zaupanje, da se duplikati nahajajo znotraj vsake skupine. V primeru, da pogoja za tako skupino ne moremo definirati, lahko skonstruiramo več različnih skupin, pri čemer je vsaka skupina definirana z različnim pogojem. V konkretnem primeru, bi lahko bil za prve gruče prvi pogoj prve 3 črke v imenu, nato bi definirali sekundarne gruče s spolom osebe, v tretji skupini bi bile gruče razdeljene po lokaciji osebe. Tako pridemo do primera, ko bi bilo v vsaki gruči čim manj oseb, ki bi se med sabo primerjale, posledično bi zmanjšali časovno zahtevnost.

Objekte, ki jih bomo primerjali med seboj, je potrebno uvrstiti v skupine najbolj podobnih. Časovno neučinkovito je primerjati moške z ženskami in obratno. Z bazo imen lahko pogrupiramo objekte na tri skupine: moške, ženske in tiste, ki so v obeh skupinah ali ne obstajajo v bazi imen. Tista imena oseb v bazi, ki so v obeh skupinah, damo k moškim in ženskim predstavnikom. Nato lahko primerjamo objekte znotraj ene skupine in moško in žensko skupino še s tisto nevtralno.

Glede na to, da je oseb in njihovih podatkov veliko, je potrebno optimizirati primerjanje le-teh. Če bi primerjali 100 000 oseb med sabo, in bi ena primerjava trajala 1s, bi za 49 995 000 primerjav (vsak z vsakim) porabili cca. 1.5 let. Resnično je sicer čas ene primerjave občutno nižji od ene sekunde, ampak več kot očitno je, da je časovna zahtevnost prevelika.

Med n osebami je

$$n * (n - 1)/2 \tag{4.1}$$

različnih parov in izmed njih je malo takšnih, ki so res duplikati.

Poleg časovne zahtevnosti je z večanjem podatkovne zbirke podatkov tudi več "umazanih", nepopolnih podatkov.

Če imamo več različnih blokov, v katere zberemo podobne elemente in proces dedupliciranja poženemo v večih iteracijah čez te različne bloke, ki predstavljajo različne kose podatkov, zmanjšamo verjetnost, da bi se zapisi v bazi, ki so si podobni v določeni lastnosti, ne bi vsaj enkrat srečali v istem bloku.

Da bi povečali število podobnih zapisov, ki bi se ujemali, bi lahko omilili

pogoj, ki združuje zapise v gruče. To bi povečalo zahtevnost, a ni nujno, da bi s to opcijo drastično povečali število ujemanj v gruči. Alternativa je implementacija več neodvisnih sortiranj v gruče, vsakič z drugimi pogoji, ki bi določali, kako se zapisi dodajajo v njih. Na primer, v eni iteraciji bi lahko za glavni ključ uporabili vredost lokacije, v naslednji pa imena oseb, ki bi določala skupino zapisov.

Poglavje 5

Dedupliciranje s pomočjo strojnega učenja

Naprednejša tehnika odkrivanja duplikatov je z uporabo strojnega učenja, pri katerem naučimo klasifikator, ki bo razločeval med dupliciranimi in neduplici-ranimi pari. Vsak par oseb, ki jih primerjamo je predstavljen kot vektor značilk, ki je sestavljen iz enotskih vektorjev za vsako dimenzijo pomnoženih z velikostjo podobnosti med atributi in pripadajočo utežjo. Tehnike, ki temeljijo na učenju zahtevajo učno množico za treniranje klasifikatorja. Učna množica vsebuje pozitivne in negativne vektorje značilk, ki predstavljajo podobne ali različne pare. Nato s tem klasifikatorjem označimo nov primerek parov, ki ju primerjamo, kot duplikat ali ne.

Ker je velika večina primerjanih parov med sabo različna, se pojavi problem pri generiranju učne množice.

Za evaluiranje kvalitete rezultatov uporabim dve metriki:

- **Natančnost** je procent pravilno identificiranih parov izmed vseh, ki so bili označeni kot duplikati.

- **Priklic** je procent pravilno identificiranih izmed vseh parov.

5.1 SVM

Najprej izračunam vektor značilk za vsak par zapisov. Za objekt Osebe sem izbral algoritem Jaro-Winkler za računanje podobnosti in izračunal njegovo vrednost na n atributih Osebe. Nato je potrebno naučiti klasifikator na nekaj parih, ki so bili naključno izbrani izmed vseh.

Problemi pri uporabi strojnega učenja je generacija zbirke za treniranje, saj je v realnem svetu veliko več neujemanj oz. unikatnih oseb, kot pa duplikatov. Dodatna težava pri strojnem učenju je ta, da v konkretnem primeru zapisi niso označeni in posledično ne obstaja zbirka učnih primerov, kjer bi bili definirani duplicirani pari in unikatni pari oseb.

Če se želimo izogniti generaciji zbirke, na kateri bomo trenirali, lahko uporabimo nenadzorovane/delno-nadzorovane tehnike.

5.2 Dedupe

Za shranjevanje podatkovne zbirke sem uporabljal PostgreSql. V bazi vsaka zapis v tabeli predstavlja eno neduplicirano osebo. Dedupe [13] zahteva, da je vsaka manjkajoča vrednost atributa osebe označena kot $NULL$. Najprej sem sestavil slovar vseh oseb in njihovih podatkov v zbirki, ki jih želim deduplicirati. Potem preverim, če obstaja datoteka z nastavitvami. V tej datoteki so shranjeni podatki o definiranem podatkovnem modelu, klasifikatorju, predikatih in besede, ki jih ignoriram, ker ne predstavljajo pomembnih informacij (angl. “stop words”). Če ta datoteka obstaja, jo lahko ob ponovnem zagonu deduplikacije na strukturiranih podatkih uporabim in s tem preskočim definiranje podatkovnega modela ter takoj instanciram Dedupe objekt, v katerega naložim že definiran podatkovni model, klasifikator, predikate in stop word-e. Če ta datoteka ne obstaja, pomeni, da je treba definirati nov podatkovni model. Ta objekt ustvarim na novo in mu podam opisana polja, ki pridejo v poštev pri deduplikaciji. Vsako polje je opisano z imenom tega polja, načinom primerjanja in parametrom, ki opisuje, če lahko podatek za to polje manjka.

Nato izberem vzorčno množico oseb, ki jih uporabim za učenje. Ta množica je zbirka naključnih in podobnih parov oseb v bazi. Za velikost sem izbral 75 000 oseb izmed približno 1 800 000 objektov.

Učenje

Učenje poteka tako, da Dedupe najde par oseb, za katerega je najmanj prepričan, da gre za isti osebi in vpraša uporabnika, ali sta to duplikata ali ne. Uporabnik ima na izbiro odgovore **Da**, **Ne** in **Ne vem**, s katerim sporoči, da ni prepričan o enakosti ponujenih oseb, verjetno zaradi pomanjklivih podatkov iz katerih ni jasno, ali gre za isti osebi ali ne. Ko uporabnik potrdi izbiro, se mu ob naslednji iteraciji oz. vprašanju ponudi naslednji par oseb, za katere je program najmanj prepričan. Uporabnik postopek ponavlja, dokler se ne odloči, da ga zaključi. Minimalna meja, ki določa, kdaj lahko učenje zaključimo, je vsaj 10 potrjenih duplikatov in 10 potrjenih različnih parov oseb. Seveda je bolje iterirati čez čimveč parov in tako izboljšati kvaliteto učenja in predikatov, po katerih potem izberemo gruče, v katere se uvrstijo podobne osebe.

Predikat je funkcija, ki za določen atribut Osebe vrne množico značilk. Te značilke so lahko npr. “prve 3 črke vrednosti polja”, “vsaka beseda v polju”, itd. Objekti, ki si delijo enake značilke, postanejo del istega bloka.

Ko sem s postopkom označevanja podatatkov končal, lahko začnem z učenjem. Najprej določim vrednost dveh argumentov. PPC omejuje delež pokritih parov, ki ga dovolim predikatom pokriti. Če predikat zbere skupaj večji odstotek možnih parov, kot je nastavljen prag s PPC parametrom, ta predikat ne bo upoštevan. Z večanjem števila podatkov je zaželjeno parameter *ppc* manjšati. Ker je tu podatkov veliko, sem ta parameter nastavil na 0.001, torej 0.1 odstotka. Na koncu vse naučene podatke zapišem v datoteke z naučenimi podatki in nastavitvami.

Nato se lahko začne postopek zbiranja podobnih kontaktov v bloke. Kreiral sem tabelo *blocking_map*, kjer bo vsak zapis v vrstici predstavljal ključ bloka in ID osebe. Ključ bloka se za vsako osebo generira na podlagi vseh

atribut	Oseba 1	Oseba 2
polno ime	baker john	baker john
lokacija	Seymour, Texas	Twyford, Berkshire, United Kingdom
zanimanja	/	/
spletne strani	http://www.seymour-isd.net	http://www.resourcing-solutions.com/
URL	https://www.linkedin.com/pub/john-baker/46/1b3/117	https://www.linkedin.com/in/johnbakerrsl
delovna mesta	Elementary School Principal at Seymour ISD	Principal Consultant / Account Manager at Selby Jennings
izobrazba	/	/
jeziki	/	french

Tabela 5.1: Primer podobnih oseb, ki jih Dedupe ponudi za evaluiranje.

atribut	Oseba 1	Oseba 2
polno ime	elina kushchova	elina kushchova
lokacija	Slovenia	/
zanimanja	/	/
spletne strani	/	/
URL	https://si.linkedin.com/in/elina-kushchova-93278a69	http://si.linkedin.com/pub/elina-kushchova/59/571/740
delovna mesta	marketing at Ambient Hotel	marketing at Ambient hotel Domžale
izobrazba	International Business at University of Ljubljana, Faculty of economics	/
jeziki	/	/

Tabela 5.2: Primer istih oseb, ki jih Dedupe ponudi za evaluiranje.

predikatov. Ker je izbranih predikatov lahko več, oseba lahko ustreza več blokom, v katerih se zbirajo podobne osebe.

Generiranje ključev

Teh kombiniranih predikatov je lahko več in posledično tudi več blokov. Primer izbranega predikata je kombinacija prvih treh črk za polno ime in prvih sedmih črk za atribut osebnih spletnih strani. Tako bi bili ključi blokov za osebo z imenom "Primož Kerin" in spletno stranjo "http://www.primozkerin.com" naslednji:

- ker:http://:0

Ker je ime sortirano po abecedi, so v tem primeru kot prve tri črke izbrane črke iz priimka. Zadnja :0 pomeni indeks kombinacijskega predikata, saj je v tem primeru le eden.

V primeru, da bi obstajal še en kombinacijski predikat, npr. prvih pet črk imena in točno ista osebna spletna stran, bi bili ključi za konkretno osebo:

- ker:http://:0

Algorithm 4 Uvrščanje oseb v bloke

```
1: function BLOCKER
2:     for oseba in osebe do
3:         osebaID ← oseba.ID
4:         for predikatID, predikat in predikati do
5:             ključiBloka ← PREDIKAT(oseba)
6:             for ključBloka in ključiBloka do
7:                 yield ključBloka+predikatID, osebaID
8:             end for
9:         end for
10:    end for
11: end function
```

- kerin:http://www.primozkerin.com:1

Tako se za vsako osebo generirajo ključi in vsak ključ v kombinacijo z ID-jem osebe predstavlja en zapis v tabeli *blocking_map*.

Zaradi velikega števila oseb bi bilo v moji implementaciji nalaganje vseh podatkov v delovni pomnilnik in nato pisanje v novo tabelo prezahtevno, zato generirane podatke zapišem v začasno datoteko formata CSV, ki jo nato lahko uporabim za manj potraten vmesni člen pri zapisovanju podatkov v tabelo. Ker bo veliko blokov vsebovalo le eno osebo, jih lahko ignoriramo pri začetni fazi deduplikacije. Za tem lahko preverimo, katere osebe se nahajajo v blokih z istim ključem in sestavimo gruče kontaktov, za katere se predpostavi, da so v primeru višjega rezultata od nastavljenega oz. izračunanega pragu, ista oseba. Glede na ta prag se generira seznam gruč, v katerih so podani ID-ji istih oseb in rezultat verjetnosti, da gre res za isto osebo v gruči oseb.

Izbira pragu

Dedupe lahko napove verjetnost, da je par oseb duplikat. Za to lahko uporabimo natančnost in priklic. Vedno je med njima potrebno skleniti nek kompromis. Če vemo, koliko nam pomeni natančnost v primerjavi s prikli-

cem ali obratno, lahko definiramo oceno F, ki pomaga najti pravi prag za odločanje, kdaj so objekti duplikati. Ta prag mora biti optimalen za naše prioritete.

Če bi imeli podatke označene in bi vedeli, kateri so pravi duplikati, bi lahko našli prag z izračunom prave natančnosti in priklica za te podatke. Ampak dobro izračunan prag bi dobili le, če bi označeni primeri bili reprezentativni glede na nabor podatkov, ki jih skušamo klasificirati.

Tukaj nastane problem, saj označeni podatki, ki smo jih označili v procesu aktivnega učenja, niso reprezentativni. V tem procesu nismo poskušali najti najbolj reprezentativnih primerov, ampak smo iskali tiste, ki bi nas najbolj naučili, torej tiste, ki jih izbere Dedupe, ker je o njih najmanj prepričan.

Uporabljen pristop je uporaba naključnega vzorca blokov podatkov in nato izračun verjetnosti, da bosta dve ali več osebi znotraj vsakega bloka duplicirani. Iz teh verjetnosti lahko izračunam pričakovano število dupliciranih in različnih parov in nato izračunam pričakovano natančnost in priklic.

Dodatno preverjanje

Tu bi se proces zaznave dupliciranih oseb lahko zaključil, a ker želimo res čim večjo zanesljivost in nič napačnih ujemanj, sem implementiral še metode, ki iterirajo po gručah in dodatno preverjajo identičnost oseb znotraj njih. Med testiranjem in pregledovanjem ustvarjenih gruč sem opazil, da se ljudje najbolj razlikujejo po delovnih mestih v podjetjih. Če se pri dveh osebah, ki ju dodatno preverjamo ujema vsaj eno delovno mesto v istem podjetju, je to že dovolj velika verjetnost, da gre res za isto osebo. Več ujemajočih se delovnih mest v podjetjih le še zviša verjetnost. Tiste pare, pri katerih se nobeno delovno mesto ne ujema, je potrebno ročno preveriti, saj obstaja možnost, da za eno osebo delovna mesta niso vpisana in tako algoritem ni mogel zagotoviti, da gre za isto osebo. Torej nič ujemanj še ne pomeni, da sta osebi res različni. Ker smo z deduplikacijo in dodatnim preverjanjem veliko število pravih oseb že združili, takih gruč, v katerih so osebe, ki jih je potrebno ročno preveriti, ostane zelo majhno število.

Ko osebe v gručah združimo, lahko osebo, v katero so se vse ostale združile in je torej skupek podatkov vseh teh oseb, poskusimo znova preveriti, če se ujema še s kakšnim ključem katerega bloka, saj sedaj vsebuje več podatkov in bi ob ponovni iteraciji blokiranja morda padla v nov blok, kjer je možnost, da zanjo zopet najdemo nove duplikate.

Ker je učenje aktivno, ob vsakem svežem zagonu deduplikacije brez uporabe obstoječih nastavitev in natreniranih podatkih, treniramo na novi podmnožici parov. To pomeni, da lahko Dedupe izbere drugačne predikate in s tem zviša ali zniža največjo pričakovano oceno F.

5.3 Preprečevanje dupliciranja novih oseb

Ko so elementi v bazi deduplicirani, zbrani v gručah podobnih ljudi, model pa naučen in nastavitve predikatov znane, si lahko s temi podatki pomagamo tudi v primeru, ko hočemo preverjati nov element, ki bi ga radi shranili v bazo. Namen tega je zmanjšati število primerjav, ki bi jih bilo potrebno storiti, če si s temi podatki ne bi pomagali, ker bi vsako novo osebo, ki bi jo želeli shraniti, morali primerjati z vsako že obstoječo osebo in se nato vsakič odločati, ali so si podatki obeh oseb dovolj podobni, da ju označimo kot duplikat.

Z uporabo knjižnice Dedupe, s katero smo naučili model in s tem dobili podatke o nastavitvah in treniranju, za vsako novo entiteto preverimo, v katero gručo spada. Ko imamo ta podatek, lahko entiteto primerjamo samo z osebami, ki že pripadajo tej gruči. Prednost tega pristopa je v zmanjšanju števila oseb, ki jih primerjamo z novo entiteto. Ostane samo še odločitev, ali je oseba duplikat ali ta verjetnost ni dovolj velika.

Slabša stran tega je, da se lahko zgodi, da je bila za to osebo izbrana napačna gruča, v kateri ne bo izbrana kot duplikat nobene osebe. Če so podatki o tej osebi nepopolni ali napačni, je možnost, da ni padla v pravo gručo in jo tako vseeno shranimo v bazo kot unikaten objekt, čeprav obstaja verjetnost, da kot duplikat ustreza osebi v eni izmed drugih gruč.

5.4 Združevanje oseb

Ko smo za določeno množico oseb prepričani, da so duplicirane, se je potrebno odločiti, na kakšen način jih združiti. Odločil sem se, da za primarni objekt izberem osebo z največ podatki, ki jih nato le dopolnimo z manjkajočimi iz podatkov ostalih oseb v množici, če obstajajo. Ker imamo shranjene bloke oseb, lahko osebo, ki nastane po združenju podatkov, znova primerjamo glede na ključe blokov z osebami znotraj blokov, v katere se uvrsti. Ta oseba ima več podatkov, zato lahko z dopolnjenimi podatki odkrijemo nove duplikate, ki jih prej zaradi nezadostne verjetnosti nismo mogli ovrednotiti. Ostale objekte, ki niso bili izbrani za primarne in so le dopolnili primarne objekte s svojimi podatki zbrišemo iz podatkovne zbirke in jo s tem dedupliciramo. Proces primerjanja oseb z novimi podatki ponavljamo toliko časa, dokler se uvrščajo v vsaj en blok, kjer sta na konu vsaj dve osebi, saj jih le tako lahko primerjamo.

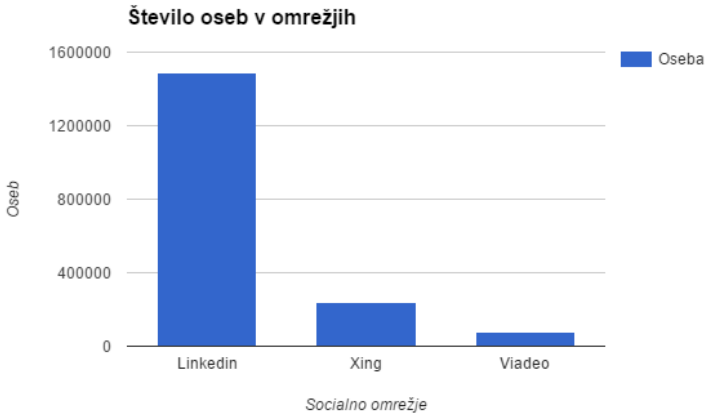
Poglavje 6

Analiza rezultatov

Pri analizi in oceni rezultatov sta me zanimala predvsem čas deduplikacije, torej časovna zahtevnost in pa pravilnost deduplikacije. Moja želja je bila 100% pravilnost zaznavanja resničnih duplikatov, torej, da se ne bi ponesreči združil kak par oseb, kjer v resnici ne gre za isto osebo.

6.1 Analiza oseb v podatkovni zbirki

Na sliki 6.1 vidimo, da v podatkovni zbirki prevladujejo osebe iz omrežja Linkedin(82, 48%), medtem, ko je oseb iz Xing-a 13, 06% in na Viadeo 4, 44%.



Slika 6.1: Število oseb v omrežjih

6.2 Časovna zahtevnost

Hitrost dedupliciranja je odvisna od števila oseb v bazi in izbire algoritma, ki ga izberemo za deduplikacijo. Ker se je knjižnica Dedupe izkazala za najhitrejšo in učinkovito zmanjšanje dupliciranih oseb, je hitrost odvisna od predikatov in posledično števila možnih blokov. Več blokov pomeni več različnih kombinacij, za katere se generirajo ključi bloka. Tako ima lahko ena oseba več ključev in s tem pade v več blokov, torej je potrebno več primerjav. Nato je odvisna od izbire dodatnega preverjanja generiranih gruč, ki ga izberemo, saj želimo imeti čimbolj pravilne rezultate in raje izpustimo kakšnega, kot pa združimo napačno ujemanje dveh oseb. Ker se algoritmi za dodatno preverjanje izvajajo na generiranih gručah, je čas deduplikacije odvisen tudi od njihovega števila.

6.3 Pravilnost deduplikacije

Pravilnost deduplikacije sem ocenjeval s številom pravilnih in nepravilnih ujemanj. Pravilna ujemanja sem določil z avtomatskim dodatnim preverjanjem, kjer sem uporabil opisana pravila, ki določajo stopnjo verjetnosti, da gre res za pravilno deduplikacijo. Nato sem ročno pregledal gruče oseb, ki sem jih ocenil, kot pravilno deduplicirane, saj le tako lahko zagotovim, da so algoritmi delovali pravilno. Ker je število ustvarjenih gruč dedupliciranih oseb, veliko, sem izmed njih izbral naključni vzorec in ocenil, če se podatki ujemajo.

6.3.1 Izvajanje deduplikacije

Na podlagi aktivnega učenja, se je izbrala kombinacija dveh predikatov:

- Prve 3 črke za atribut *izobrazba*
- Enako polno ime

Ustvarjenih je bilo 23 241 gruč prepoznanih duplikatov.

	Blokiranje	Gručanje	Skupaj
Čas(s)	54,0	40,0	94,0

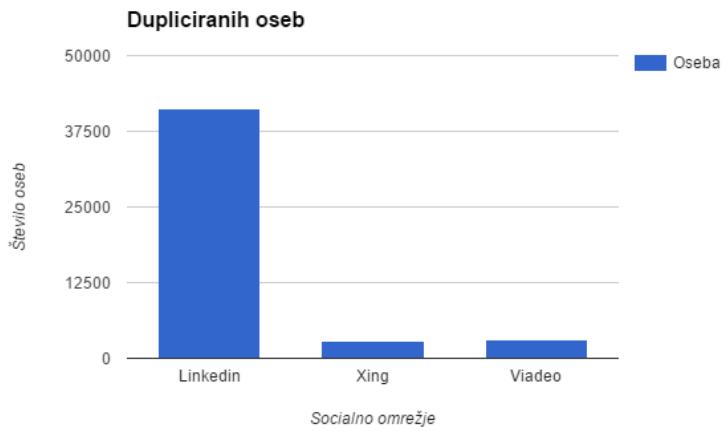
Tabela 6.1: Čas izvajanja deduplikacije

pravilnost	LinkedIn	Xing	Viadeo	Med omrežji
Zagotova ujemanja	15663 (79,05%)	238 (26,05%)	1134 (80,31%)	813 (69,13%)
Mogoča ujemanja	3306 (16,68%)	514 (56,23%)	188 (13,31%)	175 (14,88%)
Ni ujemanj	847 (4,27%)	162 (17,72%)	90 (6,38%)	188 (15,99%)
Gruč (23241)	19816 (85,26%)	914 (3,9%)	1412 (6%)	1176 (5%)

Tabela 6.2: Ujemanja

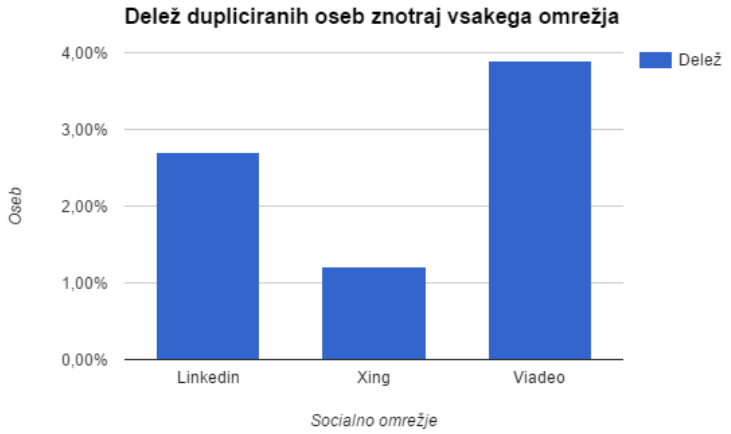
V primerjavi z ročnim generiranjem skupin podobnih Oseb se tudi zaradi paralelnega izvajanja proces deduplikacije izvede v 94-ih sekundah, kar ocenjujem za zelo hitro. Pri hitrosti pomagata tudi enostavna predikata.

Ker prevladujejo osebe pridobljene na Linkedin-u(87, 4%), je v njem našlo tudi največ duplikatov, kot je vidno na sliki 6.2. Xing(6, 0%) in Viadeo(6, 6%) predstavljata približno enak delež dupliciranih oseb.



Slika 6.2: Dupliciranih oseb

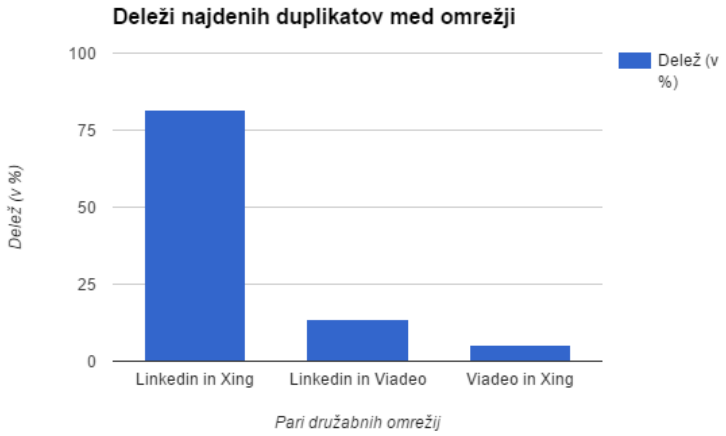
A, če deleže dupliciranih oseb primerjam glede na število oseb v vsakem omrežju (6.3), vidim, da je delež duplikatov največji na družabnem omrežju Viadeo.



Slika 6.3: Delež najdenih duplikatov znotraj vsakega omrežja

Dedupe je izmed 1 803 270 oseb prepoznal duplikate pri 47 303 osebah, kar predstavlja $2,62\%$ vseh oseb.

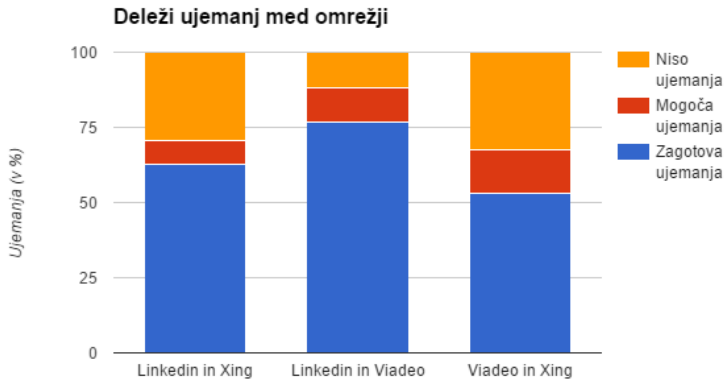
Analiza duplikatov med omrežji



Slika 6.4: Delež ujemanj med omrežji

Na sliki 6.4 se vidi, da je največ prepoznanih duplikatov bilo najdenih med omrežjema Linkedin in Xing, kar sem tudi pričakoval, glede na to, da sem o osebah iz teh dveh omrežij imel največ podatkov.

Kot lahko razberemo iz slike 6.5, sem z dodatnim preverjanjem na osnovi pravil, med vsemi zaznanimi duplikati med omrežji prepoznal večji delež resničnih duplikatov, v primerjavi z mogočimi ujemanji in tistimi pari, ki zagotovo niso duplikati.



Slika 6.5: Pravilnosti ujemanj med omrežji

Poglavje 7

Zaključek

Med izdelavo diplomske naloge sem razvil in uporabil algoritme, ki učinkovito deduplicirajo obstoječe duplikate v podatkovni bazi in preprečujejo shranjevanje novih. Spoznal sem prave izzive pri obdelavi javnih podatkov, ki jih uporabniki vnašajo v polja svojih profilov. Ker podatki niso točno specificirani, je njihova kvaliteta v povprečju zelo nizka za razliko od podatkov v moderiranih podatkovnih bazah in zato je dedupliciranje objektov še toliko težje. Zaradi veliko objektov v svoji podatkovni zbirki, sem ugotovil, da bi bilo primerjanje vseh oseb med sabo časovno neučinkovito in za zmanjšanje števila primerjav uporabil blokiranje. Ugotovil sem, da je težko določiti pravila, po katerih bi prepoznaval podobne osebe in jih tako spravil v iste bloke, zato sem s pomočjo aktivnega strojnega učenja ta pravila kombiniral in jih izbral več. Čeprav se je zaradi tega ustvarilo več skupin oz. blokov podobnih oseb, se je možnost pokritja več duplikatov povečala. Nato sem z uporabo knjižnice Dedupe za deduplikacijo na strukturiranih podatkih ustvaril gruče dupliciranih oseb na podlagi prej zgeneriranih blokov. Ko so bile gruče ustvarjene, sem za dodatno preverjanje uporabil svoje algoritme za primerjanje atributov oseb in profilnih fotografij in tako izboljšal rezultate knjižnice Dedupe. Tako sem v bazi združil le resnične duplikate in ignoriral tiste, za katere se je izkazalo, da v resnici niso bili duplikati. Uporaba različnih tehnik in algoritmov, predstavljenih v tem diplomskem delu, dedupliciranje in s tem

čiščenje strukturiranih zbirk znatno olajša in prihrani ročno delo z zelo zadovoljivimi končnimi rezultati. Vsekakor pa je na tem področju do trenutka, ko bo to delo v celoti lahko prevzel računalnik potrebno še veliko raziskav.

Slike

Tabele

Literatura

[1] I. Tetchner, Link Humans, LinkedIn Usage by Country: 2014 [Online]. Dosegljivo: http://linkhumans.com/blog/linkedin-usage-2014 [Dostopano 2. 2. 2016].

[2] Facts and figures [Online]. Dosegljivo: https://recruiting.xing.com/en/facts-and-figures [Dostopano 2. 2. 2016].

[3] H. Newcombe, J. Kennedy, S. Axford, A. James. "Automatic linkage of vital records." *Science*, 130, str. 954—959, 1959.

[4] I.P. Fellegi and A.B. Sunter, "A Theory for Record Linkage," *J. Am. Statistical Assoc.*, vol. 64, no. 328, str. 1183–1210, 1969.

[5] D.A. Cohn, L. Atlas, and R.E. Ladner,"Improving Generalization with Active Learning," *Machine Learning*, vol. 15, no. 2, str. 201–221, 1994.

[6] Y.R. Wang and S.E. Madnick, "The Inter-Database Instance Identification Problem in Integrating Autonomous Systems," *Proc. Fifth IEEE Int'l Conf. Data Eng. (ICDE '89)*, str. 46–55, 1989.

[7] E.-P. Lim, J. Srivastava, S. Prabhakar, and J. Richardson, "Entity Identification in Database Integration," *Proc. Ninth IEEE Int'l Conf. Data Eng. (ICDE '93)*, str. 294-301, 1993

[8] M.A. Hernández and S.J. Stolfo, "Real-World Data Is Dirty: Data Cleansing and the Merge/Purge Problem," *Data Mining and Knowledge Discovery*, vol. 2, no. 1, str. 9–37, 1998.

[9] T. Kuralt, "Avtonomen sistem za združevanje podatkovnih omrežij" *diplomsko delo*, 2009

[10] M. Bilgic, L. Licamele, L. Getoor, B. Shneiderman, "D-Dupe: An Interactive Tool for Entity Resolution in Social Networks", *Proceedings of IEEE Symposium on Visual Analytics Science and Technology*, 2006

[11] Christen, P. "A Comparison of Personal Name Matching: Techniques and Practical Issues." *Data Mining Workshops*, str. 290–294, 2006.

[12] Majumdar, Angshul, Rabab Kreidieh Ward. "Discriminative SIFT features for face recognition." *Electrical and Computer Engineering*, str. 27–30, 2009.

[13] Gregg, Forest and Derek Eder. Dedupe. [Online]. Dosegljivo: https://github.com/datamade/dedupe [Dostopano 26. 8. 2015].

[14] M. A. Hernandez and S. J. Stolfo. "The merge/purge problem for large databases." *Proc. of ACM SIGMOD*, str. 127-–138, 1995.

[15] "The Soundex Indexing System". National Archives and Records Administration. 30.5.2007 [Online] Dosegljivo: http://www.archives.gov/research/census/soundex.html [Dostopano 7. 2. 2016]

[16] Brandon Amos, Bartosz Ludwiczuk, Jan Harkes, Padmanabhan Pillai, Khalid Elgazzar, and Mahadev Satyanarayanan. "OpenFace: Face Recognition with Deep Neural Networks." [Online]. Dosegljivo: http://github.com/cmusatyalab/openface [Dostopano 11. 1. 2016].

[17] Florian Schroff, Dmitry Kalenichenko, James Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering.", *CoRR*, 2015