BACHELOR'S THESIS 2021

Utvärdering och implementering av förslagsalgoritmer

Simon Widdenhielm, Lukas Sundberg

ISSN 1651-2197
LU-CS/HBG-EX: 2021-05

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

Utvärdering och implementering av förslagsalgoritmer

LUNDS UNIVERSITET

Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för datavetenskap

Examensarbete:
Simon Widdenhielm
Lukas Sundberg

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2021

Sammanfattning

Detta examensarbete har genomförts i samverkan med Telavox AB med syfte att utvärdera och implementera en ny algoritm som tar fram kontaktförslag i Telavox telekommunikations-applikation.

I början av examensarbetet identifierades tre möjliga nya algoritmer. Som ett resultat av utvärderingen av förslagsalgoritmerna och diskussioner med Telavox föll valet på *CP-rank algorithm,* vilket är en algoritm som rangordnar en användares kontakter i ett kommunikationsmedium, som till exempel Facebook eller Twitter. Rangordningen baseras på data om en användares kommunikation med sina olika kontakter. Förutom att rangordna kontakterna väljer algoritmen även ut ett rekommenderat kommunikationsverktyg såsom telefonsamtal eller textmeddelande för var och en av dem.

CP-rank algorithm implementerades i Telavox system och utvärderades sedan genom att utföra intervjuer på anställda på Telavox där de fick beskriva hur de upplevde resultatet av algoritmen. Den generella åsikten bland de deltagande var att algoritmen gav bättre kontaktförslag än de som fanns implementerade i Telavox mobil- och webbapp. Examensarbetet skapade ett underlag för framtida utveckling av förslagsalgoritmen på Telavox som kan användas som grund i framtida projekt.

Nyckelord

Förslagsalgoritm, Markovkedjor, social-strength, intervjuer.

Abstract

This bachelor thesis was done in cooperation with Telavox AB and had the main purpose of evaluating different algorithms that rank contacts in a communication-application with the end goal of choosing the most suitable algorithm and implementing it into Telavox business-communication platform.

At the beginning of the thesis work three algorithms were chosen as potential candidates to be evaluated in regards to Telavox needs and the domain context of the Telavox communication platform. After the evaluation of the different suggestion algorithms and discussions with supervisors from Telavox it was decided to use the CP rank algorithm, which the algorithm ranks a users contacts based on a number of different parameters eg. the amount of interactions a user has with a specific contact. The algorithm also ranks the different communication tools. Combining these inputs the algorithm presents a contact suggestion and a preferred communication tool with that specific contact that is suggested. The communication tools that the algorithm refers to are phone calls or text messages, but the algorithm can be applied for other communication tools.

The CP rank algorithm was implemented in Telavox system. After implementation the algorithm was evaluated by conducting a set of user tests on employees of Telavox where the participants gave their opinions on the suggestions that the algorithm generated. The general opinion was that the algorithm generated a better list of contact suggestions than the current algorithm on the platforms. The thesis created a foundation for further development of the suggestion algorithm in Telavox application.

Keywords

Ranking Algorithm, Markovchains, social-strength, interview

Förord

Vi vill börja med att tacka Telavox som gav oss möjligheten att utföra detta examensarbete. Vi vill också tacka våra handledare från Telavox: Jonas Nolte och Anders Andersson för all hjälp och vägledningen under examensarbetet. Vi vill även tacka vår handledare Christian Nyberg och vår examinator Christin Lindholm på Lunds Tekniska Högskola.

Innehållsförteckning

1 Inledning

Detta examensarbete har utförts i samarbete med Telavox AB och har handlat om att utvärdera algoritmer som tar fram kontaktförslag och att välja ut en av dessa för implementation i Telavox system.

1.1 Bakgrund

Telavox är ett svenskt företag med huvudkontor i Malmö som erbjuder ett antal olika versioner av en applikation. Applikationen, som är Telavox huvudprodukt, samlar all kommunikation hos ett företag i en enda plattform. I applikation samlas ett företags chatt, e-post, telefoni, växeltjänst etc. och finns i versioner för Android, iOS samt som en webbtjänst. De typer av företag som Telavox har som kunder skiljer sig åt mycket. Det kan vara allt från företag med tusentals anställda till mindre företag med fåtal anställda. Eftersom Telavox kunder är så diversifierade så måste deras applikation vara anpassningsbar.

Examensarbetet inriktar sig på applikationens förslagslista. Det är en lista med ett antal kontakter som visas i applikationen som man då snabbt kan kontakta direkt. Som den fungerar just nu baseras vilka som visas i listan på vilka kontakter användaren ringt mest. Anställda på Telavox har uttryckt att algoritmen för kontaktförslag inte är tillräcklig och att det finns fler faktorer man kan använda sig av. En nackdel med att enbart använda samtalshistoriken som grund för förslagslistan är att de andra kommunikationsformerna inte beaktas. För en användare som sällan eller aldrig använder samtalsfunktionen, utan t.ex. främst använder sig av chattfunktionen, kommer förslagslistan inte att visa vilka vederbörande haft mest kontakt med. En annan nackdel med algoritmen är att inga andra faktorer förutom antalet kontakttillfällen beaktas. En faktor man skulle kunna använda sig av vid utformandet av en ny algoritm är vid vilken tid på dygnet som en användare brukar ha kontakt med annan. Att beakta applikationens samtliga kommunikationsformer i kombination med att använda sig av faktorer såsom vilken tid det är på dygnet kan utgöra grunden för en ny algoritm.

1.2 Syfte

Examensarbetet har syftet att först välja ut en algoritm som förbättrar förslagslistan för att sedan implementera och testa den och mäta resultatet. Den ska implementeras i alla applikationer som Telavox erbjuder. Förhoppningen är att en ny implementation av förslagslistan leder till en mer funktionell och användbar förslagslista vilket resulterar i att användarna av applikationen sparar tid i fall Telavox skulle välja att använda sig av den.

1.3 Målformulering

I detta examensarbete ska först en prototyp skapas som implementerar en ny algoritm för förslagslistan. Denna prototyp ska sedan utvärderas.

1.4 Problemformulering

I detta examensarbete kommer utgå från följande frågeställningar:

- Hur fungerar algoritmen som Telavox använder idag?
- Vilka algoritmer finns idag för att välja ut kontaktförslag i en kontaktlista?
- Vilka kriterier ska användas för att välja algoritmen?
- Hur ska förslagslistan i prototypen utvärderas i förhållande till kriterier som kommer att formuleras under examensarbetet?

1.5. Motivering

Anledningen till att det här examensarbetet valdes har sin bakgrund i ett antal förslag som Telavox kom med. Valet föll på favoritlistan eftersom den är något som man kan göra på väldigt många olika nivåer och att man direkt kan testa sina lösningar. Vid kontakt med Telavox uttryckte de också att förslagslistan idag är väldigt bristfällig och att de skulle ha

nytta av en ny algoritm. En annan faktor som bidrog till valet var ett direkt intresse för ämnet baserat på användning av appar som messenger där liknande funktionalitet finns.

Från Telavox perspektiv ligger nyttan med examensarbetet i att de får en bättre produkt. Ur ett ännu bredare perspektiv kan examensarbetet bidra till en förslagslista som spar tid åt dess användare.

1.6 Avgränsningar

Prototypen som skall skapas kommer avgränsas till Telavox webbapplikation.

Examensarbetet kommer använda Java.

Antalet algoritmer som kommer undersökas avgränsas till maximalt tio stycken.

2 Teknisk bakgrund

I detta kapitel beskrivs de system hos Telavox som är relaterade till examensarbetet samt tekniker, verktyg och teorier som har använts under examensarbetet. Kapitlet börjar med att beskriva de algoritmer som har utvärderats efter ett antal kriterier som bestämdes i samarbete med Telavox. Dessa finns i kapitel 3.2. Därefter beskrivs Telavoxs applikation Flow följt av en generell beskrivning av dataarkitekturen i Telavoxs system som examensarbetet berör. Slutligen beskrivs programverktyg och system som används under anpassningen och implementeringen av den utvalda algoritmen.

2.1. Förslagsalgoritmer

Tre algoritmer valdes ut för att utvärderdering efter de kriterier som står i kapitel 3.2. Hur de fungerar finns beskrivet i detta delkapitel.

2.1.1 PageRank

PageRank [1] är en algoritm som används av Google för att rangordna webbsidor vid användning av deras sökmotor. Varje webbsida tilldelas ett numeriskt värde kallat PageRank som beskriver hur pass viktig den är. Grundprincipen för hur algoritmen fungerar är att en webbsida får ett högre värde ju fler andra hemsidor som länkar till den. Länkarna till webbsidan viktas olika tungt beroende på PageRank-värdena hos webbsidorna som länkarna kommer från.

2.1.2 Friend Recommendations in Social Networks using Genetic Algorithms and Network Topology

Denna algoritm [8] är ger förslag på nya vänner i ett socialt nätverk. Först skapas en mängd vänförslag som sedan rangordnas. Det första steget i algoritmen är att filtrera bort vänförslag till kontakter som man inte har gemensamma vänner med. Sedan läggs också sådana användare som i detta sammanhanget anses som extroverta in i mängden, vilket de i artikeln väljer att definiera som användare med många vänner.

Sedan bestäms rangordningen av vänförslagen utifrån användning av en genetisk algoritm. I datavetenskap är en genetisk algoritm en typ av algoritm som inspirerats av teorin om naturligt urval, och de används ofta för olika typer av sökproblem och för att göra vissa typer av urval ur en mängd. I denna algoritm sparas ett vänförslag i form av en sträng som representerar en genuppsättning. Varje gen består av en viss typ av data som vägs in i beräkningen av rangordningen såsom gemensamma vänner, gemensamma intressen och ålder. En gen har ett numeriskt värde, 1 eller 0, för att representera om vänförslaget är lämpligt enligt ett viss kriterium. Rangordningen av vänförslagen baseras på antal matchningar av gener där olika gener väger olika tungt. Gemensamma vänner är den viktigaste genen.

2.1.3 CP-rank

Algoritmen som valdes ut fungerar så att den skapar en rangordning av en användares kontakter baserat på användningen av olika kommunikationsverktyg såsom textmeddelanden eller telefonsamtal. Den skapar också en rangordning av kommunikationsverktygen för var och en av kontakterna så att det kommunikationsverktyg som en användare är mest sannolik att använda sig av i kommunikation med en viss kontakt hamnar överst i rangordningen.

Det första steget i algoritmen är att räkna ut ett mått som kallas “social strength”. Det är ett värde som räknas ut för en användares samtliga kontakter, samt för de olika kommunikationsverktygen. Det beskriver hur pass socialt närstående en användare är med en kontakt, och hur pass benägen en användare är att kommunicera med en kontakt genom ett visst kommunikationsverktyg. För att räkna ut social strength för en kontakt beaktas all kommunikation som en användare haft med kontakten och antalet interaktioner som användaren haft med den via de olika kommunikationsverktygen räknas. Interaktionerna med en kontakt vägs olika tungt beroende på hur mycket en användare använder de olika kommunikationsverktygen. Om en användare kommunicerar mest via textmeddelanden kommer denna kommunikationsform att väga tyngre än de andra i beräkningen. Därefter görs social strength-beräkningen för de olika kommunikationsverktygen. Dessa värden sätts sedan in i en matris som beskriver övergången mellan de olika tillstånden, det vill säga kontakterna och de olika kommunikationsverktygen. Utifrån denna kan man beräkna rangordningen av kontakterna och kommunikationsverktygen.

Beräkningen av social strength för en kontakt går till på så sätt att en summaformel används där de olika kommunikationsverktygen gås igenom. Det som summeras är ett bråk där täljaren består av antalet interaktioner som användaren haft med kontakten för ett visst kommunikationsverktyg multiplicerat med ett värde som beskriver hur pass mycket

användaren använder detta kommunikationsverktyg. Nämnaren av bråket är en summering av antalet interaktioner som användaren haft med alla kontakter för kommunikationsverktyget. Beräkningen av social strength för ett visst kommunikationsverktyg sker på samma sätt, fast med skillnaden att nämnaren består av antalet interaktioner med en specifik kontakt. Värdet som beskriver hur mycket användaren använder ett visst kommunikationsverktyg räknas ut med en annan summaformel av ett bråk där det totala antalet interaktioner för ett kommunikationsverktyg delas med det totala antalet interaktioner för samtliga kommunikationsverktyg.

$$S_{i,a} = \sum_{j=1}^{q} \frac{f_{i,j,a} * thres_{i,j}}{\sum_{k=1}^{n} f_{i,j,k}}$$

Ekvation 1: *Formeln för social strength-beräkningen*

$S_{i,a}$ i *ekvation 1* är värdet på social strength för en kontakt *a* i ett kommunikationsmedium *i*. Under examensarbetet gjordes beräkningen enbart på ett kommunikationsmedium, Telavox applikation.

Den matematiska notationen för beräkningen av användningen av ett visst kommunikationsverktyg beskrivs i *ekvation 2*.

$$thres_{i,j} = \frac{\sum_{k=1}^{n} f_{i,j,k}}{\sum_{m=1}^{q} \sum_{k=1}^{n} f_{,i,m,k}}$$

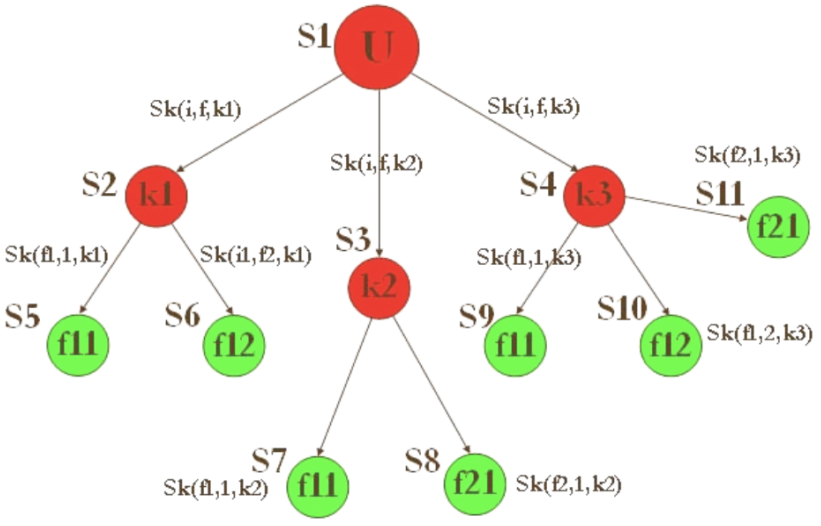
Ekvation 2*: Formeln för thres-värdet i social strength-beräkningen*

Rangordningen av kontakterna och kommunikationsverktygen bestäms genom användning av markovkedjor som beskriver övergångar mellan olika tillstånd i en graf. Övergångarna mellan noderna i grafen har en viss sannolikhet, och denna sannolikhet beror enbart på dess befintliga tillstånd. I algoritmen representerar de olika noderna en användares kontakter och de olika kommunikationsverktygen. Bågarnas värde ges av social strength-beräkningarna och övergångarna mellan dem representeras i form av en matris. Utifrån denna matris räknas en sannolikhetsvektor ut som ger sannolikheterna för att en användare kommer att kommunicera med en viss kontakt via ett specifikt kommunikationsverktyg. Dessa sannolikheter används sedan för att rangordna kontakterna och kommunikationsverktygen.

Matrisen och grafen för en användare med tre kontakter kan se ut enligt exempel i *figur 3*

:

$$P_{tr} = \begin{bmatrix} 0 & 0 & 0 & 0 & s_k(i1,f1,k1) & s_k(i1,f2,k1) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s_k(i1,f1,k2) & s_k(i2,f1,k2) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_k(i1,f1,k3) & s_k(i1,f2,k3) & s_k(i2,f1,k3) \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$



Figur 3: *Exempel på en övergångsmatris och motsvarande Markovkedja* [3]

Den översta noden representerar en användare, och bågarna ner till k1-k3 representerar sannolikheterna att användaren väljer att kommunicera med en av sina kontakter. Bågarna ner till nästa nivå representerar sannolikheten att kommunikationen med en kontakt sker via ett visst kommunikationsverktyg.

Sannolikhetsvektorn har som utgångspunkt värdet 1 för tillstånd S1 och 0 för alla andra eftersom markovkedjan alltid börjar i det tillståndet. Genom att multiplicera matrisen med sannolikhetsvektorn två gånger ges sannolikheterna för att kommunikationen med en kontakt sker via ett visst kommunikationsverktyg.

2.2 Telavox

Telavoxs är ett globalt företag i telekommunikationssektorn med avdelningar över hela Norden och Storbritannien. Den huvudsakliga produkten som företaget tillhandahåller sina kunder med är systemet Flow som har allt ett företag behöver för att kommunicera och kollaborerar med andra företag eller kunder. Flow består av tre delar: Flow Admin, smartphone-applikationerna Flow iOS och Flow Android.

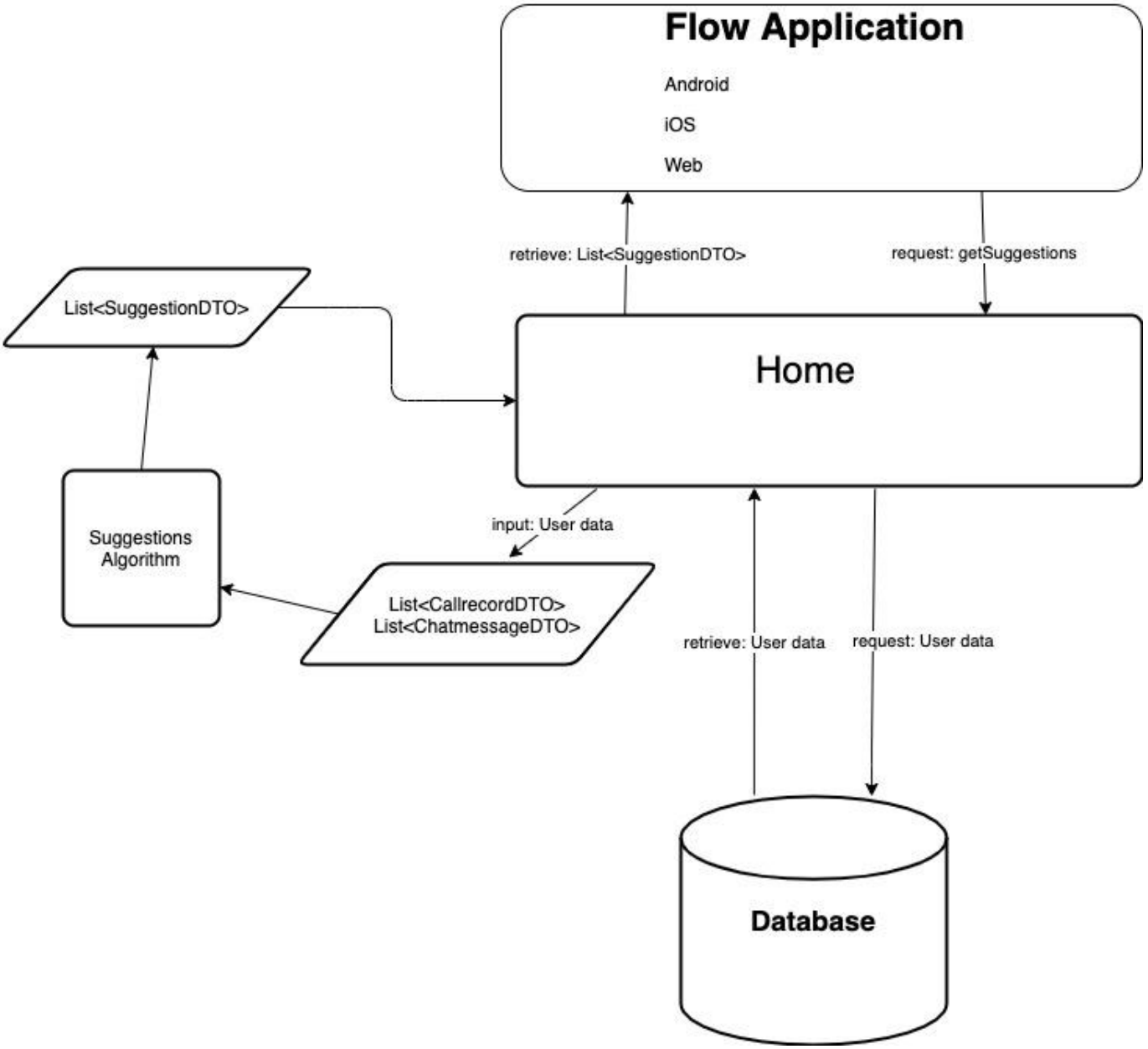
Flow Admin är ett administrationsverktyg som fungerar som en kontrollenhet där administratörer på ett företag kan skapa användare, begränsa eller öka behörigheter för användare inom företaget. Flow-applikationen som finns tillgängliga på både telefonen och webben riktat sig till anställda inom ett företag. Applikationen är en social plattform där användare kan chatta, ringa och ha videosamtal med varandra. Applikationen erbjuder också möjligheten att skapa gruppchattar och videokonferenser där företag har möten och diskussioner i olika kanaler.

2.3 Dataarkitektur hos Telavox system

Home är backenddelen av Telavox system som sköter anrop mot server, databaser och annan affärslogik. Home är uppdelat i tre lager: DAO, Services och Resources. DAO står för data access objects och ansvarar för kommunikation med databasen. Services ansvarar för all affärslogik det vill säga beräkningar och operationer på data. Resources ansvarar för kommunikationen mellan server och klienter. Mellan dessa lager skickas data som

DTO:er(Data transfer object) som en generell struktur som kan kommunicera med alla lager i home.

Flow-applikationen skickar regelbundet förfrågningar om uppdatering av förslagslistan till Home. Servicelagret i Home requestar efter DTO-objekt som innehåller information om en användares interaktioner, DTO:erna innehåller information om vem som användaren har en interaktion med och vid vilket datum som interaktionen har ägt rum . För beräkning av förslagslistan hämtas samtalshistoriken och chatthistoriken sedan en månad tillbaka. Efter att algoritmen har gjort operationer och beräkningar i servicelagret skickas det vidare en lista av SuggestionDTO:er till resourcelagret som sedan skickar vidare listan tillbaka applikationsklienten. Ett flödesdiagram som visuellt illustrerar hur förslagsalgoritmen fungerar i Telavox Flow Application beskrivs i *figur 4*.



Figur 4: *Flödesdiagram som beskriver hur förslagsalgoritmen fungerar i Telavox system.*

2.4 Postman

Postman [7] är ett utvecklingsverktyg som används för att skapa, dela, dokumentera och testa API-slutpunkter. Verktyget startades som ett sidoprojekt av Abhinav Asthana där han hade ambitionen att skapa ett verktyg som förenklar processen av att testa och utveckla API:er. Över 6 miljoner programmerare använder utvecklingsmiljön dagligen i arbetet när information behöver delas från ett gränssnitt till ett annat. Postman finns tillgängligt som en gratisversion som erbjuder begränsade tjänster, där en användare kan testa API slutpunkter men möjligheten att dela och kollaborera API:er begränsas. Postman erbjuder ett lätthanterligt användargränssnitt för att göra HTML-förfrågningar, utan besväret med att skriva en massa kod för att kunna testa ett API: s funktionalitet.
Programverktyget användes under utvecklingsfasen för att skapa och testa API-endpoints som utformningen av förslagsalgoritmen var beroende av.

2.5 Sequel Pro

Sequel Pro [6] är en databashanterare som licensieras av MIT university. Programmeringsverktyget är gratis med öppen källkod som gör det möjligt för alla i hela världen att delta i vidareutvecklingen av programmet och förbättra källkoden. Databashanteraren är mac-baserad och används för att hantera och organisera data på ett effektivt sätt. Sequel Pro kan kopplas upp mot lokala och fjärradatabaser och använder sig av programspråket MYSQL för all syntax som är kopplad till databashantering.

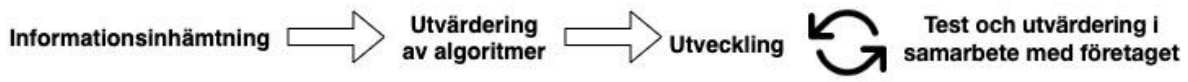
Programverktyget användes under utvecklingsfasen för att ställa förfrågningar till Telavoxs databaser för att hämta ut användardata såsom samtalshistorik och chatthistorik.

3 Metod

Examensarbetet har till stora delar utförts hemifrån på distans då den rådande coronasituationen som gäller under utförande av examensarbetet har begränsat möjligheten till fysiska möten. Telavox har erbjudit portabla datorer vilket har underlättat distansarbetet. Telavoxs har tilldelat examensarbetarna handledare som har varit med under hela processen av examensarbetet. Avstämningsmöten har hållits varje vecka där planering, utveckling och problem har diskuterats med handledarna.

Telavox utvecklingsgrupper arbetar efter agila processmodeller som uppmuntrar till en kreativ miljö där idéer snabbt kan växa och realiseras. En agil projektmodell som använts i tidigare projekt var Scrum [9]. Det som kunde passa oss i den arbetsmodellen var att dela upp projektet i två veckors sprintar med tydliga delmål samt att ha en utvärdering i slutet av varje sprint för att kunna lägga upp planeringen för nästa sprint. Arbetet skedde nästan alltid gemensamt antingen i ett fysiskt möte eller via videokonferens. På grund av detta valdes medvetet övriga delar av Scrum bort eftersom arbetet alltid skedde parvis och därmed kändes det onödigt med dagliga standup-möten för att uppdatera varandra.

Utveckling av algoritmen delades upp i fyra faser: Informationsinhämtning, utvärdering av algoritmer, utveckling eller implementering, test och utvärdering i samarbete med företaget se figur 5. De två första faserna utfördes sekventiellt för att skapa ett fast underlag inför de två nästkommande faser som utfördes iterativt för att uppnå en optimal och reviderad implementation av förslagsalgoritmen.



Figur 5: *Flödesdiagram som beskriver de faser som examensarbetet var uppdelat i.*

3.1 Informationsinhämtning

För att välja ut en lämplig algoritm skedde informationsinhämtning genom litteraturstudier. Ett antal vetenskapliga artiklar som beskrev olika rangordningsmetoder lästes i syfte att se om de skulle kunna användas för att förbättra Telavox förslagslista. En av algoritmerna som undersöktes var PageRank som används av Google för att rangordna sökresultat i deras sökmotor. En annan algoritm som undersöktes används för att ge vänförslag i ett socialt nätverk. Dessa algoritmer visade sig dock inte vara lämpliga att använda eftersom de inte direkt går att applicera på problemet som skulle lösas. Algoritmen som till slut valdes ut beskrivs i två vetenskapliga artiklar [2,3] och denna algoritm var den enda som fanns för att skapa en rangordning av kontakter i ett socialt nätverk. Eftersom algoritmen ger en direkt lösning på problemet som skulle lösas föll valet på den.

3.2 Utvärdering av algoritmer

För att kunna utvärdera algoritmerna behövde ett antal kriterier bestämmas. Detta gjordes i samspel med handledarna genom att föra diskussioner med dem i syfte att komma fram till kriterier som både täcker in Telavox önskemål på algoritmen och som var rimliga att utgå ifrån i ett examensarbete. Genom denna process bestämdes följande kriterier:

- Algoritmen ska vara direkt applicerbar på problem som ska lösas
- Algoritmen ska kunna implementeras utan att utgöra en onödig belastning på Telavox servar
- Algoritmen ska vara enkel att förstå
- Algoritmen ska enbart beakta interaktioner med en användares egna kontakter

När algoritmerna utvärderades med användning av kriterierna valdes CP-rank ut eftersom den uppfyllde samtliga av dem. Det var den enda av de tre algoritmerna som var utarbetad med syftet att rangordna en användares kontakter i en kommunikationsplattform. Huruvida de två andra algoritmerna skulle gå att applicera på problemet som skulle lösas krävde mer resonemang eftersom de är skapade för andra syften. I följande stycken beskrivs tankar om dessa två algoritmer sett utifrån kriterierna som ställts upp, och en anledning ges till varför de inte valdes ut.

PageRank sågs som en de algoritmerna som skulle kunna förbättra förslagslistan eftersom det är en rangordningalgoritm som använder sig av viktning av ett visst kriterium. Tanken var att algoritmen skulle kunna modifieras genom att rangordna en användares kontakter istället för webbsidor och att det som i algoritmen representeras av länkar mellan webbsidorna skulle kunna ersättas med antalet interaktioner som en användare haft med sina kontakter. Eftersom algoritmen viktar länkarna genom att tilldela dem ett större värde baserat på en webbsidas PageRank-värde visade sig algoritmen vara olämplig att använda. I den modifierade lösningen hade detta inneburit att en kontakt tilldelats ett PageRank-värde som även är baserat på antalet interaktioner med andra kontakter. Enligt krav från Telavox skulle enbart en användares interaktioner med sina egna kontakter beaktas.

Den genetiska algoritmen sågs också som ett potentiellt val av algoritm att använda, men den visade sig också vara olämplig. Mängden som algoritmen baseras på väljs ut baserat på gemensamma vänner och inte vilka kontakter en användare har. En möjlig modifiering av denna mängduppbyggnad hade varit att istället lägga in en användares samtliga kontakter i mängden. Eftersom algoritmen bygger på jämförelser av genuppsättningar som beskriver typiska saker som är viktiga i ett socialt nätverk såsom gemensamma intressen och närliggande ålder var detta något som inte sågs som en passande lösning på grund av att Telavox applikation huvudsakligen är ett enkelt kommunikationsverktyg och inte ett socialt nätverk på samma sätt som Facebook och liknande applikationer där användare delar med sig av sina intressen och annan dylik information.

3.3 Utveckling av prototyp

Utvecklingsfasen påbörjades med en inlärningsprocess där fokus låg på att sätta sig in i Telavox system- och kodstruktur. Handledarna presenterade ett grundskelett av klasser som projektet kunde utgå ifrån, med klasser i varje lager av back-end arkitekturen Home. Första delmålet var att hämta alla samtal för en användare. Detta delmål uppfattades som en rimlig start då CP-Rank bygger på att den informationen kan hämtas samt att uppgiften gav en bra överblick av klassuppbyggnaden i Telavox kodbas. Nästa steg var att på motsvarande vis hämta alla chattmeddelanden. Steget därefter bestod i att implementera CP-Rank i applikationens servicelager där alla operationer och beräkningar på data sker. Samtalen och chattmeddelandena lagrades i två olika hashmap där nyckeln i de båda fallen består av ett heltal som representerar ett id för kontakt. Värdet i vardera mapparna representerar antalet interaktioner en användare haft via antingen textmeddelanden eller telefonsamtal.

CP-Rank kan delas upp i tre steg. Först beräknas social strength för en användares alla kontakter samt de olika kommunikationsverktygen kopplade till var och en av kontakterna.

Nämnaren i social strength formeln beräknas genom att loopa igenom samtliga interaktioner som en användare haft och summera antalet interaktioner för varje kommunikationsverktyg.

Thres-värdet i formeln i *ekvation 2* representerar hur stor andel av kommunikationen som en användare har haft via ett visst kommunikationsverktyg i detta fallet samtal och chatt. Detta beräknades genom att dela antalet interaktioner en användare haft via ett visst kommunikationsverktyg delat med det totala antalet interaktioner som en användare har haft. Täljaren är en enkel multiplikation där ena faktorn är thres-värdet och den andra är antalet interaktioner med en viss kontakt med ett visst kommunikationsverktyg. Bråkräkningen loopades igenom två gånger för att få med både samtalen och textmeddelande.

Nästa steg av algoritmen var att sätta värden på elementen i en matris med syfte att kunna ge förslag på ett kommunikationsverktyg för varje kontakt. Kolumnen i matrisen är de olika tillstånden i grafen och raderna representerar sannolikheten för övergången till ett visst tillstånd. Antalet förslag som en användare presenteras med i applikationen är begränsat till fem stycken. Därför begränsades antalet kontakter i matrisuträkningen till de fem kontakter som hade högst social strength-värde. Matrisen fylldes i med de båda social strength-värdena enligt grafen i *figur 3*.

Grafen representerar en enkelriktad markovkedja. För att räkna ut sannolikhetsvektorn multiplicerades matrisen med en vektor som innehöll sannolikheterna för att grafen befinner sig i ett visst tillstånd. Eftersom markovkedjan alltid utgår från tillstånd S1 (se *figur 3*) sattes sannolikheten för tillstånd S1 i början till 1. Därmed sattes dess värde i vektorn till 1 och resterande till 0. För att få sannolikheterna för tillstånden två steg ner i grafen utfördes multiplikationen två gånger. Detta gav sannolikheterna för att en användare ska kommunicera med de fem olika kontakterna via de två olika kommunikationsverktygen. I och med detta har algoritmen tagit fram förslagen på kontakter och kommunikationsverktyg.

3.4 Test och utvärdering

För att testa och utvärdera implementationen av algoritmen CP-Rank genomfördes ett antal intervjuer med anställda på Telavox efter att förslagsalgortimen hade körts med deras användardata och tagit fram kontaktförslag och förslag på kontaktmetod. Testerna utfördes på två avdelningar på Telavox: Utvecklingsavdelningen och säljavdelningen.

Frågorna skickades ut till de deltagande innan intervjun genomfördes. Följande frågor ställdes:

1: Tycker du att de fem kontaktförslagen som väljs ut av algoritmen känns rimliga med avseende på hur du brukar kommunicera med dina kontakter? Om inte, varför?

2: Tycker du att förslagen på kontaktmetoder för var och en av kontakterna speglar hur du brukar interagera med denna person? Om inte, varför?

3: Är det nödvändigt att ge förslag på kontaktmetoder, eller hade det räckt att enbart rangordna kontakterna?

4: Hur hade algoritmen kunnat förbättras?

5: Tycker du att algoritmen borde användas i Telavox applikation istället för den nuvarande?

Hur intervjuerna gick till beskriv i detalj i kapitel 4.4

3.5 Källkritik

Under examensarbetet har all informationsinhämtning skett via internet. En del av webbsidorna som refereras till är manualer eller instruktioner för programverktyg som kommer direkt från tillverkaren och kan därmed anses vara tillförlitliga. Andra källor i examensarbetet är forskningspublikationer som kommer från erkända universitet.

PageRank [1]

Referensen är hämtad från en publikation från Stanfords universitet. Stanford ett av de mest prestigefyllda universiteten inom datavetenskap. Författarna av denna publikation är grundarna till de globala konglomeratet Alphabet som äger både Google och Youtube.

Friend Recommendations in Social Networks using Genetic Algorithms and Network Topology [8]

Forskningspublikationen är publicerad av ett Amerikansk universitet och författarna är alla välrenommerade forskare inom området.

Social Prioritization Algorithm [2][3]

Förslagsalgoritmen som valdes utgår från två artiklar där den första artikeln [2] innehåller en referens till den andra artikeln [3]. Den första artikeln hittades via sökmotorn LubSearch och är författad av forskare vid Luleå universitet. Den andra artikeln är också en forskningspublikation från Luleå universitet. Författarna av dessa artiklar har många publikationer som berör rankningen av kontakter inom sociala nätverk.

Postman [5]

Telavox utvecklare använder i skrivande stund Postman för att testa API-anrop. Till följd av detta blev Postman ett verktyg som användes under examensarbetet. Postmans hemsida innehåller en manual som användes under examensarbetet. Hemsidan är skapad av tillverkarna av Postman och således är referensen tillförlitlig.

Sequel Pro [4]

Likt Postman så använder utvecklarna på Telavox Sequel Pro. Under examensarbetet användes Sequels Pros egna hemsida som en manual vid testning av API:er. Hemsidan är skapad av kreatörerna av verktyget och därmed kan referensen anses vara pålitlig.

Telavox [10]

Bakgrundsbeskrivningen av Telavox baseras på information från deras egna officiella hemsida således är källan en tillförlitlig referens.

Scrum [7]

Referensen till Scrum är inhämtad från kurslitteratur i kursen Projekt Årskurs 3. Kursen är en del av programmet högskoleingenjör inom datateknik på Lunds Universitet vilket är en av de mest prestigefyllda universiteten i Sverige.

4 Analys

Detta kapitel handlar om val, problem och problemlösningar under examensarbetet gång. Examensarbetet har framförallt ställts inför ett avgörande val - vilken förslagsalgoritm som skall implementeras. Detta val har varit det huvudsakliga valet som examensarbetet till stora delar utgår från.

Diskussioner om målet med förslagsalgoritmen har skett under avstämningsmöten med handledare från Telavox. I synnerhet har förslagsalgoritmens tidskomplexitet diskuterats och olika val som påverkar algoritmens potentiella belastning på Telavox system under körtiden. Ett val relaterat till tidskomplexiteten var hur långt tillbaka i tiden skall användarhisitoriken som hämtas till förslagsalgoritmen utgå ifrån. Ett annat val var hur ofta bör förslagsalgoritmen köras under produktion för att minimera belastningen av Telavox system.

4.1 Informationsinhämtning

Informationsinhämtningen skedde genom att söka efter och läsa vetenskapliga artiklar på internet. De hemsidor som användes för att hitta sådana artiklar var Google och LUBSearch där en mängd olika söktermer testades. Dessa söktermer bestod av olika kombinationer av “contact ranking”, “contact suggestions”, “contact prioritization”, “social media”, “ranking”, och “algorithm”.

Att informationsinhämtningen skedde på internet motiverades av att ämnet som skulle undersökas är så pass nytt att det var svårt att på något annat sätt hämta in information. Den typen av rankingalgoritmer som är intressanta i examensarbetet används mest på sociala medier och de har inte funnits så länge så det kändes rimligt all att relevant forskning på ämnet borde gå att hitta på internet.

Algoritmen som slutligen valdes ut presenterades för handledarna på Telavox. Artikeln som innehöll algoritmen skickades till dem i pdf-format så att de skulle kunna bedöma den. De var inte insatta i all matematik som algoritmen använder sig av, men de kunde ändå förstå hur den fungerar i stora drag och ansåg den vara ett lämpligt val.

Det visade sig vara svårt att hitta algoritmer för det problem som skulle lösas och den enda som gick att applicera direkt fanns i de två vetenskapliga artiklarna som valdes ut. Bristen på allmänt tillgänglig forskning inom ämnet beror antagligen på att utvecklingen och användningen av denna typ av algoritmer främst sker hos privata företag såsom Facebook. Om ett företag har investerat mycket pengar i att utveckla en algoritm har det inte någon anledning att publicera den eftersom konkurrenter skulle kunna dra nytta av den.

4.2 Utvärdering av algoritmer

Informationsinhämtningen resulterade i att tre algoritmer valdes ut för utvärdering: PageRank, en genetisk algoritm och CP-rank-algoritmen. Under utvärderingen fördes en diskussion med två av handledarna på Telavox, Jonas Nolte och Daniel Hellström, om vilka krav en algoritm behöver uppfylla för att väljas ut för implementation. Dessa diskussioner ledde till att ett krav formulerades om att algoritmen ska vara direkt applicerbar på problemet som skulle lösas och inte behöva genomgå några större modifieringar. Att välja en algoritm som inte gick att applicera direkt hade inneburit ett stort arbete för att omarbeta den, vilket både författarna och handledarna på Telavox ansåg vara en alltför stor uppgift för ett examensarbete. Ett annat krav som formulerades var att algoritmen inte ska utgöra en alltför stor belastning på Telavox servrar. Detta krav kom från handledarna på Telavox som motiverade det med att korta exekveringstider är en viktig del i att säkerställa en bra användarupplevelse för användare av Telavox applikation. En algoritm hade inte kunnat användas om den lett till en märkbar fördröjning i applikationen. Ytterligare ett krav som formulerades var att algoritmen ska vara enkel att förstå. En komplicerad och svårbegriplig algoritm hade kunnat ta så lång tid att anpassa och implementera att det inte hade hunnits med under ett examensarbete. Dessutom var en övergripande förståelse av en algoritm nödvändig för att över huvud taget kunna utvärdera den. Det sista kravet som formulerades var att algoritmen enbart ska beakta interaktioner med en användares egna kontakter. Detta krav uppkom i samband med utvärderingen av PageRank-algoritmen som vid användning hade inneburit att samtliga användare av Telavox applikation hade kunnat ges som förslag för en användare. När detta framfördes till handledarna på Telavox sade de att detta inte var önskvärt från deras sida.

PageRank sågs som en av de algoritmerna som skulle kunna förbättra förslagslistan eftersom det är en rangordningalgoritm som använder sig av viktning av. Tanken var att algoritmen skulle kunna modifieras genom att rangordna en användares kontakter istället för webbsidor

och att det som i algoritmen representeras av länkar mellan webbsidorna skulle kunna ersättas med antalet interaktioner som en användare haft med sina kontakter. Eftersom algoritmen viktar länkarna genom att tilldela dem ett större värde baserat på en webbsidas PageRank-värde visade sig algoritmen vara olämplig att använda. I den modifierade lösningen hade detta inneburit att en kontakt tilldelats ett PageRank-värde som även är baserat på antalet interaktioner med andra kontakter. Enligt krav från Telavox skulle enbart en användares interaktioner med sina egna kontakter beaktas.

Den genetiska algoritmen sågs också som en möjlighet, men den visade sig också vara olämplig. Algoritmen använder sig av en mängd potentiella förslag som består av de användare som en viss användare har gemensamma vänner med. En möjlig modifiering av denna mängd hade varit att istället lägga in en användares samtliga kontakter i mängden. Eftersom algoritmen bygger på jämförelser av genuppsättningar som beskriver vad som är viktigt i ett socialt nätverk såsom gemensamma intressen och närliggande ålder var denna algoritm inte någon lämplig lösning på grund av att Telavox applikation huvudsakligen är ett enkelt kommunikationsverktyg och inte ett socialt nätverk på samma sätt som Facebook och liknande applikationer där användare delar med sig av sina intressen och annan liknande information.

Att valet av algoritm till sist föll på CP-rank-algoritmen var helt enkelt för att det var den enda som hittades som gick att applicera på problemet utan modifiering. Den skapar en rangordning av kontakterna, vilket efterfrågades av Telavox, och ger dessutom ett förslag på kontaktmetod för var och en av dem, vilket gav examensarbetet ytterligare djup.

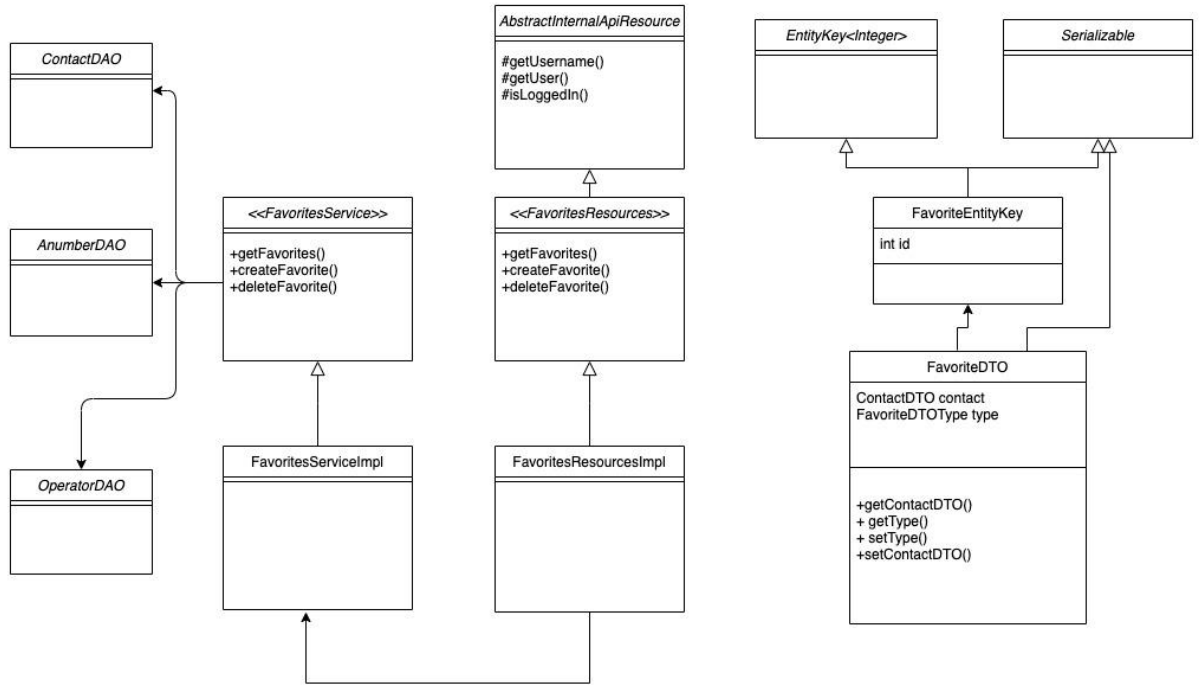
4.3 Utveckling

De tekniska verktyg som användes under examensarbetet såsom Java, PostMan och Sequel Pro är samma verktyg som Telavox använder sig av internt för mjukvaruutveckling. När utvecklingsmiljön sattes upp var det dessa verktyg som installerades enligt instruktioner från Telavox, och de användes sedan under hela examensarbetet.

I samband med att utvecklingsmiljön sattes upp fick författarna också tillgång till hela Telavox kodbas. Detta var nödvändigt för att utveckla och testa den kod som skrevs under examensarbetet, och det möjliggjorde också för författarna att förstå hur Telavox system fungerar.

Anledningen till att utvecklingen delades upp i flera olika faser var att det kändes rimligt att lägga fokus på en viss del av utvecklingen vid olika tidpunkter. När algoritmen valts ut studerades uppbyggnaden hos Telavox system. Anledningen till detta var att möjligheten att kunna implementera algoritmen byggde på en grundläggande förståelse av systemet och kunskaper om att kunna hämta ut data rörande telefonsamtal och textmeddelanden.

För att förstå hur klasserna skulle interagera med varandra och hur de var tänkta att fungera internt studerades klasserna som hanterar applikationens favoritlista (se klassdiagrammet i *figur 10*). Favoritlistan är en lista med kontakter som en användare själv väljer ut för att snabbare kunna komma i kontakt med dessa. De klasser som hanterar favoritlistan är strukturerade på samma sätt som de som hanterar all annan funktionalitet i Home-delen. Klasserna som hanterar favoritlistan är uppdelad i två lager: resources och services. Resources-klasserna är API-endpoints som anropas från användargränssnittet när information efterfrågas eller manipuleras. Resources är huvudsakligen ett lager som sköter kommunikationen mellan klienten och servern. När en klient antingen manipulerar data eller hämtar data, anropas en utav metoderna i FavoriteResources. Anropet kan antingen hämta en lista av användarens favoritkontakter eller förändra innehållet av favoritlistan, t.ex ta bort eller lägga till kontakter i favoritlistan. FavoriteResources anropar i sin tur på metoderna i FavoriteServices som hämtar data via DAO klasserna. Sedan sköter FavoriteServices all affärslogik innan datan skickas tillbaka som FavoriteDTO:er till API-slutpunkten FavoriteResources.



Figur 6: *Klassdiagram för resources & services för kontaktfavoriter-klasser*

Tanken bakom att ta utgå från dessa klasser var att de, i likhet med vad som skulle implementeras, också hanterar en lista av kontakter, vilket kan betraktas som en rimlig utgångspunkt. Det visade sig dock inte vara så mycket nyttigt att hämta från dessa klasser och en hel del tid gick åt att försöka efterlikna deras interna struktur när det egentligen var något helt annat som behövde göras. Den viktiga skillnaden var att algoritmen som skulle implementeras byggde på att hämta in stora mängder data och att utföra beräkningar på den datan, medan klasserna som hanterar favoritlistan bara hämtar in en liten mängd data. Klasserna som hanterar favoritlistan sparar data i form av DTO:er, vilket är rimligt för små mängder data, men inte lämpligt när större mängder ska hanteras. Ett beslut togs därmed av examensarbetarna om att spara den data som behövdes internt i klassen istället för skapa en stor mängd DTO-objekt.

Att hämta alla chattmeddelanden som behövdes var inte så enkelt som det såg ut vid en första anblick, utan en del problem stöttes på när det skulle göras. Alla chattmeddelanden sparas i så kallade chat sessions som representerar ett chattrum mellan två eller flera personer. För algoritmen var det bara intressant att hämta meddelanden som en användare skickat till eller tagit emot från enskilda kontakter, men det visade sig inte finnas någon färdig metod för att enbart hämta dessa typer av meddelanden. Istället fick samtliga meddelanden hämtas, det vill

säga inklusive de som förekom i gruppchatter. Meddelandena filtrerades sedan på så sätt att de meddelanden som förekom i gruppchatter togs bort. I koden gjordes detta genom kontroll av attributet chatSessionMemberSet i varje chat session som hämtats. ChatSessionMemberSet innehåller en lista på alla användare i en chat session och om storleken på denna lista inte var lika med två behölls inte dess chat session. När filtreringen hade utförts uppkom ett problem i form av att vissa chat sessions kunde vara helt tomma. Detta var inte något som förväntades, utan det antogs att alla skulle innehålla minst ett meddelande. Anledningen till att de kunde vara tomma var att en ny chat session skapades så fort ett nytt meddelandefönster öppnats även om inget meddelande skickats än. Detta resulterade i null pointer exceptions när meddelanden skulle hämtas från en tom lista. Mycket felsökning krävdes för att hitta orsaken till detta problem, men det löstes genom att kontrollera storleken på listan som hade hämtats. Att sedan hämta ut den information som behövdes från ett meddelande var ytterligare ett problem. När samtalen hämtades representerades kontakten som en användare haft ett samtal med i form av en contactDTO vilket är ett DTO-objekt som innehåller information om en viss kontakt, t.ex namn,telefonnummer,email. Denna representation fanns ej att tillgå vid hämtning av meddelanden, utan där representerades kontakten istället med ett konto-id vilket är en unik sifferkombination som är unik för varje användare. För att kunna koppla ihop både samtal och meddelanden med samma typ av objekt behövdes därför detta konto-id göras om till en contactDTO. Detta gjordes genom klassen contactDTOFactory vilket är en klass som bygger på design mönstret factory pattern. Klassen tillhandahåller funktioner för att skapa ContactDTO:er utifrån parametrar såsom telefonnummer, konto-id. Även om det löste problemet var det inte särskilt effektivt, utan det optimala hade varit om både samtal och meddelanden haft samma sätt att representera en kontakt.

Ett utav de största tekniska problemen som stöttes på under utvecklingen hade att göra med möjligheten att kunna testa den kod som skrivits. Under en stor del av utvecklingen testades algoritmen enbart på examensarbetarnas egna konton som tilldelats av Telavox. Problemet med detta var att dessa konton inte använts tillräckligt mycket för att kunna göra en rimlig bedömning av algoritmen. För att lösa detta problem kontaktades handledarna som gav tillgång till en testanvändare med ett större antal interaktioner. Senare gav de också möjligheten att kunna logga in på olika användare i systemet för att kunna testa algoritmen på dem. Ett annat problem med att testa koden var att den databas som användes för tester enbart hade data för en vecka tillbaka i tiden. Implementationen av algoritmen var utformad efter att hantera data för en månad tillbaka vilket gjorde att testkörningarna gav oväntade resultat. Att testdatabasen inte innehöll all data som behövdes upptäcktes inte förrän intervjuer med användarna satte igång då flera som deltog i dem rapporterade att vissa kontaktförslag

upplevdes om märkliga på så sätt att dessa kontaktförslagen inte var personer som de brukar ha mycket kontakt med. Då löstes problemet genom att köra algoritmen mot en databas som innehöll data om kontakter som var minst en månad gammal.

Att sätta en begränsning på hur lång tid tillbaka data skulle hämtas ifrån, det vill säga en månad tillbaka, gjordes dels för att koden som skrivits inte skulle ta för lång tid att exekvera och dels för att kontaktförslagen inte skulle baseras på väldigt gammal information. Att hämta all kommunikation som en användare haft över flera år hade tagit alltför lång tid och det hade heller inte gett någon bra bild av en användares nuvarande kommunikationsvanor. Å andra sidan ansågs det av författarna inte heller rimligt att basera kontaktförslagen på data som inte sträckte sig tillräckligt långt tillbaka i tiden. Tanken var att fånga en användares vanor och om det bara fanns data för en vecka tillbaka hade det lett till att ett kontaktförslag hade kunnat ges för en person som en användare bara tillfälligt haft mycket kontakt med under den veckan. Att just en månad valdes som tidsbegränsning var i grunden ett godtyckligt val, men det ansågs ändå som rimligt utifrån målet att fånga en användares kommunikationsvanor, vilket handledarna på Telavox instämde i.

Algoritmen testades under utvecklingsfasen genom att göra anrop till den API-endpoint som skapats för det syftet. Detta gjordes med verktyget Postman och resultatet presenterades där i JSON-format. Algoritmens implementerades aldrig på så sätt att resultatet kunde ses i Telavox applikations användargränssnitt, utan det var även Postman som senare användes i testerna.

Under ett tidigt skede av utvecklingen framförde handledarna på Telavox att det hade varit bra att tänka på hur ofta förslagslistan ska uppdateras genom att hämta ny data och att implementera en lösning för detta. Motiveringen bakom detta var att det hade utgjort en onödig belastning på Telavox servrar om algoritmen hade körts varje gång en användare haft en ny interaktion. På grund av tidsbrist var detta dock inte något som hanns med under examensarbetet.

Ett annat problem som uppkom under utvecklingen var att algoritmen var svår att förstå eftersom det krävdes förkunskaper i delar av matematiken inte har ingått i examensarbetarnas kurser.

Delar av algoritmen använde sig av markovkedjor vilket författarna inte har studerat tidigare. Problemet löstes genom att ta hjälp av två anställda på Telavox som stött på denna typ av matematik under sina studieår. De förklarade de sista stegen i algoritmen på ett enklare sätt än vad den vetenskapliga artikeln gjorde.

Ytterligare ett problem som stöttes på var att den data som behövdes för algoritmen, antalet telefonsamtal och chattmeddelanden mellan olika personer, inte gick att hämta på ett effektivt sätt. I Telavox kodbas fanns inga färdiga metoder för detta, utan den datan fick hämtas genom att gå igenom alla samtal och chattmeddelanden som en användare haft med alla sina kontakter. För att göra implementationen av algoritmen mer effektiv hade antalet interaktioner mellan en användare och olika behövt sparas i en databastabell. Detta var inget som löstes under examensarbetet, men problemet framfördes till de anställda på Telavox ifall de skulle vilja använda sig av koden i sin produkt.

4.4 Test och utvärdering

Algoritmen utvärderades genom att intervjua anställda på två avdelningar på Telavox som använder applikationen i den dagliga kommunikationen på företaget. Totalt deltog 16 personer varav 8 var anställda på utvecklingsavdelningen och 8 var anställda på säljavdelningen.

Anledningen till att intervjuer med användare valdes som utvärderingsmetod var att subjektiva upplevelser ansågs som det enda rimliga sättet att bedöma denna algoritm och jämföra med den algoritm som i skrivande stund används. Efter intervjuerna var genomförda insågs det dock att utvärderingen hade kunnat ske på ett annorlunda sätt. Ett exempel på en sådan utvärderingsmetod hade kunnat gå till på så sätt att de deltagande i intervjuerna hade kunnat skriva en lista på vilka kontaktförslag de förväntat sig att få för att sedan jämföra med resultatet från körningen av algoritmen.

Möjligheten att utvärdera kontaktförslagen utifrån sparad tid då en användare direkt tar kontakt med ett av kontaktförslagen istället för söka upp en kontakt i kontaktlistan och sedan kontakta den bedömdes vara en omöjlighet i utformningen av intervjun. Intervjuer ger också nya perspektiv på framtida utvecklingsmöjligheter och skapar en bättre förståelse av hur kontaktförslagen används idag.

Testgrupperna bestod av anställda från två avdelningar och anledningen var att få fram perspektiv från olika typer av användare. Från utvecklingsavdelningens håll var förväntningen att få fram förslag på förbättringar på algoritmen och synpunkter på koden, medan den främsta anledningen till att utföra intervjuerna på anställda på säljavdelningen var att få åsikter från personer som inte jobbar med något tekniskt och som inte är inblandade i

utvecklingen över huvud taget. Dessutom använder utvecklingsavdelningen nästan uteslutande applikationens chattfunktion medan säljavdelning har en bredare användning och nyttjar både chatt och samtal i sitt arbete.

Intervjuerna utfördes i form av öppna riktade intervjuer [9], vilket innebär att en intervju utgår från en mängd fastställda frågor, men att ordningen mellan dessa inte spelar någon roll och att man kan hoppa fritt mellan dem. Frågorna och de olika kontaktförslagen och kontaktmetoderna som tagits fram för alla deltagande skickades ut i förväg så att de deltagande skulle få tillräcklig tid att reflektera över resultatet och sina svar. Intervjuerna skedde över videokonferens på Telavox applikation och varje intervju varade cirka femton minuter. Under intervjuerna delades arbetet upp mellan de två författarna så att den ena gick igenom frågorna och konverserade med den deltagande och den andra antecknade vad som sades. Anteckningarna sammanställdes i ett dokument för att få en bättre överblick av resultatet av intervjuerna.

Intervjuer med användare utfördes först bland de anställda på utvecklingsavdelningen och där uppdagades en brist i den data som hämtades från databasen. De samtal och meddelanden som hämtades sträckte sig bara en vecka tillbaka i tiden, medan den kod som skrivits var utformad för att använda sig av data från en månad tillbaka. Detta problem upptäcktes efter att många av de deltagande på utvecklingsavdelningen kritiserat de kontaktförslag som algoritmen valt ut. Detta åtgärdades genom att byta till en databas som innehöll all data som algoritmen behövde.

Efter detta utfördes intervjuer av användare på säljavdelningen och där ströks frågan om vad som skulle kunna förbättras i algoritmen eftersom de anställda där inte förväntades kunna läsa kod. Förslagen på kontakter i detta testet upplevdes av de deltagande som betydligt mycket bättre än det som rapporterades i föregående test, och inga av förslagen ansågs som helt orimliga.

Utvecklingsavdelningen kom med synpunkter på framtida utvecklingsmöjligheter. Det vanligaste förslaget på förbättring var att algoritmen även ska betrakta ett videosamtal som en interaktion eftersom detta är något som används flitigt på Telavox i utvecklingsavdelningen. Videosamtal är en relativt ny kontaktmetod på Telavox applikation, i skrivande stund finns det bara en betaversion av videosamtal som endast är tillgänglig för utvecklingsavdelningen därmed inkluderades inte videosamtal-interaktioner i förslagsalgoritmen.

5 Resultat

I detta kapitel beskrivs examensarbetets resultat. Det första delkapitlet sammanfattar intervjuer med användare från säljgruppen och utvecklingsgruppen. Det andra kapitlet beskriver det slutliga resultatet av prototypen av förslagsalgoritmen i Telavox System.

5.1 Sammanfattning av intervjuer

Här ges en sammanfattning av hur de deltagande i intervjuerna från båda avdelningarna svarade på intervjufrågorna.

1: Tycker du att de fem kontaktförslagen som väljs ut av algoritmen känns rimliga med avseende på hur du brukar kommunicera med dina kontakter? Om inte, varför?

På den första frågan svarade samtliga deltagande i utvecklingsgruppen att de flesta kontaktförslagen upplevdes som rimliga, men att en eller två av dem upplevdes som orimliga. Detta visade sig, som beskrivits tidigare, bero på att den data som hämtades inte var från tillräckligt långt tillbaka i tiden. När detta hade åtgärdats och intervjuerna utfördes på säljgruppen svarade de deltagande där att samtliga kontaktförslag upplevdes som rimliga utifrån hur de kommunicerat de senaste veckorna.

En sak som deltagande i båda testgrupperna uttryckte var att de inte använder sig av förslagsfunktionen, och några av dem visste inte om att den fanns över huvud taget.

En anmärkning som kom från en av de deltagande i utvecklingsgruppen var att även om förslagen var rimliga sett utifrån den senaste tidens kommunikation, så finns det ändå annan funktionalitet i applikationen som ser till att de personer som en användare interagerar mest med finns nära till hands. Det finns en favoritlista som fungerar så att en användare kan markera vissa av sina kontakter som favoriter så att dessa kan nås snabbare. Dessutom är applikationens chattfunktion implementerad så att de senaste meddelandena visas längst upp, vilket gör att det är hög sannolikhet att de kontakterna som valt ut som förslag också visas långt upp bland meddelandena.

2: Tycker du att förslagen på kontaktmetoder för var och en av kontakterna speglar hur du brukar interagera med denna person? Om inte, varför?

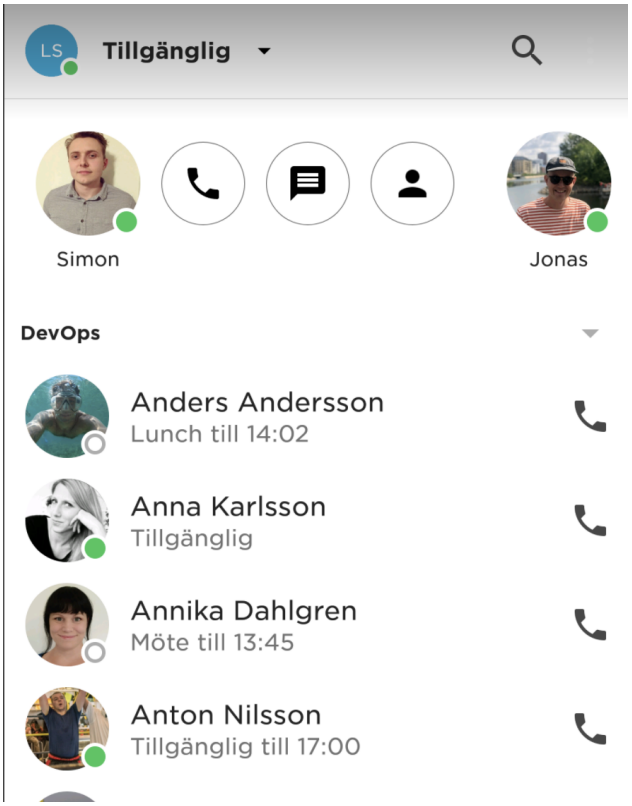
De deltagande i utvecklingsgruppen svarade samtliga att förslaget på kontaktmetod var rimligt utifrån deras förväntningar. Alla i gruppen fick förslaget chatt för alla kontakter som tagits fram av algoritmen, vilket är ett rimligt resultat eftersom de i denna grupp nästan uteslutande använder sig av chatt eller videosamtal när de kommunicerar med sina kollegor via applikationen. Några i denna grupp sade att de också ringer en del, men att samtalen i så fall går till folk utanför Telavox. Som algoritmen är implementerad beaktas enbart kommunikation med kollegor. Detta täcker in den större delen av användningen av applikation då applikationen är främst ett verktyg för intern kommunikation i ett företag.

Bland de deltagande i säljgruppen var åsikterna om de rekommenderade kontaktmetoderna mer spridda. Vissa tyckte att alla var rimliga, medan andra hade förväntat sig att få telefonsamtal som rekommenderad kontaktmetod för vissa kontakter när de istället fick chatt. En del av förklaringen till detta ligger i att de flesta samtal som de i säljgruppen ringer till varandra inte går via Telavox applikation, utan att de då istället ringer till sina vanliga telefonnummer. Den typen av samtal registreras inte av Telavox.

3: Är det nödvändigt att ge förslag på kontaktmetoder, eller hade det räckt att enbart rangordna kontakterna?

Angående om det var nödvändigt att ge förslag på kontaktmetoder var de generella åsikterna i de två grupperna skilda från varandra. De i utvecklingsgruppen var mer positiva till att ge förslag på kontaktmetoder, medan de i säljgruppen generellt sett tyckte att det hade räckt med att bara rangordna kontakterna.

Flera deltagande i utvecklingsgruppen uttryckte att det var oklart hur förslaget på kontaktmetod skulle presenteras i användargränssnittet, och detta var heller inte något som författarna av examensarbetet hade dragit någon slutsats om. Som Telavox applikation fungerar i skrivande stund dyker två symboler upp när man klickar på en användare i förslagslistan där den vänstra kopplar upp ett samtal med kontakten och den högra leder till ett chattrum med kontakten:



Figur 7: *Skärmdump av Telavox applikation som visar delar av användargränssnittet och hur kontaktförslag presenteras i applikationen*

Även om det var oklart hur förslaget på kontaktmetod skulle presenteras hade många i utvecklingsgruppen ideer om hur det skulle kunna göras. En idé var att ordningen mellan kontaktmetoderna skulle baseras på resultatet från algoritmen så att den rekommenderade kontaktmetoden skulle hamna längst till vänster. En annan idé var att på något sätt markera den rekommenderade kontaktmetoden på ett speciellt sätt.

De deltagande i säljavdelningen kom dock inte med några tankar om hur resultatet skulle presenteras, utan flera hade istället en bild av att algoritmen skulle begränsa valmöjligheterna så att enbart den rekommenderade kontaktmetoden skulle dyka upp när man klickar på någon i förslagslistan.

4: Hur hade algoritmen kunnat förbättras?

Frågan om hur algoritmen skulle kunna förbättras ställdes enbart till de deltagande i utvecklingsgruppen. Ett av de vanligaste förslagen var att videosamtal är något som borde räknas med vid beräkningen av rangordningen eftersom detta är något som utvecklingsgruppen använder sig väldigt mycket av. Många av de som föreslog detta tyckte även att videosamtal borde kunna rekommenderas som kontaktmetod.

Ett annat vanligt förslag var att algoritmen inte enbart borde baseras på interaktioner mellan två personer, utan att även ta med gruppchatter i beräkningen. Dessa gruppchatter skulle då också kunna ges som kontaktförslag. Motiveringen till detta var att gruppchatter är något som används väldigt ofta internt på utvecklingsavdelningen och att de måste beaktas för att få en bra bild av hur en användare vanligtvis kommunicerar genom applikationen.

Ett ytterligare förslag som flera deltagande kom med var att telefonsamtalen borde viktas olika beroende på hur långa de är istället för att tilldela alla samma värde. Detta förslag motiverades med att det är orimligt att ett samtal som enbart varar några sekunder har samma värde som ett som varar flera minuter.

En av utvecklarna kom med förslaget att chattmeddelandena skulle kunna viktas olika beroende på om de skickats av en användare eller om de blivit mottagna från någon annan. Anledningen till detta förslag var att vederbörande fått flera kontaktförslag som denne mottagit många meddelanden ifrån, men som denne inte skickat många meddelanden till. Lösningen till detta vore vikta skickade meddelanden tyngre än mottagna.

En annan utvecklare kom med ett förslag på förbättring som har med kontakternas status att göra. I applikationen kan ställa in sin status till “stör ej” eller “möte till klockan xx:yy” och förslaget var att algoritmen borde beakta detta eftersom det är orimligt att få ett kontaktförslag på en person som har markerat sig själv som upptagen.

5: Tycker du att algoritmen borde användas i Telavox applikation istället för den nuvarande?

Sju av åtta på utvecklingsavdelningen tyckte att algoritmen är bättre än de som används idag. Som förslagslistan fungerar i skrivande stund finns det två olika implementationer på mobil- och webbappen. På mobilappen bestäms rangordningen utifrån hur många samtal en användare ringt till sina kontakter, och i webbappen bestäms den utifrån vilkas profiler man gått in på. Den generella åsikten på utvecklingsavdelningen var att den algoritm som implementerats under examensarbetet kom med bättre förslag än de två andra och att det hade varit bättre att använda samma algoritm på alla plattformar.

Vissa på utvecklingsavdelningen var dock mer tveksamma till om algoritmen skulle användas. En anledning till detta var att vissa av förslagen upplevdes som märkliga, vilket berodde på problemet med databasen. En annan anledning var att vissa i utvecklingsgruppen sade att de inte använder sig av förslagslistan och att det således vore ointressant för dem om algoritmen implementeras eller ej.

I säljgruppen var de deltagande mer eniga om att de tyckte att algoritmen borde användas. Samtliga tyckte att de förslag som algoritmen kom med var bättre än de som de nuvarande gav. Förslagen från de algoritmer som används i skrivande stund upplevdes generellt sett som väldigt märkliga medan förslagen från CP-Rank upplevdes som betydligt rimligare.

5.2 Prototyp

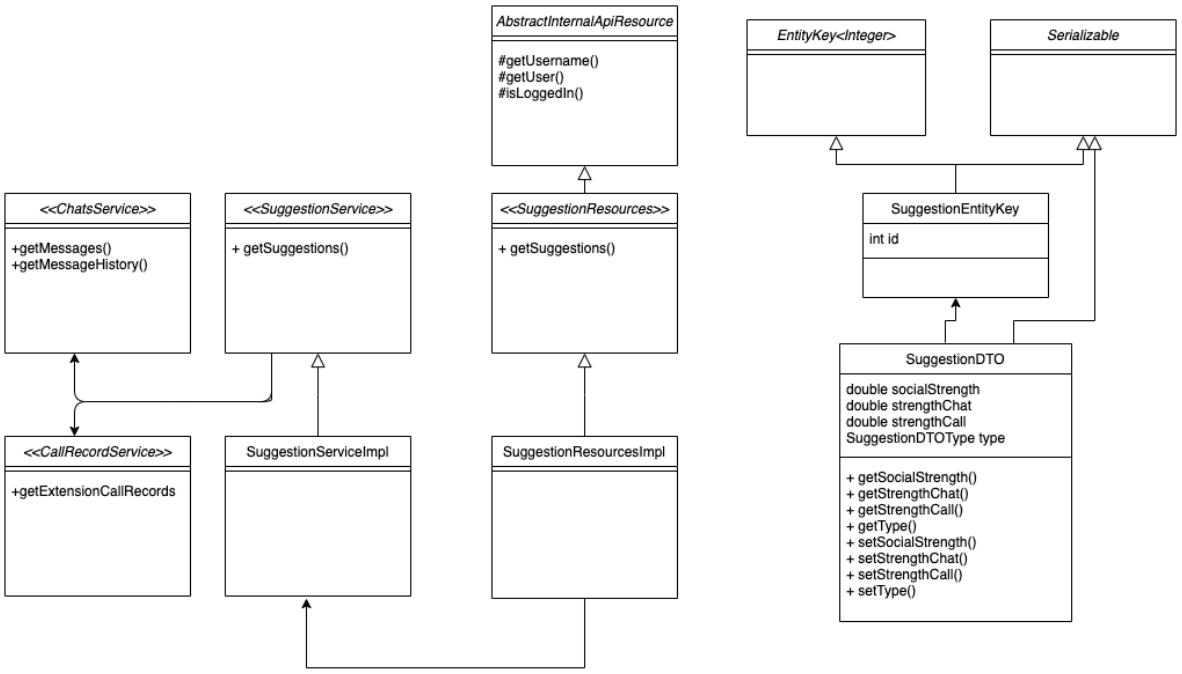
I detta delkapitel beskrivs prototypen som utvecklats under examensarbetet. Beskrivningen ges i form av ett klassdiagram se *figur 10*, ett sekvensdiagram som beskriver vad som händer i Telavox system när algoritmen körs (se *figur 11)* samt bilder på den kod (se *figur 8 & 9)* som skrevs för uträkningen av social strength och beräkningen som använde sig av markovkedjor. Klassdiagrammet i figur 10 för prototypen är modellerat efter klassdiagrammet för favoritlistan i figur 6. Sekvensdiagrammet i figur 11 är en förenklad modell av hur flödet av förslagsalgoritmen fungerar i Telavox System.

```
Set<Integer> contactIDList = new HashSet<>();
contactIDList.addAll(chatInteractionMap.keySet());
contactIDList.addAll(callRecordMap.keySet());


int nbrCalls = 0;

for (Map.Entry<Integer, Integer> pair : callRecordMap.entrySet()) {
    nbrCalls += pair.getValue();
}

int nbrChats = 0;

for (Map.Entry<Integer, Integer> pair : chatInteractionMap.entrySet()) {
    nbrChats += pair.getValue();
}
double threshCall = (double) nbrCalls / (nbrCalls + nbrChats);
double threshChat = (double) nbrChats / (nbrChats + nbrCalls);

for (int contactId : contactIDList) {
    double strength = 0.0;

    //Calculates strength for 1 contact
    int calls = 0;
    int chats = 0;

    if (callRecordMap.containsKey(contactId)) {
        calls = callRecordMap.get(contactId);
        strength += calls * threshCall;
        strength /= nbrCalls;
    }
    if (chatInteractionMap.containsKey(contactId)) {
        chats = chatInteractionMap.get(contactId);
        strength += chats * threshChat;
        strength /= nbrChats;
    }
    double strengthChat = 0;
    double strengthCall = 0;


    if (chats > 0)
        strengthChat = (double) chats / (chats + calls);

    if (calls > 0)
        strengthCall = (double) calls / (calls + chats);

    SuggestionDTO suggestionDTO = new SuggestionDTO();
    suggestionDTO.setStrength(strength);
    suggestionDTO.setStrengthChat(strengthChat);
    suggestionDTO.setStrengthCall(strengthCall);
    contactDTO = new ContactDTO();
    contactDTO.setKey(new ContactEntityKey(contactId));
    suggestionDTO.setContact(contactDTO);
    suggestionDTOList.add(suggestionDTO);

}
```

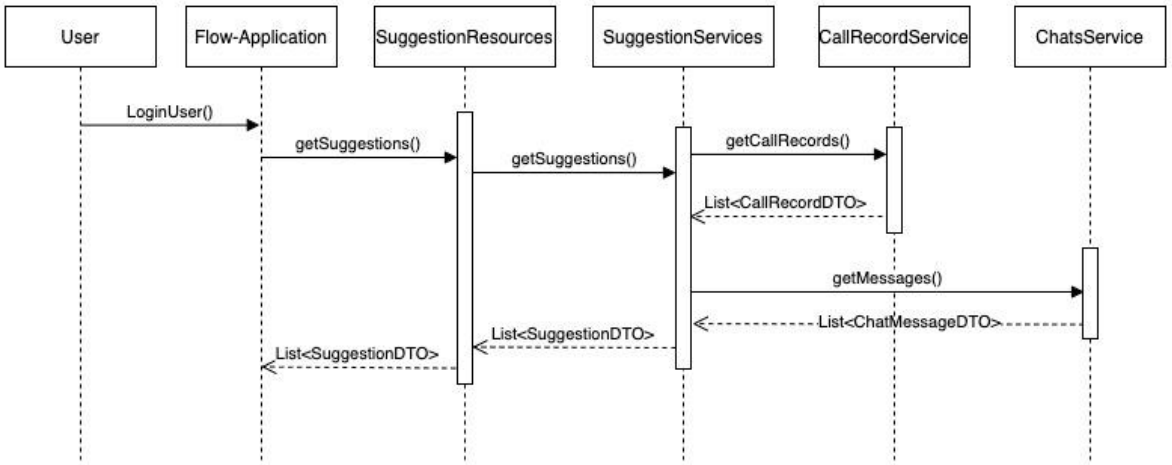
Figur 8: *Koden för Social strength uträkningen. Koden används i metoden getSuggestions() i klassen SuggestionService se klassdiagrammet i figur 10.*

```
    double[][] matrix = new double[16][16];
    int col = 6;
    for (int row = 1; row < 6; row++) {
        SuggestionDTO suggestionDTO = suggestionDTOList.get(row - 1);
        matrix[0][row] = suggestionDTO.getStrength();
        matrix[row][col] = suggestionDTO.getStrengthChat();
        col++;
        matrix[row][col] = suggestionDTO.getStrengthCall();
        col++;
    }
    for (int i = 6; i < 16; i++) {
        matrix[i][i] = 1;
    }

    double[] res = new double[16];
    double[] vec = new double[16];
    vec[0] = 1.0;
    double tmp = 0.0;

    for (int k = 0; k < 2; k++) {
        for (int i = 0; i < 16; i++) {
            tmp = 0.0;

            for (int j = 0; j < 16; j++) {
                tmp += vec[j] * matrix[j][i];
            }
            res[i] = tmp;
        }

        for (int i = 0; i < 16; i++) {
            vec[i] = res[i];
        }

    }

    int ctr = 0;
    for (int i = 6; i < vec.length - 1; i += 2) {...}

    return suggestionDTOList.subList(0,5);
}
```

Figur 9: *Koden för skapande av matrisen till markovkedjor och matrismultiplikationer. Koden används i metoden getSuggestions() i klassen SuggestionService se klassdiagrammet i figur 10.*



Figur 10: *Klassdiagram för resources & services för kontaktförslags-klasser*



Figur 11: *Sekvensdiagram för dataflödet när en användare loggar in på Flow-applikationen och kontaktförslagslistan uppdateras*

När en användare loggar in och autentiseras så anropar Flow-applikationen en uppdatering av förslagslistan. Detta är bara ett exempel på när uppdateringen av förslagslistan kan ske. Flow-applikationen kallar på data från API-endpoint:en SuggestionResources i resurslagret som i sin tur anropar SuggestionServices som nyttjar CallRecordServices och ChatsService för att hämta all nödvändig data till algoritmen som exekveras efter att datan har hämtats. Efter att algoritmen är färdig och har tagit fram fem kontaktförslag med kontaktmetod så paketeras förslagen i SuggestionDTO:er innan de returneras till SuggestionResources. Där får applikationen får fram kontaktförslag genom att SuggestionDTO:erna konverteras till JSON-Objekt innan applikationslagret kan presentera kontaktförslagen för användaren.

6. Slutsatser

Valet av algoritm att implementera föll på CP-rank-algoritmen eftersom den uppfyllde de krav som sattes upp tillsammans med handledarna på Telavox angående applicerbarhet, effektivitet och möjlighet att förstå den. De andra algoritmerna som utvärderades, PageRank och den genetiska algoritmen gick inte att direkt applicera på problemet som skulle lösas med att rangordna en användares kontakter. De gick inte heller att modifiera på något lämpligt sätt. CP-rank-algoritmen var den enda algoritm som hittades under informationsinhämtningen som var utvecklad för att rangordna en användares kontakter i en telekommunikations-applikation. Att det inte gick att hitta någon annan algoritm utarbetad med det syftet var något som först förvånade, men som vid närmare eftertanke kändes rimligt med tanke på att det främst är privata företag så som Facebook som har intresse av att använda sig av sådan algoritmer och med tanke på sådana algoritmer är företagshemligheter. Men även om det hade varit intressant att ha tillgång till flera sådana algoritmer för att jämföra med CP-rank-algoritmen kändes den ändå i sig som ett bra val eftersom den gav en tydlig lösning till problemet som skulle lösas. Att algoritmen, förutom att rangordna kontakterna, också ger ett förslag på kontaktmetod för var och en av dem var något som väckte stort intresse hos Telavox eftersom det sågs som en bra utveckling på förslagslistan gentemot hur den fungerar i skrivande stund.

Implementationen av algoritmen utvärderades genom att utföra intervjuer på anställda på Telavox. Innan intervjun fick de deltagande ta del av resultatet från en körning av algoritmen i form av en lista på deras fem högst rangordnade kontakter och tillhörande kontaktmetoder. De deltagande fick svara på ett antal frågor om hur de upplevde resultatet. Intervjuerna utfördes i två omgångar på två grupper anställda. Den första gruppen bestod av anställda på utvecklingsavdelningen och den andra bestod av anställda på säljavdelningen. Intervjuerna utfördes först på de anställda på utvecklingsgruppen och den första frågan som ställdes till de deltagande var om de upplevde de kontakter som valts ut som rimliga sett utifrån hur de kommunicerat med applikationen under den senaste tiden. På den frågan var det generella svaret att de flesta kontaktförslag upplevdes som rimliga, men att en eller flera av dem ansågs som orimliga. Detta resultat visade att något inte stod rätt till och det visade sig att den databas som algoritmen kördes mot inte hade data från en månad tillbaka som algoritmen var anpassad efter, utan bara från en vecka tillbaka. När intervjuerna sedan utfördes på de anställda på säljavdelningen hade detta åtgärdats och när samma fråga ställdes till dem svarade samtliga att de kontaktförslag som valt ut upplevdes som rimliga.

En annan fråga som ställdes var om de kontaktmetoder som valts ut för de olika kontaktförslagen upplevdes som rimliga och från de deltagande i utvecklingsgruppen var det

generella svaret att så var fallet, men samtliga där fick chatt som rekommenderad kontaktmetod för alla sina kontakter eftersom de nästan aldrig ringer till sina kollegor med applikationen. På säljavdelningen är det vanligare att de anställda ringer varandra och där svarade flera att de hade förväntat sig att få telefonsamtal som rekommenderad kontaktmetod på en del kontakter där resultatet blev chatt. Att de svarade så gav upphov till en tanke om att samtalen kanske borde ha viktats på ett annat sätt. Kopplat till frågan var en annan fråga om det över huvud taget var nödvändigt att rekommendera kontaktmetoder. På utvecklingsavdelningen var man generellt sett positiv till detta, men på säljavdelningen var folk mer tveksamma till om det behövdes. Att rekommendera en kontaktmetod sågs generellt sett där som ett sätt att begränsa valmöjligheten.

Från utvecklingsgruppen kom flera förslag om hur algoritmen kunde förbättras. Ett av de vanligaste förslagen var att gruppchatter borde tas med i beräkningen och att de borde kunna ges som ett kontaktförslag. Ett annat vanligt förslag var att videosamtal borde tas med i beräkningen av rangordningen och att det är något som borde kunna ges som rekommenderad kontaktmetod.

Under intervjuerna fick de deltagande också svara på frågan om de tyckte att algoritmen borde användas i Telavox applikation. På utvecklingsavdelningen gick åsikterna om detta isär. Vissa ansåg att den borde användas eftersom förslagen den gav upplevdes som en förbättring, medan de mer negativt inställda generellt sett motiverade sitt svar med att vissa av förslagen upplevdes som märkliga på så sätt att förslagen inte var personer de brukade kommunicera med eller med att en ny algoritm inte är nödvändig eftersom de inte använder sig av förslagslistan. På säljavdelningen var de deltagande mer positiva och tyckte generellt sett att algoritmen borde användas.

Angående algoritmens exekveringstid är den i sig inte särskilt krävande att köra, men den baseras på information som inte gick att hämta på något enkelt sätt. Den data som behövs är antalet interaktioner som en användare haft med var och en av sina kontakter via olika kommunikationsverktyg. Om detta var något som hade kunnat hämtas direkt från en databastabell hade datainsamlingen gått väldigt mycket snabbare än vad blev fallet. Informationen hämtades istället genom användning av metoder i klasserna ChatsService och CallRecordsService som returnerade listor med information om chattmeddelanden och telefonsamtal. Den mesta av informationen som dessa listor innehöll var onödig sett till det som behövdes för algoritmen, så det var ett väldigt ineffektivt sätt att hämta det på. Men trots ineffektiviteten i sättet att hämta datan på påverkade detta såklart inte resultatet som körningarna av algoritmen gav. Den kod som författarna skrivit hade inte kunnat användas i

Telavox applikation, men den var ändå tillräcklig för att utvärdera de förslag som algoritmen gav. Att ändra sättet att hämta data på tror författarna av detta examensarbete hade varit relativt enkelt för de anställda på Telavox ifall de skulle vilja använda sig av algoritmen.

En fråga som författarna ställde sig själva under examensarbetets gång var om komplexa algoritmer för att ta fram kontaktförslag är effektiva. Båda författarna använder sig flitigt av Facebook Messenger och anser att förslagslistan som finns där ger många väldigt märkliga förslag. Hur algoritmen bakom den förslagslistan fungerar är inte offentlig information, men det kan antas att mer tid och arbete lagts ner på den än vad som gjorts på detta examensarbete. Därför kan man betvivla om en mer komplex algoritm ger ett bättre resultat än en enklare lösning. En sådan lösning som är enklare än algoritmen som valdes ut hade kunnat vara att basera rangordningen av kontakterna på hur många gånger en användare interagerat med dem och inte utföra mer beräkningar efter detta. Resultatet från en sådan lösning hade antagligen varit ganska likt resultatet från den algoritm som valdes ut eftersom den i princip rankar de kontakter högst som en användare interagerat mest med. Genom att utgå från en sådan lösningsmetod hade mer tid kunnat läggas på andra saker såsom att filtrera bort kontakter som ställt in sin status till “stör ej” i applikationen eller kontakter som är markerade som favoriter.

6.1. Svar på frågorna från problemformuleringen

I början av examensarbetet formulerades ett antal frågor, se avsnitt 3.4. Nedan följer svar på och tankar om dessa frågor:

Hur fungerar algoritmen nu?

I början av examensarbetet antogs det att samma algoritm för att välja ut kontaktförslag användes på samtliga plattformar som Telavox applikation finns tillgänglig på. Detta visade sig inte stämma, utan den algoritm som används i mobilapplikationen är inte samma som den som används i webbapplikationen. I mobilapplikationen baseras rangordningen av kontakterna på hur många gånger en användare ringt dem, medan kontaktförslagen i webbapplikationen är de kontakter vars profil en användare besökt senast. Den senast besökta kontakten hamnar i webbapplikationen överst i förslagslistan, den näst senast besökta kontakten näst överst i lista och så vidare.

Vilka algoritmer finns idag för att välja ut kontaktförslag i en kontaktlista?

CP-rank-algoritmen var den enda publicerade algoritmen utarbetad med syftet att ta fram kontaktförslag i en kontaktlista som påträffades under examensarbetet.

Facebook Messenger är ett exempel på en applikation som använder sig av kontaktförslag, men den algoritmen är inte publicerad.

Vilka kriterier ska användas för att välja algoritmen?

Följande kriterier användes för att välja en algoritm:

- Algoritmen ska vara direkt applicerbar på det problem som ska lösas
- Algoritmen ska kunna implementeras utan att utgöra en onödig belastning på Telavox servar
- Algoritmen ska vara enkel att förstå
- Algoritmen ska enbart beakta interaktioner med en användares egna kontakter

Hur ska förslagslistan i prototypen utvärderas i förhållande till kriterier som kommer att formuleras under examensarbetet?

Förslagslistan utvärderades genom att utföra intervjuer med användare av applikationen. Syftet med examensarbetet var att skapa en bättre förslagslista och huruvida den blev bättre eller ej bedömdes genom att beakta de generella åsikter som deltagarna i intervjuerna hade om den. Förslagslistan ansågs av författarna ha blivit bättre om de flesta deltagande tyckte att kontaktförslagen upplevdes som rimliga och om de flesta deltagande föredrog förslagslistan framför de som redan fanns.

6.2. Reflektion över etiska aspekter

I detta kapitel analyseras de etiska aspekter som examensarbetet har berört.

6.2.1. Sekretess

Eftersom Telavox applikation är ett kommunikationsmedium där användare utbyter privat information med varandra i form av text, tal och video är sekretess en viktig etisk aspekt.

Under utvecklingsfasens tidiga del testades algoritmen enbart på författarna själva, vilket kan sägas vara oproblematiskt ur en etisk synvinkel. När utvecklingsfasen närmade sig sitt slut fanns dock ett behov av att testa algoritmen på andra användare för att se hur de upplevde resultatet. I och med detta gav handledarna på Telavox författarna tillgång till ett system som möjliggjorde för författarna att med tillfälliga inloggningsuppgifter köra algoritmen på andras användare. Att köra algoritmen på en annan användare gav inte möjligheten att läsa deras meddelanden eller att lyssna på deras telefonsamtal i efterhand eftersom sådant sparas i krypterat format, men berörde ändå annan känslig information. Rangordningen som algoritmen resulterar i ger en bra bild av vilka en användare interagerat mest med, vilket är privat information. Dessutom ser man vilken kontaktmetod som valt ut för de olika kontakterna. På grund av detta togs ett besult om att enbart köra algoritmen på användare som gett tillåtelse i förhand. Först kördes den enbart på handledarnas konton, men sedan behövde den testas på fler personer i och med att intervjuerna satte igång. I det meddelande som skickades ut till anställda på Telavox för att fråga om de var intresserade av att delta i en intervju var det noga beskrivet vilken information algoritmen använde sig av och vad den resulterade i så att ingen som inte var bekväm med detta skulle bli vilseledd att delta.

En annan etisk aspekt gällande sekretess har att göra med datalagring. Eftersom det som algoritmen resulterar i är privat information om vilka personer en användare interagerat mest med och med vilken typ av kommunikationsverktyg måste detta lagras på ett säkert sätt. Vissa typer hade kanske föredragit att sådan information inte lagras över huvud taget eftersom sådant kan hamna i fel händer vid en databasläcka.

6.2.2. Samhällsnytta

Arbetet med att välja ut och implementera en algoritm som ger kontaktförslag var i grunden ett sätt att försöka effektivisera kommunikation mellan människor. Att ge relevanta förslag i en lång lista kontakter sparar tid för användare på så sätt att de slipper leta efter de kontakter som dyker upp där. Ju oftare det finns en kontakt på förslagslistan som en användare vill ha kontakt med desto bättre. Tidsbesparingen som en förslagslista ger vid enstaka användning är inte jättestor, men sett över en längre period kan det ändå bli en hel del tid sammanlagt. Den algoritm som implementerats under examensarbetet upplevdes ge bättre förslag än den som finns i Telavox applikation för närvarande enligt resultaten från intervjuerna. Om den används skulle det kunna leda till att användare av Telavox applikation får ett lite effektivare sätt att kommunicera, och samhällsnyttan ligger i just detta.

6.3. Framtida utveckling

Detta kapitel beskriver framtida utvecklingsmöjligheter som upptäcks under examensarbetets gång.

6.3.1. Viktning av samtal/meddelande

I den algoritm som valdes ut har en interaktion mellan en användare och en kontakt samma värde oavsett vilket kommunikationsverktyg som används. I implementationen av algoritmen innebär detta att ett textmeddelande har samma värde som ett telefonsamtal. Detta kan anses som orimligt eftersom det i ett telefonsamtal generellt sett uttalas fler ord än vad som finns i ett textmeddelande.

En möjlig vidareutveckling på algoritmen hade varit att låta värdet av ett samtal bero på dess längd samtidigt som ett textmeddelande fortfarande ges värdet 1. Samtalen hade till exempel då kunnat viktas genom att ett uppkopplat samtal ges värdet 1 för att sedan öka i värde för varje påbörjad minut. Sett utifrån den kod som skrevs under examensarbetet hade detta varit enkelt att implementera eftersom den data som samlades in om alla telefonsamtal i form av callrecordDTOs innehåller längden för ett samtal. Som beskrivet tidigare i detta kapitel var sättet att samla in data på inte särskilt effektivt, men det hade i detta fall kunnat optimeras genom att se till att ett samtals längd kan hämtas utan annan onödig information.

Ett annat sätt att vikta samtal och textmeddelanden mot varandra hade varit att låta värdena för textmeddelanden och telefonsamtal bero på ett beräknat genomsnitt av hur många ord de innehåller. Att beräkna det genomsnittliga antalet ord i ett textmeddelande i Telavox applikation hade kunnat göras genom att analysera ett stort antal meddelanden. Det genomsnittliga antalet ord i ett telefonsamtal hade kunnat uppskattas genom att beräkna genomsnittslängden för ett telefonsamtal för att sedan uppskatta hur många ord detta motsvarar baserat på ett genomsnitt av hur många ord en människa uttalar per minut. Om resultatet från sådana genomsnittsberäkningar hade visat att ett telefonsamtal i genomsnitt innehåller fem gånger så många ord som ett textmeddelande hade det vägts fem gånger tyngre.

Ett möjligt problem med att väga textmeddelandena och telefonsamtalen mot varandra baserat på hur många ord de innehåller är att de två olika sätten att kommunicera skiljer sig åt i innehåll. Naturligt tal innehåller ofta fler utfyllnadsord än textmeddelanden och mycket som sägs är inte viktigt ur en informativ synvinkel. Textmeddelanden brukar å andra sidan vara mer korta och koncisa. Sådant hade också behövts beaktas om den typen av viktning hade använts.

6.3.2. Anpassa algoritmen till Telavox applikation.

Många olika sätt att utveckla algoritmen på hade kunnat göras genom att anpassa den mer till Telavox applikation. De i utvecklingsgruppen som deltog i intervjuerna kom med en mängd förslag på sådant, och några av de förslagen sågs som särskilt intressanta av författarna.

Att inkludera videosamtal i beräkningen av rangordningen av en användares kontakter hade kunnat ge bättre kontaktförslag eftersom videosamtal är ett vanligt sätt att kommunicera på i Telavox applikation. Detta var inte något som kunde göras under examensarbetet eftersom det inte fanns någon färdig metod för att hämta den data som behövdes om en användares videosamtal. Om antalet videosamtal som en användare haft med sina olika kontakter hade sparats någonstans hade det varit väldigt enkelt att inkludera videosamtal i beräkningen.

En annan anpassning av algoritmen till Telavox applikation hade kunnat vara att beakta vilka kontakter som finns i favoritlistan. Favoritlistan består av en mängd kontakter som en användare själv valt ut, vilka sannolikt ofta är några av de som en användare interagerar mest

med. För att undvika att en kontakt förekommer i både förslagslistan och favoritlistan hade algoritmen kunnat ändras på så sätt att en kontakt som finns i favoritlistan inte ska kunna ges som kontaktförslag.

Ytterligare en vidareutveckling av algoritmen hade varit att anpassa förslagen efter vilken tid det är på dygnet. Detta hade varit intressant eftersom en sådan anpassning hade kunnat fånga mönster i vilka kontakter en användare är benägen att kommunicera med vid olika tidpunkter. Baserat på detta hade mer relevanta förslag kunnat ges. Anpassningen hade kunnat göras genom att dela upp ett dygn i ett antal tidsintervall för att sedan utföra beräkningarna baserat på den kommunikation som förekommit inom ett sådant intervall.

7. Terminologi

Affärslogik - Affärslogik är din applikationslogik. Koden som skapar eller manipulerar data i ett lager mellan databasen och användargränssnittet.

Social Strength - Ett mått på den sociala anknytningen mellan två kontakter

DTO - (Data Transfer Object) ett objekt som används för att skicka data mellan olika delar av Telavox system. Objektet har en generell struktur och kan t.ex representera ett meddelande eller ett samtal.

DAO - (Data Access Object) är ett designmönster som är en generell struktur på hur ett objekt kommunicerar med databaser. DAO erbjuder enkel datahantering utan att avslöja detaljer om databasen.

Genetisk Algoritm - En genetisk algoritm är en algoritm som efterliknar processen för naturligt urval. De hjälper till att lösa optimerings- och sökproblem. Algoritmer imiterar naturliga biologiska processer, såsom arv, mutation och naturlig selektion.

Home - Home är backend-delen av Telavox system.

Flow - Telavox molnbaserade kommunikationsplattform.

PageRank - Algoritmen googles sökmotor använder för att ranka sökresultat

CP-rank-algorithm - Contact prioritization algorithm

8. Referensmaterial

[1] The PageRank Citation Ranking: Bringing Order to the Web.
(Hämtad 2021-02-15)
http://ilpubs.stanford.edu:8090/422/

[2] Enriching and Simplifying Communication by Social Prioritization
(Hämtad 2021-02-25)
https://www.researchgate.net/publication/221273601_Enriching_and_Simplifying_Communication_by_Social_Prioritization

[3] Ranking Algorithm by Contacts Priority for Social Communication Systems
(Hämtad 2021-02-29)
https://www.researchgate.net/publication/220876679_Ranking_Algorithm_by_Contacts_Priority_for_Social_Communication_Systems

[4] Sequel Pro
(Hämtad 2021-03-16)
https://www.sequelpro.com/

[5] Postman The Collaboration Platform for API Development
(Hämtad 2021-03-16)
https://www.postman.com/

[6] Friend Recommendations in Social Networks using Genetic Algorithms and Network Topology
(Hämtad 2021-02-29)
https://www.cse.unr.edu/~sushil/pubs/newestPapers/2011/cec/cec2011v4.pdf?fbclid=IwAR20b5VrgOhcsU7-ylsMy3RTuvL1pQEpgGiLxTb-5T2DDcgNS2hgcKrvOzc

[7] Kanban and Scrum: making the most of both
(Hämtad 2021-02-21)
http://www.agileinnovation.eu/wordpress/wp-content/uploads/2010/09/KanbanAndScrum_MakingTheMostOfBoth.pdf

[8] Telavox | Telefoni, växeltjänst, chatt, möten och kontaktcenter
(Hämtad 2021-02-15)
https://telavox.com/sv/

[9] A. Lantz, ”Att välja intervjuform,” i Intervjumetodik, andra upplagan, Annika Lantz och Studentlitteratur, 2007, pp. 29-35.

9. Appendix

9.1 Matematisk notation

Här finns alla formler som används för att rangordna kontakterna och kommunikationsverktygen.

För att räkna ut social strength används följande formel:

$$S_{i,a} = \sum_{j=1}^{q} \frac{f_{i,j,a} * thres_{i,j}}{\sum_{k=1}^{n} f_{i,j,k}}$$

Formeln ger värdet på social strength för en kontakt för ett visst kommunikationsmedium *i*. I detta examensarbete används bara ett kommunikationsmedium, nämligen Telavox applikation. $f_{i,j,a}$ är antalet interaktioner som en användare haft med en viss kontakt via ett visst kommunikationsverktyg. $thresh_{i,j}$ är hur stor andel en användare använder sig av ett visst kommunikationsverktyg. Summan i nämnaren är det totala antalet interaktioner som en användare haft via ett visst kommunikationsverktyg.

$thresh_{i,j}$ räknas ut på följande sätt:

$$thres_{i,j} = \frac{\sum_{k=1}^{n} f_{i,j,k}}{\sum_{m=1}^{q} \sum_{k=1}^{n} f_{,i,m,k}}$$

När social strength räknats ut för alla kontakter och kommunikationsverktyg sätts dessa värden in i en övergångsmatris. Utifrån denna beräknas en sannolikhetsvektor P_f ut genom ekvationen:

$P_{f=} P_f P_{tr}$ där P_{tr} är sannolikhetsmatrisen

Sannolikhetsvektorn P_f definieras som: $P_f = \lim_{k \to \infty} (P(k) = P(0)P_{tr}^{k})$