Verleden Namen

Familieverbanden uit Genlias-data

Maarten Oosten

doctoraalscriptie

onder begeleiding van
Hendrik Jan Hoogeboom, Walter Kosters
en
Kees Mandemakers (IISG)

LIACS
Universiteit Leiden

augustus 2008

Inhoudsopgave

Inleiding

Van de meeste mensen uit het verleden kennen we alleen nog de namen. Op de geboorte-, huwelijks- en overlijdensakten van de burgelijke stand zien we er vele terug. We kunnen meer over de mens achter de naam te weten komen door alle akten waarop iemand voorkomt bijeen te zoeken en naast elkaar te leggen. Zo komt een summiere rode draad tot stand die ons inzicht geeft in het leven van vroeger. Zij vormen dan ook een waardevolle bron voor sociologisch en demografisch historisch onderzoek. Het Internationaal Instituut voor Sociale Geschiedenis (IISG) put uit die bron voor verschillende projecten, waaronder de Historische Steekproef Nederland (HSN). Het bijeenzoeken van de akten van één persoon is echter niet altijd gemakkelijk en vormt een technische uitdaging. Tijdens mijn afstudeerstage bij het IISG ben ik onder begeleiding van Hendrik Jan Hoogeboom en Walter Kosters van het LIACS die uitdaging aangegaan in samenwerking met Kees Mandemakers, hoofd van de HSN.

Doelstelling

Historisch demografisch onderzoek houdt zich bezig met het beantwoorden van vragen over het dagelijks leven in historische tijden. Denk daarbij aan vragen over werkomstandigheden, migratie, gezinsomstandigheden, sociale positie, patronen in huwelijksgedrag, carriëreontwikkeling en over de samenhang tussen al deze elementen. Voor dat soort onderzoek zijn momentopnamen zoals volkstellingen of individuele akten niet geschikt. Er is behoefte aan informatie over de levensloop van mensen, aan informatie over meerdere momenten van het leven van één persoon. Dat soort longitudinale informatie kan worden verkregen door verschillende huwelijksakten naast elkaar te leggen waarop dezelfde persoon terugkomt. Om zulke akten te vinden zullen we ze op persoonsniveau moeten koppelen. Dit *matchen* van huwelijksakten vormt de kern van dit project.

De HSN, een project van de KNAW (Koninklijke Nederlandse Akademie van Wetenschappen), heeft als doel om levensloopinformatie op grote schaal te verzameling en beschikbaar te maken voor onderzoek. Voor dit project zijn 78.000 representatieve mensen willekeurig geselecteerd uit de periode 1812 tot 1922 op basis van geboorteakten. De HSN verzamelt over hen zoveel mogelijke informatie over hun leven: naast de geboorteakte huwelijksakten, overlijdensakten, vermeldingen in de burgerlijke stand enzovoort. Daarnaast heeft de HSN een grote hoeveelheid digitale huwelijksakten tot haar beschikking gekregen van Genlias. Genlias is een samenwerkingsverband van vele archiefinstellingen in Nederland, die met behulp van talloze vrijwilligers geboorte-, huwelijks- en overlijdensakten hebben gedigitaliseerd. De huwelijksakten uit die collectie vormen het hart van dit project.

Het doel van dit onderzoek is dan ook om zo veel mogelijk koppelingen tussen huwelijksakten te maken. Daarbij spelen de namen op de akten een hoofdrol. Ze zijn het belangrijkste criterium om twee akten te matchen, maar ook het meest complicerende. Zo kan het zijn dat dezelfde persoon op meerdere akten vermeld

staat, maar niet met iedere keer precies dezelfde naam. Ik heb me erop toeglegd om deze koppelingen desalniettemin toch te kunnen maken.

Meer dan namen vergelijken

Niet alleen kan één en dezelfde persoon met verschillende namen op akten staan, ook kan één en dezelfde naam aan verschillende personen toebehoren. Dit maakt het matchen erg complex en het ontneemt ons iedere definitieve zekerheid of een match "waar" is.

Laten we als voorbeeld *Johanna Oude Elberink* uit Almelo nemen, van huwelijksakte 10, gedateerd 4 juni 1824 te Almelo. Het is de akte van haar huwelijk op zestienjarige leeftijd met *Gerardus Jansen*, geboren in 1796 in Almelo, 12 jaar ouder. Laten we deze huwelijksakte akte *A* noemen.

We nemen eerst een positief voorbeeld, waarbij naam en identiteit wel overeenkomen. We vergelijken akte *A* met akte *B*, nummer 56 van 31 december 1861 te Almelo. Op deze akte trouwt *Johannes Franciscus Severius Jansen*, geboren 1838, met zijn bruid *Appelona Maria Berbera Freeze*. De ouders van Johannes worden vermeld als *Gerardus Jansen* en *Johanna Oude Elberink*. Het betreft hier duidelijk onze Johanna, dezelfe persoon met dezelfde naam.

Op akte *C* worden de gevaren duidelijk. Op deze akte, nummer 20 van 19 september 1862 te Tubbergen, staat de moeder van de bruidegom vermeld onder de naam *Johanna Oude Elberink*. Toch moeten we al snel concluderen dat, al heet ze hetzelfde, het niet onze Johanna betreft. Ze is getrouwd met ene *Johannes Hilberink* en haar zoon, de bruidegom, is geboren in 1834. Als we akten *A* en *B* in ogenschouw nemen, sluit dit nagenoeg uit dat het dezelfde vrouw is, want daar is zij in zowel 1824 als in 1838 getrouwd met Gerardus Jansen.

Tenslotte treffen we een akte uit Zwolle aan, nummer 147 van 7 november 1861. Deze akte, akte *D*, is van het huwelijk van *Veronica Hermina Jansen*, geboren in 1832, met *Derk Scholte*. Als Veronica's ouders worden vermeld de heer *Gerardus Jansen* en mevrouw *Johanna Olde Elbering*. Het is aannemelijk dat dit dezelfde Johanna (en dezelfde Gerardus) is als op akte *A* en *B*. Helemaal zeker kunnen we het echter op basis van huwelijksakten niet weten.

We mogen in ieder geval concluderen dat we niet blind op de naam mogen vertrouwen bij het koppelen van de huwelijksakten. We zullen naar de context van jaartallen en familie moeten kijken en een manier vinden om namen op een intelligente manier met elkaar te vergelijken.

Uit deze voorbeelden wordt eveneens duidelijk dat we vele personen vermeld zien staan op de akten, waarvan er vele naar één werkelijk persoon verwijzen. We zullen dan ook onderscheid maken tussen de vermeldingen van personen en de personen zelf. Centraal in het koppelvraagstuk is dan ook de vraag of twee vermeldingen van personen (later zullen we dit persoonsknopen noemen) op dezelfde werkelijke persoon slaan.

Het verleden

Het werken met historische documenten is inspirerend. Tijdens het project heb ik vele namen voorbij zien komen en als het ware glimpen van huwelijken, gezinnen en levens van reeds lang gestorven mensen opgevangen. Dit besef van een uitgestrekt verleden dat achter ons ligt en van groot belang is voor het heden heeft bijgedragen aan de titel van deze scriptie[1].

[1] Het Van Dale-woordenboek van het hedendaags Nederlands definiëert *verleden* als volgt:

Tijdens dit project ben ik in gesprekken met vrienden en familie vaak de interesse in het verleden en in onze voorvaderen tegengekomen. Dat heeft er onder andere toe geleid dat mij een verzameling van genealogische informatie over mijn eigen familie is toevertrouwd. Die heb ik met veel plezier geraadpleegd op de momenten dat ik geen toegang had tot de databanken van de HSN. Wees daarom niet verbaasd als in dit document in bepaalde voorbeelden opeens de naam Oosten voorkomt.

- verleden, het
  de tijd die voorbij is
- verleden (bijv. naamwoord)
  in de tijd voorbij zijnd
- verlijden (overgankelijk werkwoord: verleed, heeft verleden)
  (een akte, testament, contract, enz.) opmaken en na ondertekening geldigverklaren

Hoofdstuk 1

De brondata

Voor het opbouwen van genealogieën wordt in dit project gebruik gemaakt van huwelijksakten. Het gebruik van huwelijksakten biedt verschillende voordelen. Ten eerste vermelden ze veel verschillende personen: de bruidegom, de bruid en van beide huwenden beide ouders. Dat is op zichzelf al een klein genealogietje, dat twee keer zo groot is als de genealogie die van bijvoorbeeld een geboorteakte afgeleid kan worden. Ten tweede komen de belangrijkste zes personen van een huwelijksakte paarsgewijs voor, in de vorm van drie echtparen[1]. Het betrouwbaar matchen[2] van echtparen is makkelijker dan het matchen van individuen. De kans dat twee echtparen waarvan de namen van beide echtelieden overeenkomen hetzelfde echtpaar zijn is namelijk vele malen groter dan de kans dat twee personen met dezelfde naam hetzelfde individu zijn. Tenslotte bevatten de huwelijksakten veel andere informatie over de vermelde personen, zoals geboortedata en beroepen. Niet alleen is die informatie van nut bij het matchingproces, hij is ook van waarde voor de sociaal-historici die voor hun onderzoek gebruik maken van de familienetwerken die het proces oplevert.

Genlias heeft aan de HSN zes collecties digitale huwelijksakten beschikbaar gesteld. Vijf daarvan, de sets waarmee mijn project zich met name heeft beziggehouden, bevatten de huwelijksakten van vijf verschillende provincies: Groningen, Overijssel, Gelderland, Zeeland en Limburg. De zesde bevat alle huwelijksakten uit de databank van Genlias bij elkaar, dus inclusief die van de vijf net genoemde, maar biedt minder informatie per akte. Tabel 1.1 biedt een overzicht van de collecties. Deze datasets bevatten niet alle gegevens die op de daadwerkelijke huwelijksakten staan. Genlias heeft alleen de belangrijkste gegevens opgenomen. De vijf provinciedatasets zijn bijzonder omdat ze ook de geboortedata of leeftijden van de huwenden

[1] De beschikbare huwelijksakten zijn allemaal van tussen 1796 en 1922. In deze periode geldt voor de overgrote meerderheid van de vermelde ouders dat ze getrouwd waren. We nemen voor het gemak aan dat dit voor alle ouders geldt.

[2] Zie inleiding.

provincie	huwelijksakten	periode
Groningen	208.223	1811–1922
Overijssel	220.650	1811–1922
Gelderland	366.045	1811–1922
Zeeland	164.498	1796–1922
Limburg	189.687	1796–1922
Genlias-totaal	2.457.471	1796–1922

Tabel 1.1: Datasets

veld	beschrijving
aktenummer	het nummer van de akte, uniek per plaats en jaar
aktedatum	de datum waarop de akte is opgetekend
akteplaats	de plaats of gemeente waar de akte is opgetekend
aktetype	het type van de akte, zie paragraaf 1.1.3 op de pagina hiernaast
gegevens bruidegom	zie tabel 1.3
gegevens bruid	zie tabel 1.3
gegevens vader van de bruidegom	zie tabel 1.3
gegevens moeder van de bruidegom	zie tabel 1.3
gegevens vader van de bruid	zie tabel 1.3
gegevens moeder van de bruid	zie tabel 1.3
opmerkingen	In dit veld staat extra informatie die op de akten voor kan komen en niet in de andere velden past. Zie paragraaf 1.2 op pagina 12.

Tabel 1.2: De door Genlias opgenomen informatie over de huwelijksakten

veld	beschrijving	genoteerd voor
voornamen		huwenden en ouders
voorvoegsel	zoals “de”, “van” of “’t”	huwenden en ouders
achternaam	zonder voorvoegsel	huwenden en ouders
beroep		huwenden en ouders
geboorteplaats		huwenden
geboortedatum	Meestal wordt alleen de leeftijd genoteerd.	huwenden
leeftijd	in jaren	huwenden

Tabel 1.3: De opgenomen informatie over de personen op de huwelijksakten

bevatten en dikwijls de beroepen van de huwenden en hun ouders. Tabellen 1.2 en 1.3 bevatten een overzicht van de gegevens die in de collecties zijn opgenomen. In paragraaf 1.1 zullen we ze nader bekijken. Alle zes de datasets bevatten de velden die in de tabellen zijn aangegeven. De collectie Genlias-totaal vermeldt echter zelden leeftijden, geboortedata of beroepen en is dus minder bruikbaar. Ook is hij pas halverwege het project beschikbaar gekomen. De nadruk van het project ligt daarom op de eerste vijf sets.

De periode waarbinnen de huwelijksakten beschikbaar zijn heeft twee harde grenzen. Hij begint in 1811, wanneer Napoleon in Nederland de burgerlijke stand invoert en achternamen verplicht stelt. In België, Zeeuws-Vlaanderen en delen van Limburg gebeurde dit eerder, namelijk in 1796. De collecties Zeeland en Limburg bevatten dan ook akten van voor 1811, maar pas na 1811 beslaan de collecties de gehele provincies. Vóór de invoering van de burgerlijke stand werden huwelijk, doop en overlijden in kerkarchieven geregistreerd. Deze registers kennen weer hun eigen problemen, zoals het gebuik van patroniemen in plaats van achternamen, en vallen buiten het bereik van dit project.

De periode wordt afgesloten door de privacywetgeving. Deze stelt alleen geboorteakten ouder dan 100 jaar, huwelijksakten ouder dan 75 jaar en overlijdensakten ouder dan 50 jaar beschikbaar. Genlias heeft de huwelijksakten tot 1922 gedigita-

liseerd en aan HSN beschikbaar gesteld. Op dit moment is Genlias bezig met de huwelijksakten van 1922 tot 1930.

1.1 De gegevens op de akten

De akten bevatten veel informatie. De databestanden van Genlias bevatten slechts een deel daarvan, en ook wij maken weer een selectie. De volgende gegevens gebruiken we in dit project.

1.1.1 Aktenummer

Iedere huwelijksakte heeft een *aktenummer*. Elke gemeente begint ieder jaar de akten weer vanaf 1 te nummeren. Dit nummer is met name van belang bij de terugkoppeling van de database naar de daadwerkelijke akten en is verder niet relevant voor het matchen.

1.1.2 Aktedatum

De *aktedatum* is de datum waarop de akte door de ambtenaar van de burgerlijke stand is opgetekend en uitgegeven. We nemen voor het matchen aan dat dit ook de huwelijksdatum is. Uit de zeldzame akten waarop de huwelijksdatum apart vermeld staat kunnen we afleiden dat deze aanname terecht is.

1.1.3 Aktetype

In dit veld staat een letter die het type van de akte aangeeft, meestal een “H” voor huwelijksakten. Deze zijn in de meerderheid, maar er zijn ook andere akten die op huwelijksakteformulieren zijn ingevuld, zoals akten van wettiging (“W”), echtscheidingsakten (“E” of “S”) en naamswijzigingsakten (“N”). Die kunnen aan de hand van dit veld worden weggefilterd. Voor een klein deel van de akten is dit veld echter leeg. In dit deel komen alle soorten akten in dezelfde verhouding voor en zijn de meeste dus huwelijksakten. Daarom is het niet weggefilterd. Een gevolg is dat er sporadisch niet-huwelijksakten in de collecties voorkomen waarmee gematcht wordt.

1.1.4 Voornaam, achternaam en voorvoegsel

Voor alle zes personen op de akte is de gehele naam gesplitst in drie stukken. Het veld *voornamen* bevat alle voornamen. Het veld *achternaam* bevat de gehele achternaam, behalve de voorzetsels en lidwoorden waarmee de achternaam begint. Deze staan in het veld *voorvoegsel*. Zo wordt bijvoorbeeld de naam “Martinus Johannes van der Meij de Bie” als volgt opgesplitst:

voornamen:	“Martinus Johannes”
voorvoegsel:	“van der”
achternaam:	“Meij de Bie”

In sommige gevallen is deze spliting niet helemaal netjes uitgevoerd. Zo wordt meestal “d’” als voorvoegsel gezien, maar soms ook niet. Ook wordt nu en dan extra informatie toegevoegd zoals “Mr.” voor de voornaam of “baronesse” erachter. Wanneer dit niet consequent wordt gedaan op alle akten waarop de persoon in kwestie voorkomt, en dat wordt het meestal niet, bemoeilijkt dit het matchen van de akten. Gelukkig is het grootste deel van de akten vrij van dergelijke inconsistenties.

Voornamen, voorvoegsel en achternaam worden nauwelijks gewijzigd gebruikt voor het matchen. Bij alle drie de velden verwijderen we spaties aan het begin en aan het einde.

1.1.5 Beroep en geboorteplaats

Zoals de namen al zeggen worden in de velden *beroep* en *geboorteplaats* het beroep en de geboorteplaats genoteerd. Van deze gegevens wordt geen gebruik gemaakt bij het matchen. Voor het beroep geldt dat dit tijdens het leven vaak verandert en de geboorteplaats is voor de ouders niet genoteerd. Beide velden zijn wel van belang voor de outputbestanden en het onderzoek dat erop wordt gedaan.

1.1.6 Geboortedatum en leeftijd

Aan de hand van deze velden kunnen de geboortejaren van bruidegom en bruid worden bepaald, een belangrijk gegeven voor het matchingsproces. Voor nagenoeg alle akten uit de collecties van de vijf provincies is één van deze twee velden ingevuld. Helaas is dat niet overal even netjes gebeurd. Allerlei soorten slordigheden voorkomen bij vele akten het correct uitlezen van geboortejaar, -maand en -dag. Zo komt het veel voor dat de leeftijd is ingevuld in het geboortedatumveld of dat van het geboortejaar alleen de laatste twee getallen zijn genoteerd en de eeuwaanduiding ontbreekt. Dit laatste kunnen we oplossen door aan te nemen dat de bruid en bruidegom altijd meer dan 15 jaar en minder dan 115 jaar voor het huwelijk zijn geboren.

1.2 De opmerkingvelden

De opmerkingvelden bevatten allerlei extra informatie over de akten. Dit verschilt vaak per akte en loopt uiteen van namen van getuigen tot vermeldingen van de plaats waar het kerkelijk huwelijk is gesloten. Regelmatig komt voor dat geboortedata of leeftijden in dit veld zijn opgetekend. Waar mogelijk zorgen we ervoor dat die gegevens herkend worden en in de juiste velden worden opgenomen. De overige informatie laten we in dit project ongemoeid.

1.3 Normalisatie

Voordat we met de huwelijksakten aan de slag gaan, zetten we ze om in een uniforme structuur en slaan we ze op in onze eigen databank. We kiezen hiervoor een vorm waarin we ook geboorteakten, overlijdensakten, gezinskaarten, stambomen of andere willekeurige genealogische informatie kunnen zetten. Op die manier kunnen we dezelfde algoritmen gebruiken voor data uit verschillende soorten bronnen. Onze normaalvorm is analoog aan GEDCOM, een veelgebruikte bestandsstructuur door genealogen.[3] We bouwen een datastructuur op van gezinsknopen en persoonsknopen. De gezinsknopen representeren de eenheid van een traditionele[4] nucleaire familie en de persoonsknopen representeren de personen die in verband staan met deze gezinnen. Bij iedere gezinsknoop horen één echtgenoot en één echtgenote en

[3] GEDCOM staat voor GEnealogical Data COMmunication en is ontwikkeld door de Kerk van Jezus Christus van de Heiligen der Laatste Dagen (de Mormoonse Kerk).

[4] In deze struktuur passen alleen traditionele familieverbanden. We nemen aan dat gezinnen bestaan uit precies één mannelijke vader of echtgenoot, één vrouwelijke moeder of echtgenote en nul of meer kinderen, dat er geen ambiguïteit bestaat omtrent het geslacht van personen en dat er per kind één huwelijk is geweest waaruit het is voortgekomen. Voor modernere familieverbanden, zoals het homohuwelijk, zou deze structuur niet flexibel genoeg zijn, maar hij volstaat voor de periode waaruit onze gegevens komen.

een willekeurig aantal kinderen. Iedere persoon kan deel uitmaken van meerdere gezinnen, bijvoorbeeld bij de eerste als kind, bij een ander als echtgenoot en na een scheiding bij een derde eveneens als echtgenoot. Met deze struktuur kunnen uiteenlopende situaties gerepresenteerd worden. Er zijn wel twee logische beperkingen. Bij een gezinsknoop kunnen maar één echtgenoot en maar één echtgenote horen en iedere persoonsknoop kan als kind maar bij één gezinsknoop horen. Overigens echtelieden hoeven niet altijd toegevoegd te worden aan een gezinsknoop. Wellicht is de vader, de moeder of zijn beiden onbekend. In het laatste geval kan een gezinsknoop dienen om van de aangesloten kinderen aan te geven dat het broers en zussen zijn. Zo ook hoeft een persoonsknoop als kind bij geen enkele gezinsknoop te horen, wanneer er bijvoorbeeld over ouders, broers noch zussen informatie is.

Als we een huwelijksakte omzetten in deze struktuur, vormen de bruidegom en zijn ouders één gezin, de bruid en haar ouders een tweede gezin en wordt het derde gezin gevormd door de bruid en bruidegom als echtelieden. Merk op dat aan het derde gezin geen kinderen verbonden zijn en dat bij de ouders van de huwenden het gezin waar ze uit voortkomen niet bekend is.[5] Figuur 1.1 op de pagina hierna geeft deze en de volgende stap weer.

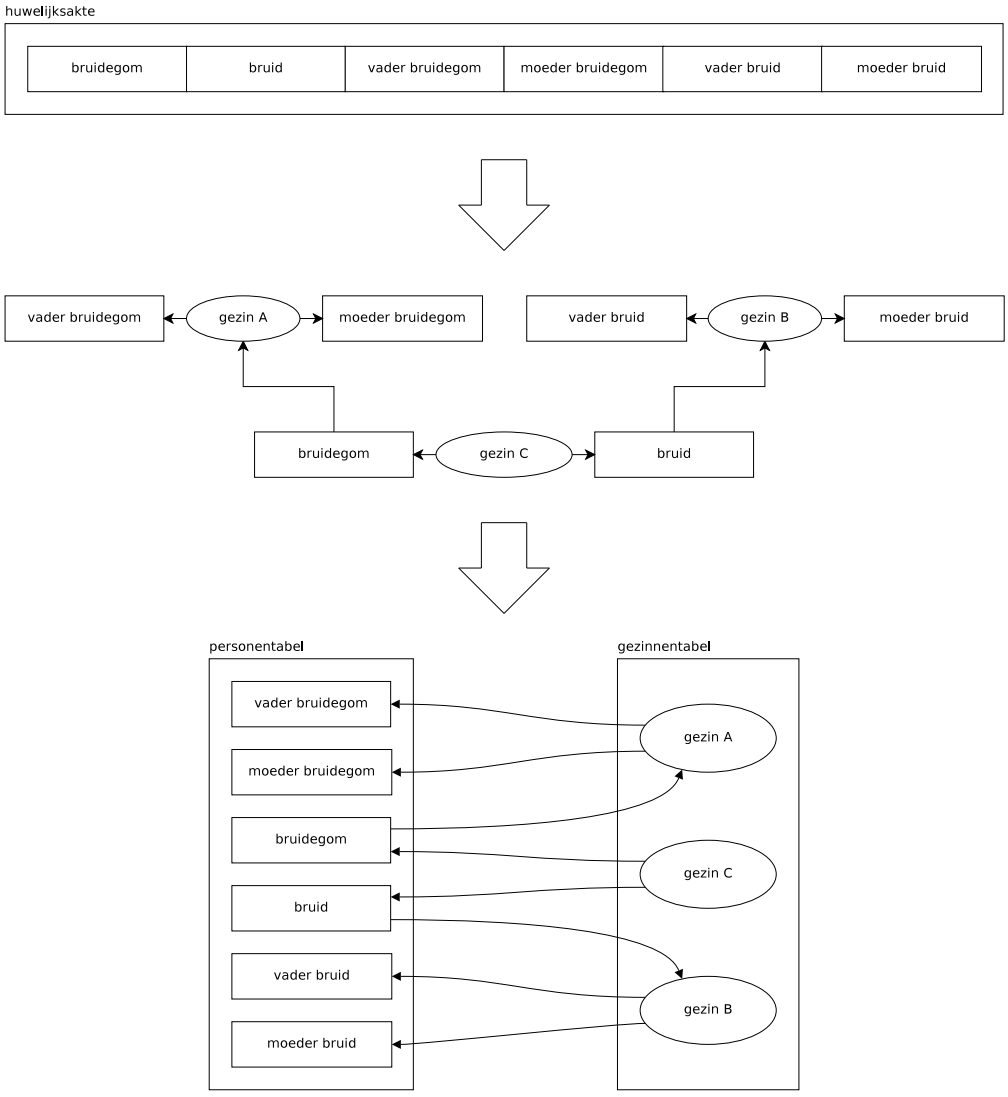
We slaan die struktuur op in onze databank in twee tabellen, P en G genaamd. In P worden alle persoonsknopen als records opgeslagen en krijgen ze een uniek identificatienummer, de *p-id*. In de records van P staan verder alle gegevens over de persoon en een referentie naar de originele akte. Er is voor iedere persoonsknoop een leeftijds- en een geboortedatumveld. Wanneer deze onbekende zijn voor een persoonsknoop, vullen we de speciale waarde `null` in.[6] Ook staat er een verwijzing naar de gezinsknoop waar de persoonsknoop als kind bij hoort, zo er één is. Deze verwijzing wordt bewerkstelligd door de *g-id* (zie hieronder) in de record op te slaan. Tenslotte wordt in iedere record vermeld welke rol de persoon vervulde op de huwelijksakte. Dit veld kan de volgende waarden aannemen: *bruidegom*, *bruid*, *vader bruidegom*, *moeder bruidegom*, *vader bruid* of *moeder bruid*. De heuristiek maakt hier gebruik van (zie hoofdstuk 5).

Voor alle gezinsknopen wordt een record in G aangemaakt. Ook deze krijgt een unieke identificatie, een *g-id*. In iedere record worden de *p-ids* opgeslagen van de echtgenoot en echtgenote in het gezin. Verder staat er een verwijzing naar de originele akte en wordt de huwelijksdatum opgeslagen voor de gezinsknopen van bruid en bruidegom. Ook hier wordt een veld genaamd *rol* gebruikt om aan te geven om welk huwelijk het gaat. Het kan de volgende waarden aannemen: *huwenden*, *ouders bruidegom* of *ouders bruid*.

Veel personen komen op meerdere huwelijksakten voor, met name als ouder. Zo kunnen per persoon meerdere persoonsknopen worden gemaakt, één per huwelijksakte waarop de persoon voorkomt. Persoonsknopen stellen dan ook niet personen voor, maar vermeldingen van personen op huwelijksakten. Hetzelfde gaat op voor gezinsknopen. Bij het matchen vergelijken we de persoonsknopen en gezinsknopen om te beoordelen of het vermeldingen zijn van dezelfde werkelijke personen en gezinnen.

[5] Hoewel in de meeste gevallen het huwelijk voorafgaat aan de komst van kinderen, is er een groot aantal akten waarbij vermeld wordt hoeveel kinderen gewettigd worden en een zeer klein aantal waarbij ook de namen van de kinderen bekend zijn. Die laatste zouden als kinderen onder het gezin van de huwenden kunnen worden gehangen, maar omdat het aantal akten met benoemde kinderen verwaarloosbaar is, is dat niet geïmplementeerd in dit project.

[6] De `null`-waarde is een concept dat in veel implementaties van SQL wordt gebruikt. De waarde is onvergelijkbaar met normale waarden. We zullen in dit document de functies isnull(x) gebruiken, die *waar* is als x een `null`-waarde is, en ifnull(x,y) die het eerste argument teruggeeft, behalve als dit `null` is, dan het tweede.



Figuur 1.1: De normalisatie van een huwelijksakte

1.3.1 Tabeldefinities

In dit document zullen we meerdere tabellen introduceren en gebruiken in de beschrijvingen van algoritmen. We stellen daarom een formele definitie van tabelstructuren voor die aan onze behoeften voldoet. We definiëren een tabelstructuur voor een tabel T als volgt:

$$T : (\overrightarrow{v}, \overrightarrow{p}, \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_m}\}, \{\overrightarrow{s_1}, \ldots, \overrightarrow{s_n}\}) \tag{1.1}$$

$\overrightarrow{v}$**:** een geordende rij veldnamen, de namen van de kolommen in de tabel

$\overrightarrow{p}$**:** de primaire sleutel van de tabel

$\overrightarrow{u_i}$**:** een unieke index op de tabel

m**:** het aantal unieke indices op de tabel (mag 0 zijn)

$\overrightarrow{s_i}$**:** een niet-unieke index op de tabel

n**:** het aantal niet-unieke indices op de tabel (mag 0 zijn)

We gebruiken hier de vectornotatie (het pijltje boven de letter) om aan te geven dat een verzameling lineair geordend is: de elementen in de verzameling hebben een vaste volgorde. Alle indices op de tabel zijn deelverzamelingen van $\overrightarrow{v}$. Dit geldt ook voor de primaire sleutel. Dat is het veld, of een combinatie van velden, waarmee in de databank records worden onderscheiden. Geen twee records in een tabel kunnen dezelfde waarden (of combinaties van waarden) hebben in de velden van de primaire sleutel. In de meeste gevallen zullen we records een uniek nummer geven, zoals *p-id* en *g-id* hierboven, om ze gegarandeerd uniek te maken en deze id gebruiken als primaire sleutel. Zo ook in onze eerste twee tabellen:

$$P : \left(\begin{array}{l} \text{p-id} \\ \text{aktenummer} \\ \text{aktedatum} \\ \text{akteplaats} \\ \text{voornamen} \\ \text{voorvoegsel} \\ \text{achternaam} \\ \text{geslacht} \\ \text{beroep} \\ \text{geboorteplaats} \\ \text{geboortedatum} \\ \text{leeftijd} \\ \text{rol} \\ \text{ouders-g-id} \end{array} \right), (\text{p-id}), \emptyset, \{(\text{ouders-g-id})\} \tag{1.2}$$

Zoals al beschreven is *p-id* de primaire sleutel. Daarnaast is er ook een index op *ouders-g-id*, zodat we efficiënt de kinderknopen van een gezinsknoop kunnen vinden. De structuur van tabel G volgt hetzelfde stramien:

$$G : \left(\begin{array}{l} \text{g-id} \\ \text{aktenummer} \\ \text{aktedatum} \\ \text{akteplaats} \\ \text{rol} \\ \text{man-p-id} \\ \text{vrouw-p-id} \end{array} \right), (\text{g-id}), \emptyset, \{(\text{man-p-id}), (\text{vrouw-p-id})\} \tag{1.3}$$

```
n ← 0
for each (m ∈ P : m(geslacht) = man) ∧ m(rol) = bruidegom)
  ⋈_{h(man-p-id)=m(p-id)} (h ∈ G)
  ⋈_{v(p-id)=h(vrouw-p-id)} (v ∈ P : v(geslacht) = vrouw ∧ v(rol) = bruid)
do
  if m(geboorteplaats) = v(geboorteplaats) then
    n ← n + 1
  fi
od
```

Figuur 1.2: Een voorbeeld van pseudocode

Ook hier gebruiken we twee indices om bij echtelieden snel de huwelijksknoop te kunnen vinden. Als een veld of combinatie van velden geïndiceerd is, worden de records op deze velden gecatalogiseerd. Er kunnen meerdere catalogi voor een tabel worden bijgehouden. Zoeken met behulp van een catalogus gaat veel sneller dan wanneer alle records bekeken moeten worden. Het bijhouden van een catalogus kost echter wel tijd, daarom maken we geen index voor alle kolommen. De primaire sleutel is de volgorde waarin de records opgeslagen worden. Op de primaire sleutel kan daarom altijd snel gezocht worden. Het is als het ware de volgorde waarin de boeken op de plank staan.

De tabellen zijn gevuld met rijen of records, die we formeel zullen beschouwen als partiële functies met de veldnamen uit de tabelstructuur als domein. Wanneer we naar de waarde van een veld in rij r willen refereren, zullen we dit uitdrukken als r(veldnaam). Voor het gemak zullen we in pseudocode toestaan dat we ook toewijzingen mogen doen aan deze uitdrukking om de waarde te wijzigen.

Figuur 1.2 bevat een paar regels pseudocode voor een simpel algoritme dat het aantal huwelijken telt tussen mensen die in dezelfde plaats zijn geboren. We gebruiken hier een *for each*-loop die itereert over alle tupels van een verzameling die vervolgens gedefinieerd wordt. We gebruik het $\bowtie$-symbool om een join in SQL-stijl aan te geven. We gebruiken hier een *θ-join*, ook wel *equijoin* of *inner join* genoemd, waarbij de θ-eis als index bij het joinsymbool genoteerd staat. We zullen ook veel gebruik maken van de *left outer join*, genoteerd met het symbool $=\!\times$. Bij deze join worden records uit de linker verzameling aan het resultaat toegevoegd ook als er geen record uit de rechter tabel voldoen aan de θ-eis. In dat geval wordt in plaats van een record uit de rechter tabel een lege partiële functie gebruikt ($\emptyset$), waarbij voor alle velden v geldt dat $\emptyset(v)$ = `null`. De `null`-waarde gedraagt zich zoals traditioneel in SQL.

Om de peudocode in de volgende hoofdstukken te versimpelen, definiëren we hier een aantal intuïtieve predikaten voor willekeurige records $a, b, k \in P$ en $h \in G$:

$$\begin{aligned}
\text{mangezin}(a,h) &\iff a(\text{p-id}) = h(\text{man-p-id}) && (1.4)\\
\text{vrouwgezin}(a,h) &\iff a(\text{p-id}) = h(\text{vrouw-p-id}) && (1.5)\\
\text{partnergezin}(a,h) &\iff \text{mangezin}(a,h) \vee \text{vrouwgezin}(a,h) && (1.6)\\
\text{manvrouw}(a,b) &\iff \exists h \in G : \text{mangezin}(a,h) \wedge \text{vrouwgezin}(b,h) && (1.7)\\
\text{getrouwd}(a,b) &\iff \text{manvrouw}(a,b) \vee \text{manvrouw}(b,a) && (1.8)\\
\text{kindgezin}(a,h) &\iff a(\text{ouders-g-id}) = h(\text{g-id}) && (1.9)\\
\text{vaderkind}(a,b) &\iff \exists h \in G : \text{mangezin}(a,h) \wedge \text{kindgezin}(b,h) && (1.10)\\
\text{moederkind}(a,b) &\iff \exists h \in G : \text{vrouwgezin}(a,h) \wedge \text{kindgezin}(b,h) && (1.11)\\
\text{ouderkind}(a,b) &\iff \text{vaderkind}(a,b) \vee \text{moederkind}(a,b) && (1.12)
\end{aligned}$$

Met deze predikaten kunnen we de verzameling waarover in het voorbeeld van figuur 1.2 op de pagina hiernaast geïtereerd wordt korter definiëren:

$$(m \in P : m(\text{rol}) = \text{bruidegom}) \bowtie_{\text{manvrouw}(m,v)} (v \in P : v(\text{rol}) = \text{bruid})$$

Hoofdstuk 2

De struktuur van een match

Na het voorbehandelen van de brondata beschikken we over een grote verzameling losse familiestructuurtjes, losse genealogieën. Het doel van het matchingproces is om deze genealogieën onderling te verbinden. Daarom gaan we op zoek naar mensen die in meer dan één genealogie voorkomen. Preciezer gezegd, we zoeken naar knopen in verschillende genealogieën die lijken te refereren naar dezelfde reële persoon. Het liefst vinden we hele subgenealogieën die overeenkomen. Deze subgenealogieën en de overeenkomsten ertussen moeten echter wel aan specifieke eisen voldoen. Om die netjes uit te drukken zullen we genealogieën formeel definïeren.

2.1 Genealogieën als grafen

We kunnen een genealogie beschouwen als een speciale soort gerichte graaf met twee verschillende soorten knopen en twee verschillende soorten pijlen.

$$F = (P, G, H, K) \tag{2.1}$$

F**:** een genealogie (familiegraaf)

P**:** de verzameling van alle persoonsknopen

G**:** de verzameling van alle gezinsknopen

H**:** de verzameling van alle huwelijkspijlen: $H \subset G \times P$. Een huwelijkspijl van gezinsknoop g naar persoonsknoop p betekent dat p ouder of echtgenoot is van gezinsknoop g.

K**:** de verzameling van alle kindpijlen: $K \subset P \times G$. Een kindpijl van persoonknoop p naar gezinsknoop g betekent dat p een kind is uit gezinknoop g.

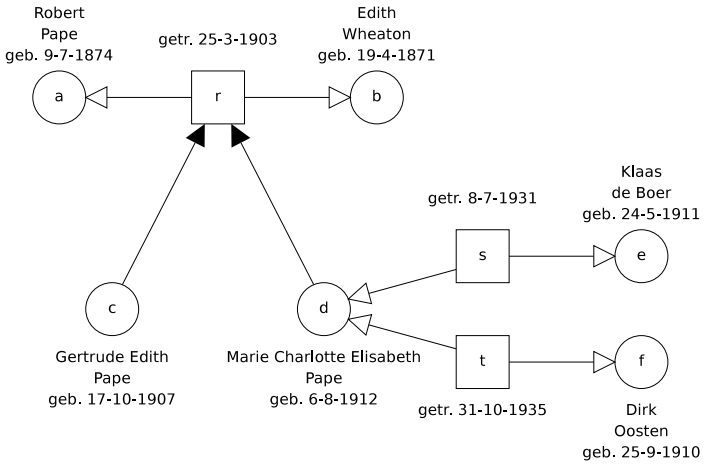
Geen persoonsknoop heeft meer dan één kindpijl:

$$\forall (a, x), (b, y) \in K : (a = b \rightarrow x = y) \tag{2.2}$$

Geen gezinsknoop heeft meer dan één huwelijkspijl naar een "vrouwelijke" persoonsknoop en naar een "mannelijke" persoonsknoop.

$$\forall (x, a), (y, b) \in H : \big((x = y \ \wedge \ a(\text{geslacht}) = b(\text{geslacht})) \rightarrow a = b\big) \tag{2.3}$$

Verder heeft een familiegraaf de eigenschap dat de pijlen vanaf de persoonsknopen naar hun (voor)ouders wijzen. Hierbij wijzen kindpijlen altijd van de persoonsknoop



Figuur 2.1: Een familiegraaf voor Marie Pape

naar het oudergezin en huwelijkspijlen van het gezin naar de echtelieden. De richting van de pijlen in acht genomen mogen er in de graaf geen cykels voorkomen. Dat zou immers betekenen dat personen hun eigen voorouder zijn.

Ook ontlenen familiegrafen nog veel "zachte" struktuureigenschappen aan de brondata waarop ze gebaseerd zijn. Zo zullen broers en zussen nooit en neven en nichten zelden met elkaar trouwen. Die eigenschappen zijn voor het definiëren van een match echter niet van belang.

Tenslotte hebben genealogieën meer dan alleen een struktuur. Derhalve beschouwen we de familiegrafen als gelabeld. Alle informatie die in tabellen P en G staat (niet te verwarren met de verzamelingen P en G) beschouwen we als de labels van de knopen in de graaf. We zullen naar een veld v in het label van knoop x refereren met $x(v)$, net als bij de tabelnotatie.

2.1.1 Voorbeeld: meerdere huwelijken

Als voorbeeld van een familiegraaf toont figuur 2.1 een graaf die enkele van de familierelaties van *Marie Pape* beschrijft. In deze graaf zijn de cirkels persoonsknopen, de vierkanten gezinsknopen, de zwarte pijlen kindpijlen en de witte pijlen huwelijkspijlen. Hier geldt:

- $P = \{a, b, c, d, e, f\}$
- $G = \{r, s, t\}$
- $K = \{(c,r),(d,r)\}$
- $H = \{(r,a),(r,b),(s,d),(s,e),(t,d),(t,f)\}$

Deze graaf voldoet aan de eisen van een familiegraaf. Zo is geen huwelijksknoop gekoppeld aan meer dan één echtegenoot en één echtegenote, maar andersom heeft Marie Pape wel meer dan één huwelijksknoop. Verder is geen persoonsknoop kind bij meer dan één gezin, maar zijn *Gertrude Pape* en Marie Pape wel kinderen bij hetzelfde gezin.

Merk op dat hier ter illustratie een familiegraaf wordt gebruikt die ingewikkelder is dan die van een huwelijksakte.

2.2 De definitie van een match

Een match is een relatie tussen twee familiegrafen. Om een match te maken leggen we twee familiegrafen voor een gedeelte over elkaar heen, zodat persoonsknoop bij persoonsknoop komt, gezinsknoop bij gezinsknoop en de pijlen qua richting en aard overeenkomen. Van de persoons- en gezinsknopen bekijken we vervolgens of ze naar dezelfde reële personen en gezinnen kunnen verwijzen. Daartoe vergelijken we de labels. Komen die in hoge mate overeen, dan hebben we een goede match, zo niet, dan hebben we een slechte match.

Twee familiegrafen hoeven dus slechts voor een gedeelte overeen te komen om een match te zijn. Bovendien is alleen de struktuur relevant. De labels bepalen of het een goede of een slechte match is. We definiëren een *simpele match* als de isomorfie tussen twee samenhangende deelgrafen van verschillende familiegrafen. Deze bestaat uit een bijectie tussen persoonsknopen en tussen gezinsknopen waarbij de pijlenstructuur tussen de op elkaar geprojecteerde knopen hetzelfde is in beide grafen. Hieronder noemen we de bijectierelatie tussen twee knopen een knoopmatch, zijnde een onderdeel van de match tussen twee familiegrafen.

Een simpele match is echter nog geen *echte match*. Een echte match voldoet aan een een structuureis waaraan een simpele match niet voldoet. Zo kunnen in een simpele match twee persoonsknopen naast elkaar worden gelegd, zonder dat de ouderknopen eveneens worden vergeleken. Omdat we aannemen dat een persoon maar twee ouders kan hebben, moeten, wanneer twee persoonsknopen naar dezelfde persoon verwijzen, de ouderknopen naar dezelfde ouders verwijzen. Met andere woorden, in een echte match zijn de aanwezige voorouderknopen van iedere gematchte knoop ook gematcht.

Op andere punten is de echte match even flexibel als de simpele match. Zo geldt niet dat wanneer twee gezinsknopen gematcht worden, de kinderknopen ook onderling gematcht moeten worden. Zo kunnen de ouders op twee geboorteakten gematcht worden, zonder dat het om dezelfde kinderen hoeft te gaan. Ook geldt voor een echte match de eis om voorouderknopen te matchen slechts wanneer die in beide grafen bekend zijn.

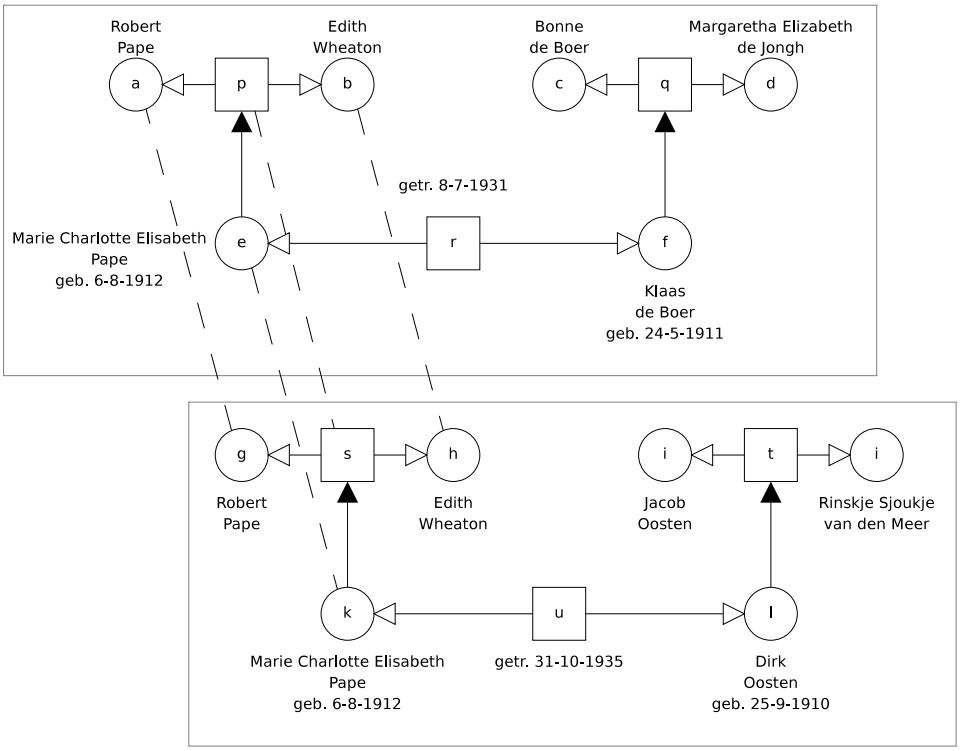
2.2.1 Voorbeeld: simpele en echte matches

Wanneer we twee huwelijksakten naast elkaar leggen, zijn er verschillende matches mogelijk. In figuur 2.2 op de pagina hierna zien we de familiegrafen van twee huwelijksakten. Merk op dat in deze grafen al een stuk minder bekend is over de ouders dan in de graaf van figuur 2.1. De onderbroken lijnen geven een paar mogelijke matches aan. Om enkele voorbeelden te geven:

- De match van e met k is een simpele match, maar geen echte.
- De match van a met g is een echte.
- De match van b met h, e met k en p met s is geen echte, maar als we a met g toevoegen wel.
- De qua labels onwaarschijnlijke match van e met h is ook een echte, maar wel een slechte.

2.2.2 Echte matches

Om een echte match te kunnen definiëren beschrijven we eerst een simpele match. Een simpele match M tussen familiegrafen $S = (P_S, G_S, H_S, K_S)$ en $T = (P_T, G_T, H_T, K_T)$ is een isomorfie tussen component S' van S en component T' van T'.



Figuur 2.2: Mogelijke matches tussen twee akten

Deze bestaat uit een bijectie $M : P_{S'} \rightarrow P_{T'}, G_{S'} \rightarrow G_{T'}$ tussen persoonsknopen en tussen gezinsknopen. Deze bijectie voldoet aan de eis dat de pijlenstructuur bewaard blijft:

$$\forall a \in P_{S'}, x \in G_{S'} : \Big((a,x) \in K_{S'} \leftrightarrow \big(M(a), M(x)\big) \in K_{T'}\Big) \tag{2.4}$$

$$\forall a \in P_{S'}, x \in G_{S'} : \Big((x,a) \in H_{S'} \leftrightarrow \big(M(x), M(a)\big) \in H_{T'}\Big) \tag{2.5}$$

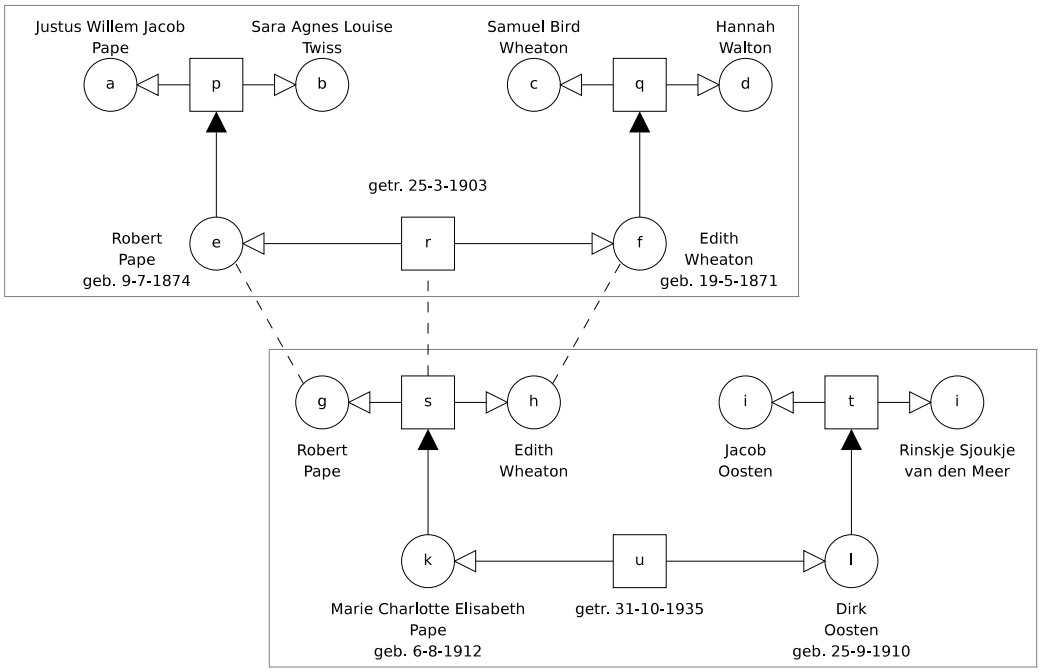
Een simpele match kan worden uitgebreid met de ouder-gezinsknopen van alle persoonsknopen en de echtgenoot- en echtgenoteknopen bij alle gezinsknopen, zolang de knopen maar in beide familiegrafen voorhanden zijn. Als we blijven uitbreiden totdat we alle mogelijke voorouderknopen ook hebben gematcht, hebben we een echte match. We definiëren hiervoor de uitgebreide match U_M van M. Deze match is een bijectie tussen componenten van familiegrafen S en T, die iets groter zijn dan S' en T'. We definiëren U_M als volgt:

$$\begin{array}{lcl} M(a) = b & \longrightarrow & U_M(a) = b \\ M(a) = b \ \wedge \ (a,x) \in K_S \ \wedge \ (b,y) \in K_T & \longrightarrow & U_M(x) = y \\ M(x) = y \ \wedge \ (x,a) \in H_S \ \wedge \ (y,b) \in H_T \ \wedge & & \\ \quad (a(\text{geslacht}) = b(\text{geslacht})) & \longrightarrow & U_M(a) = b \end{array} \tag{2.6}$$

Een simpele match M is nu een echte match als geldt:

$$U_M = M \tag{2.7}$$

Van iedere simpele match kan dus een echte match gemaakt worden door hem volgens formule 2.6 herhaaldelijk uit te breiden, totdat hij niet meer uitgebreid kan worden. Met andere woorden, we matchen persoonsknopen pas op de juiste manier wanneer we al hun voorouderknopen, zo die er zijn, in de vergelijking meenemen.



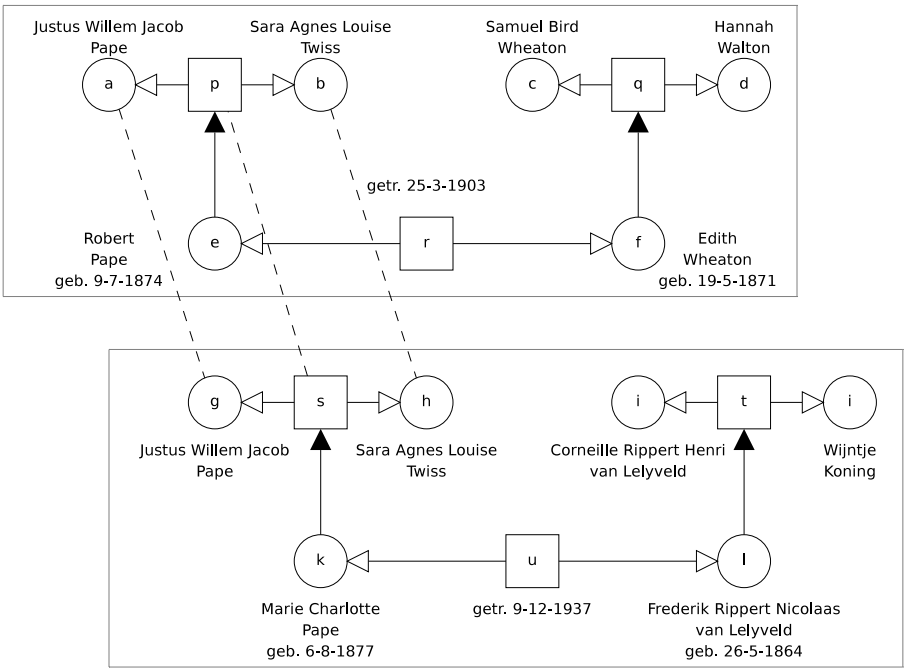
Figuur 2.3: De standaard-match tussen twee huwelijksakten, type A

2.3 Soorten matches

Het aantal manieren om een echte match te maken tussen twee familiegrafen kan heel groot zijn. In dit project hebben we echter voor het grootste deel met huwelijksakten gewerkt. Daarom zijn we met name geïnteresseerd in twee soorten matches, die we hieronder zullen behandelen. We beperken ons daar in de methodologie echter niet toe. In de toekomst zullen met de software die tijdens dit project is geschreven ook matches worden gemaakt tussen huwelijksakten en overlijdensakten van Genlias, waarop de overledene en zijn ouders staan, en tussen huwelijksakten en geboorteakten. De geboorteakten hebben dezelfde struktuur als overlijdensakten, maar zijn over het algemeen completer. Op overlijdensakten ontbreken vaak één of beide ouders. De geboorteakten die we willen gebruiken zijn afkomstig uit eigen onderzoek van de HSN en zijn ieder van één van de historische personen uit de steekproef.

In dit schrijven concentreren we ons op de twee soorten matches die we tussen huwelijksakten willen maken. De eerste soort match is die tussen de ouders van bruid of bruidegom op één akte met de huwenden van een andere akte. Met andere woorden, we gaan op zoek naar de eigen huwelijksakte van een ouderpaar. Zie figuur 2.3. Via deze soort matches kunnen we de trouwakten van kinderen koppelen aan die van hun ouders, die weer worden gekoppeld aan die van de grootouders. Zo kunnen we hele genealogieën reconstrueren. We zullen dit matchtype A noemen.

De tweede soort match is van twee ouderparen op verschillende akten, zoals in figuur 2.4 op de volgende pagina. Wanneer we alle matches van het eerste type hebben gevonden, hoeven we niet meer naar deze matches te zoeken, omdat de matches van twee ouderparen met een bruidspaar neerkomen op een match tussen twee ouderparen. Vaak komt echt voor dat de huwelijksakte van de ouders niet te vinden is. In dat geval zijn zulke matches van belang om broer-zusrelaties te kunnen bepalen, zoals in figuur 2.4 op de pagina hierna bij *Robert Pape* en *Marie Charlotte Pape*. Deze soort matches noemen we type B.



Figuur 2.4: De alternatieve match tussen twee huwelijksakten, type B

We zijn op zoek naar matches van twee of meer persoonsknopen. Matches van één persoonsknoop in beide grafen beschouwen we als onbetrouwbaar. Zelfs als de match perfect is, is er een relatief grote kans dat de persoonsknopen toch naar verschillende individuen verwijzen. Een match van twee persoonsknopen heeft daarop een veel kleinere kans, in de orde van grootte van het kwadraat ervan.

De grootte van een match is dus een belangrijke factor bij het bepalen van de betrouwbaarheid van de match. In dit project is hier echter nog geen rekening mee gehouden. De twee soorten matches die we hoofdzakelijk gebruiken bestaan namelijk altijd uit twee persoonsknopen en één gezinsknoop. Al die matches hebben dezelfde grootte.

Zoals is gebleken in het voorbeeld van paragraaf 2.2.1 op pagina 21 zijn echte matches niet per se goede matches. Om de kwaliteit van een echte match te bepalen moeten de labels van de gematchte knopen worden vergeleken. Hier zullen we in het volgende hoofdstuk op ingaan.

Hoofdstuk 3

De mate van identiteit

Het doel van het matchingproces is het vinden van goede matches. Zoals in het vorige hoofdstuk beschreven is, kunnen alle huwelijksakten op velerlei manieren met elkaar gematcht worden. Het grootste deel van dergelijke matches is echter zeer slecht. Voordat we kunnen gaan zoeken naar goede matches, moeten we bepalen wat een match goed maakt.

Een match is goed als de labels van de gematchte knopen overeenkomen. Bij een perfecte match zijn de labels van de knopen identiek. Als het verschil tussen de labels te groot is, beschouwen we de match als te slecht en gooien we hem weg.

Om het verschil tussen de labels te kwantificeren rekenen we een getal uit dat we d noemen, naar het Engelse "difference". Dat getal is een maat voor het totale verschil tussen de labels en aan de hand ervan bepalen we of een match geslaagd is of niet. We definiëren dat d gelijk is aan 0, wanneer er geen verschillen zijn en dat d gelijk is aan 1, wanneer de verschillen maximaal zijn. De waarde d is afhankelijk van de verschillen tussen de labels. Daarom definiëren we voor ieder paar gematchte knopen a en b de waarde $d(a,b) = d(b,a)$ als maat voor het verschil tussen hun labels. De totale d berekenen we als volgt:

$$d^{\mathrm{M}}(I) = \frac{1}{w_{\mathrm{p}} + w_{\mathrm{g}}} \left(w_{\mathrm{p}} \sqrt{\frac{1}{|P'|} \sum_{p \in P'} d^{\mathrm{p}}(p, I(p))^2} + w_{\mathrm{g}} \sqrt{\frac{1}{|G'|} \sum_{g \in G'} d^{\mathrm{g}}(g, I(g))^2} \right) \tag{3.1}$$

$d^{\mathbf{M}}(I)$**:** de d waarde van match I

P'**:** de verzameling persoonsknopen van de gematchte component van één van familiegrafen van de match I

G'**:** de verzameling gezinsknopen van de gematchte component van één van familiegrafen van de match I

w_{p}**:** het relatieve gewicht van het verschil tussen persoonsknopen

w_{g}**:** het relatieve gewicht van het verschil tussen gezinsknopen

$d^{\mathbf{p}}(a,b)$**:** de d waarde die het verschil in de labels van persoonsknopen a en b representeert

$d^{\mathbf{g}}(c,d)$**:** de d waarde die het verschil in de labels van persoonsknopen c en d representeert

$I(a)$**:** de knoop die in match I gematcht is met knoop a

In deze functie nemen we niet het gemiddelde van d over de persoons- of gezinsknopen, maar het kwadratisch gemiddelde. Dit zorgt ervoor dat uitschieters naar boven zwaarder meetellen. Wanneer een match grotendeels klopt, maar twee persoonsknopen slaan toch duidelijk niet op dezelfde persoon, moet dat zijn weerslag hebben op de kwaliteit van de gehele match, zonder dat dit al te gemakkelijk door een aantal goede matches tussen persoonsknopen gecompenseerd wordt.

We baseren d^{p} op allerlei mogelijke verschillen tussen persoonsknopen, zoals geslacht, voor- en achternaam en geboortedatum. De waarde voor d^{g} is alleen afhankelijk van de huwelijksdata van beide gezinsknopen. We hanteren voor beide een maximum van 0,5. Als één d-waarde daar boven komt is het verschil te groot en keuren we de match af. Voor persoonsknopen is deze bovengrens ruim. Over het algemeen zitten er tussen de 0,3 en 0,5 alleen maar slechte matches. Daarentegen is 0,5 wel zo laag dat we in pure aantallen slecht een heel kleine fractie overhouden van het totale aantal mogelijke matches, dat ook wel astronomisch groot is. Het aantal dat overblijft is in ieder geval klein genoeg om te behandelen en op te slaan. Voor gezinsknopen geldt dat de d-functie altijd 0 of 1 is. Komen de huwelijksdata overeen, dan is die 0, zoniet dan is die 1. We behouden alleen degene die 0 zijn. Dat betekent dat gezinsknopen in formule 3.1 op de pagina hiervoor voor spek en bonen meedoen. De verschillen zijn toch altijd 0, anders keuren we de match af. Daarom zetten we de wegingsfactoren w_g op 0 en w_p op 1.

3.1 Verschil tussen persoonsknopen

Om een d-waarde uit te rekenen voor het verschil tussen twee persoonsknopen kijken we alleen naar de namen en het geslacht. Net als bij gezinsknopen geldt dat veel informatie zoals geboortedatum, geboorteplaats en beroep niet aanwezig is voor één van de twee gematchte knopen. Beroep is vaak wel aanwezig in beide knopen, maar het interessante aan het beroep is juist dat het kan en mag verschillen. Zo komen we bij een goede match te weten welke beroepen iemand tijdens zijn leven heeft gehad. Daarom betrekken we dat niet bij het matchingsproces.

Met alleen de naam en het geslacht in gedachten rekenen we het verschil $d^{\mathrm{p}}(a,b)$ tussen knopen a en b als volgt uit:

$$d^{\mathrm{p}}(a,b) = \begin{cases} 1 & , \quad \text{als } d^{\mathrm{s}}(a,b) = 1 \\ d^{\mathrm{n}}(a,b) & , \quad \text{anders.} \end{cases} \tag{3.2}$$

met

$$d^{\mathrm{s}}(a,b) = \begin{cases} 0 & , \quad \text{als } a(\text{geslacht}) = b(\text{geslacht}) \\ 1 & , \quad \text{anders.} \end{cases} \tag{3.3}$$

en

$$d^{\mathrm{n}}(a,b) = \frac{1}{w_{\mathrm{vn}} + w_{\mathrm{vv}} + w_{\mathrm{an}}} \left(w_{\mathrm{vn}} d^{\mathrm{vn}}(a,b) + w_{\mathrm{vv}} d^{\mathrm{vv}}(a,b) + w_{\mathrm{an}} d^{\mathrm{an}}(a,b)\right) \tag{3.4}$$

d^{s}**:** het verschil in geslacht (sexe) tussen persoonsknopen a en b

$d^{\mathrm{n}}(a,b)$**:** het verschil tussen de namen van persoonsknopen a en b

$d^{\mathrm{vn}}(a,b)$**,** $d^{\mathrm{vv}}(a,b)$ **en** $d^{\mathrm{an}}(a,b)$**:** maten voor de verschillen tussen de voornamen (vn), de voorvoegsels (vv) en de achternamen (an) van knopen a en b

w_{vn}**,** w_{vv} **en** w_{an}**:** de gewichten waarmee het relatieve belang van de verschillen in voornamen, voorvoegsel en achternaam ingesteld kunnen worden

Het uitrekenen van de d-waarden voor de drie naamvelden verloopt op uiteenlopende wijze. Met name d^{vn} voor de voornamen wordt op complexe wijze tot stand gebracht. We beginnen daarom met de uiteenzetting van d^{vv} en d^{an}. De specificatie van de d^{vn}-waarde zal daarop voortborduren.

3.2 Verschil in voorvoegsels en achternamen

3.2.1 De Levenshteinafstand

Het verschil in voorvoegsels en achternamen wordt bepaald aan de hand van een *edit distance*, namenlijk de *Levenshtein*-afstand [L66, p. 707–710]. Deze afstand tussen twee strings is gelijk aan het minimum aantal wijzigingen dat je moet aanbrengen in één van de twee strings om hem in de andere te veranderen. Hierbij worden drie soorten wijzigingen onderscheiden: het toevoegen van een teken, het veranderen van één teken in een ander en het verwijderen van een teken. Het maakt niet uit met welke string je begint.

De maximum afstand tussen twee strings is gelijk aan de lengte van de langste string. De afstand wordt maximaal wanneer de strings geen enkel teken gemeen hebben. Om de kortste string in een evenlange substring van de langere te veranderen moeten alle tekens van de kortere string worden veranderd. Vervolgens moeten evenveel tekens worden toegevoegd als het verschil in de lengte.

Om de Levenshteinafstand te normaliseren tot een waarde die tussen 0 en 1 ligt, zullen we hem delen door de maximale afstand. De *relatieve* Levenshteinafstand wordt dan:

$$\mathcal{L}(s,t) = \frac{\text{Levenshtein}(s,t)}{\max(|s|,|t|)} \tag{3.5}$$

met

$$\max(a,b) = \begin{cases} a & , \quad a \geq b \\ b & , \quad \text{anders} \end{cases} \tag{3.6}$$

$\mathcal{L}(s,t)$**:** de relatieve Levenshteinafstand tussen strings s en t

Levenshtein(s,t)**:** de Levenshteinafstand tussen strings s en t

$|a|$**:** het aantal tekens in string a

Voor $\mathcal{L}$ geldt dat deze 0 is, wanneer twee strings identiek zijn, en 1 wanneer twee strings volledig verschillen. Dit maakt $\mathcal{L}$ geschikt om te gebruiken voor d^{vv} en d^{an}. Levenshtein heeft bij zijn afstand een algoritme beschreven dat voor twee strings s en t de Levenshteinafstand uitrekent met tijd- en ruimtecomplexiteit in de orde van $\mathcal{O}(|s| \cdot |t|)$. Dit betekent dat we de afstand kunnen gebruiken zonder ons zorgen te maken over tijd of geheugen, ook al zullen we in hoofdstuk 5 toch op de tijd bezuinigen. De verschillen tussen achternamen en voorvoegsel kunnen dan eenvoudig worden uitgerekend.

$$d^{\text{vv}}(a,b) = \mathcal{L}(a(\text{voorvoegsel}), b(\text{voorvoegsel})) \tag{3.7}$$

$$d^{\text{an}}(a,b) = \mathcal{L}(a(\text{achternaam}), b(\text{achternaam})) \tag{3.8}$$

Hierbij zijn x(voorvoegsel) en x(achternaam) het voorvoegsel en de achternaam van persoonsknoop x.

3.2.2 Voorbeeld: veranderde achternamen

Als voorbeeld nemen we de verwante achternamen “Schmeitz” en “Smets”. De Levenshteinafstand tussen deze twee namen is gelijk aan 4. Van “Schmeitz” naar “Smets” gerekend moeten de “c”, “h” en “i” verwijderd worden en de “z” veranderd in een “s”. Om de relatieve afstand te vinden delen we deze waarde door de lengte van de langste naam (“Schmeitz”), namelijk 8:

$$\mathcal{L}(\text{“Schmeitz”}, \text{“Smets”}) = \frac{\text{Levenshtein}(\text{“Schmeitz”}, \text{“Smets”})}{\max(|\text{“Schmeitz”}|, |\text{“Smets”}|)} = 0{,}5 \tag{3.9}$$

In de praktijk (zie hoofdstuk 5) blijkt 0,5 een erg slechte waarde. Achternamen met zo'n grote Levenshteinafstand lijken zeer zelden op elkaar. Ookal zijn deze namen van origine verwant, we zullen matches met zulke grote verschillen als te slecht beschouwen. De achternamen “Kok” en “Kock” zullen daarentegen een acceptabelere waarde opleveren. Hier hoeft alleen de “c” te worden verwijderd of toegevoegd:

$$\mathcal{L}(\text{“Kok”}, \text{“Kock”}) = \frac{\text{Levenshtein}(\text{“Kok”}, \text{“Kock”})}{\max(|\text{“Kok”}|, |\text{“Kock”}|)} = 0{,}25 \tag{3.10}$$

Deze waarde is al een stuk beter. Over het algemeen zorgt deze manier van berekenen ervoor dat relatief kleine verschillen getolereerd worden, maar grote niet.

3.2.3 Het maximale verschil tussen achternamen

We constateren dat wanneer achternamen een onderling verschil hebben van $d > 0{,}5$, ze ook werkelijk niet meer op elkaar lijken. Ten gunste van de heuristiek (zie hoofdstuk 5) nemen we $d^{\text{an}}_{\max} = 0{,}5$ als bovengrens voor de relatieve Levenshteinafstand tussen achternamen. Wanneer waarden boven deze grens uitkomen, beschouwen we de d als 1 en kappen we de vergelijking af. Dat heeft een aantal voordelen. Ten eerste kunnen we het algoritme dat de Levenshteinafstand uitrekent een maximumwaarde meegeven, namelijk de helft van de lengte van het langste woord. Wanneer de Levenshteinafstand groter wordt dan deze waarde, stopt het algoritme en geeft de waarde 1 terug. Dit scheelt tijd.

Ten tweede hoeven we achternamen waarbij de langste langer is dan twee keer de kortste überhaupt niet meer te vergelijken. Daar zal de d-waarde sowieso boven de 0,5 uitkomen. Ook die geven we de waarde 1.

Ten slotte maken we dankbaar gebruik van deze bovengrens bij de voorbewerkingen voor het algoritme dat naar akten zoekt die gematcht kunnen worden. Het aantal combinaties achternamen waarvoor geldt dat $d \leq 0{,}5$ is namelijk vele malen kleiner dan het totale aantal combinaties. Zie wederom hoofdstuk 5.

3.3 Voornamen

Het verschil tussen de voornamen van twee persoonsknopen is gebaseerd op dezelfde berekening als bij achternamen en voorvoegsel. Als beide persoonsknopen één voornaam hebben, kunnen we het verschil als volgt uitrekenen:

$$d^{\text{lvn}}(a,b) = \mathcal{L}(a(\text{voornamen}), b(\text{voornamen})) \tag{3.11}$$

Voor de meeste persoonsknopen geldt echter dat het voornamenveld meerdere voornamen bevat, gescheiden door spaties of (helaas) andere tekens. We zouden deze paarsgewijs kunnen vergelijken, de eerste met de eerste, de tweede met de tweede en zo voorts. Maar vaak hebben de vergeleken persoonsknopen niet evenveel voornamen. Ook, zo blijkt uit de brondata, komt voor dat voornamen omgewisseld worden. We zullen met deze problemen rekening moeten bij het berekenen van d^{vn}.

$\mathcal{L}$	Peter	Jan	Wilhem
Johannes	0,875	0,625	0,875
Wilhelm	0,857	1	0,143
Peter	0	1	0,833
Gerard	0,667	0,833	1

Tabel 3.1: De relatieve Levenshteinafstanden tussen twee sets voornamen

Het zal de alerte lezer niet zijn ontgaan dat we in formule 3.11 op de linker pagina een definitie voor d^{lvn} geven en niet voor d^{vn}. Lvn staat hier voor *losse voornaam*. In plaats van het vergelijken van de voornamenvelden in hun geheel, splitsen we deze velden voor beide knopen in losse voornamen. Die losse voornamen proberen we vervolgens met elkaar in overeenstemming te brengen en de mate waarin dit lukt bepaalt de waarde d^{vn}.

3.3.1 Het koppelen van losse voornamen

Om makkelijker over de voornamen te spreken, stellen we de volgende definities voor. Voor een voornamenveld $\alpha = a(\text{voornamen})$ geldt dat α_1 de eerste voornaam is en $|\alpha|$ het aantal voornamen. Dus $\alpha_{|\alpha|}$ is de laatste voornaam. Laten we twee voorbeelden nemen:

- $\alpha =$ “Peter Jan Wilhem”
- $|\alpha| = 3$
- $\alpha_1 =$ “Peter”
- $\alpha_2 =$ “Jan”
- $\alpha_3 =$ “Wilhem”

We vermoeden dat in de derde naam “Wilhem” een schrijf- of tikfout is geslopen en dat er een “l” ontbreekt, maar zeker weten doen we het niet. We laten de voornaam zo staan. De naam α willen we vergelijken met:

- $\beta =$ “Johannes Wilhelm Peter Gerard”
- $|\beta| = 4$
- $\beta_1 =$ “Johannes”
- $\beta_2 =$ “Wilhelm”
- $\beta_3 =$ “Peter”
- $\beta_4 =$ “Gerard”

We zoeken nu naar koppelingen tussen de voornamen van beide namen met een zo laag mogelijke d^{lvn}-waarde. We nemen van deze waarden het gewogen gemiddelde over de koppelingen tussen de voornamen en noemen dit d^{gvn}. Tabel 3.1 bevat de relatieve Levenshteinafstanden tussen de twee sets namen. Net als in formule 3.1 op pagina 25 gaan we bij d^{gvn} uit van de wortel van het gemiddelde kwadraat van

de d^{lvn}-waarde om uitschieters naar boven extra gewicht te geven. In dit geval krijgen we het beste resultaat als we "Peter" met "Peter", "Jan" met "Johannes" en "Wilhem" met "Wilhelm" koppelen. De naam "Gerard" blijft nu over.

Net als bij achternamen houden we echter een $d^{\text{lvn}}_{\text{max}}$ aan van 0,5. We beschouwen matches tussen voornamen met een hogere d-waarde dan dat niet als matches. Ook hier biedt dat grote tijdwinst en bij de heuristiek grote ruimtewinst. We koppelen "Jan" dus niet met "Johannes".[1] Van de zeven losse voornamen blijven er dus drie namen ongekoppeld, terwijl er vier gekoppeld worden.

We kunnen dit weergeven als een partiële functie k met de volgende afbeelding: $k(1) = 3, k(3) = 2$. Dit levert op:

$$\begin{aligned} d^{\text{gvn}}(\alpha,\beta,k) &= \sqrt{\tfrac{1}{2}\left(\mathcal{L}\left(\alpha_1,\beta_{k(1)}\right)^2 + \mathcal{L}\left(\alpha_3,\beta_{k(3)}\right)^2\right)} \\ &= \sqrt{\tfrac{1}{2}\left(0^2 + 0{,}143^2\right)} = 0{,}101 \end{aligned} \tag{3.12}$$

Dat is een goede d-waarde. We houden echter wel drie namen over die überhaupt niet gematcht konden worden. Daar zullen we in de volgende paragraaf naar kijken. We rekenen d^{gvn} in het algemeen als volgt uit:

$$d^{\text{gvn}}(\alpha,\beta,k) = \sqrt{\frac{1}{|k|}\sum_{(i,j)\in k} d^{\text{lvn}}\left(\alpha_i,\beta_j\right)} \tag{3.13}$$

k**:** de partiële functie die indices van voornamen in α afbeeldt op de indices van voornamen in β waarmee ze gekoppeld zijn, hier ook beschouwd als de verzameling van alle tupels (i,j) waarvoor geldt dat $k(i) = j$

Het onderling koppelen van de voornamen is overigens nog best een lastige zaak. Om alle mogelijke combinaties te vergelijken is vrij veel rekenwerk nodig. We hebben er in dit project voor gekozen om *gretig* te werk te gaan. We koppelen van beide sets voornamen de twee losse voornamen die samen de beste waarde hebben als eerste. In het onderhavige voorbeeld zijn dit "Peter" en "Peter". Vervolgens herhalen we dit voor de overige namen. Als tweede combinatie koppelen we "Wilhem" en "Wilhelm". Daarna zijn "Jan" en "Johannes" aan de beurt, maar omdat we inmiddels een d-waarde van boven de 0,5 hebben stoppen we daar. We zullen deze aanpak nader bespreken in het volgende hoofdstuk.

3.3.2 Ongematchte voornamen

Er blijven voornamen ongematcht wanneer twee sets voornamen niet evenveel namen bevatten en wanneer er voor niet alle voornamen goede matches gemaakt kunnen worden. De d-waarde die we hiervoor uitrekenen noemen we d^{ovn}, waarbij *ovn* staat voor *ongematchte voornamen*. We rekenen die als volgt uit:

$$d^{\text{ovn}}(\alpha,\beta,k) = \frac{|\alpha_o| + |\beta_o|}{|\alpha| + |\beta|} \tag{3.14}$$

met

$$\begin{aligned} |\alpha_o| &= |\{1 \le i \le |\alpha| : \neg\exists(i,j) \in k\}| \\ |\beta_o| &= |\{1 \le j \le |\beta| : \neg\exists(i,j) \in k\}| \end{aligned}$$

[1] Dit zullen we herzien in hoofdstuk 4.

$|\alpha|$ **en** $|\beta|$**:** het aantal voornamen in voornamenvelden α en β

$|\alpha_o|$ **en** $|\beta_o|$**:** het aantal voornamen in voornamenveld α en β dat niet gekoppeld is

Deze waarde is 1 wanneer geen naam gematcht kon worden en 0 als er geen ongematchte namen zijn. In het onderhavige voorbeeld zijn van de 7 voornamen in totaal er 4 gematcht en blijven er 3 over. Daar geldt dus $d^{\text{ovn}}(\alpha,\beta) = \frac{3}{7} = 0{,}429$.

3.3.3 Veranderde volgorden

Vaak komt het voor dat voornamen in verschillende volgorden worden opgegeven of om andere redenen in een andere volgorde genoteerd staan. Ook dergelijke verschillen willen we kwantificeren en we willen een d-waarde uitrekenen om te gebruiken in de uiteindelijke formule voor d^{vn}. Deze d noemen we d^{vvn}, voor *volgorde van voornamen*, en hij moet een aantal eigenschappen hebben. Enerzijds willen we dat $d^{\text{vvn}} = 0$ wanneer de volgorde niet veranderd is en anderzijds dat d^{vvn} maximaal 1 wordt bij grote veranderingen in volgorde.

Ten tweede willen we grote en kleine sprongen van één voornaam gelijk behandelen. Wanneer de naam $\alpha = \alpha_1\alpha_2\alpha_3\alpha_4$ verandert in $\alpha' = \alpha_3\alpha_1\alpha_2\alpha_4$, beschouwen we dit als één sprong van de naam α_3. Het verschil in volgorde tussen α en α' en tussen α en $\alpha'' = \alpha_1\alpha_3\alpha_2\alpha_4$ moet hetzelfde zijn, al springt in het tweede geval de naam α_3 minder ver. Het tellen van het aantal transposities (omwisselingen van namen) is niet geschikt als verschilmaat. Er zijn immers twee transposities nodig om van α naar α' te komen (eerst α_1 en α_3, vervolgens α_2 en α_1), en maar één om van α naar α'' te komen (α_2 en α_3). Tenslotte moet natuurlijk gelden dat $d^{\text{vvn}}(\alpha,\alpha',k) = d^{\text{vvn}}(\alpha',\alpha,k^{-1})$. We verlangen de tweede eigenschap omdat we in de data zien dat wanneer de volgorde in voornamen verandert, er vaak één naam naar voren wordt gehaald. Effectief verschuiven dan alle andere namen één stapje naar achteren. Dat zou een te groot volgordeverschil opleveren als we in transposities rekenen.

We rekenen het verschil in volgorde alleen uit voor de gematchte voornamen. De voornamen die niet gekoppeld zijn laten we buiten beschouwing. Eveneens kijken we niet naar de kwaliteit van de match. We reduceren een set voornamen tot een string symbolen, waarbij ieder symbool een gematchte voornaam representeert en symbolen voor de ongematchte voornamen weggelaten worden. De andere set voornamen wordt eveneens tot een string symbolen teruggebracht, waarbij de symbolen zo gekozen worden dat gematchte voornamen dezelfde symbolen hebben. We houden twee even lange strings over die bestaan uit dezelfde symbolen in mogelijk een andere volgorde.

We kunnen nu het gezochte volgordeverschil uitrekenen met behulp van de langste gemene deelrij van beide strings.

$$d^{\text{vvn}}(\alpha,\beta,k) = \begin{cases} 0 & , \quad \text{als } |s_\alpha| = |s_\beta| = 1 \\ \frac{|s_\alpha| - |\text{LGD}(s_\alpha, s_\beta)|}{|s_\alpha| - 1} & , \quad \text{anders.} \end{cases} \tag{3.15}$$

$d^{\text{vvn}}(\alpha,\beta,k)$**:** het volgordeverschil tussen voornamen α en β

$\text{LGD}(s,t)$**:** de langste gemene deelstring van strings s en t

$|s|$**:** de lengte van string s (merk op dat hier altijd geldt dat $|s_\alpha| = |s_\beta|$)

s_α**:** een string symbolen die de volgens k gematchte namen van voornamenset α representeert

Een efficiënt algoritme om de langste gemene deelrij uit te rekenen is gelukkig voorhanden door middel van dynamisch programmeren [CL01, p. 350–355]. Dit algoritme rekent de lengte van de langste gemene deelrij(en) uit met een tijd- en ruimtecomplexiteit in de orde van $\mathcal{O}(|s| \cdot |t|)$, waarbij s en t de bekeken strings zijn.

Het is snel in te zien dat het gebruik van de langste gemene deelrij (LGD) de gewenste afstand oplevert. Wanneer we tellen hoeveel sprongen we nodig hebben om van de eerste string naar de tweede string te komen, kijken we niet naar de spronggrootte. Daarom hoeven we een symbool maximaal één keer te laten springen, namelijk naar de plaats die hij in de de tweede string heeft. Het volstaat dus om te kijken welke symbolen dienen te verspringen en welke we kunnen laten staan. Het maximum aantal symbolen dat we kunnen laten staan is gelijk aan de lengte van de LGD. Die bestaat tenslotte precies uit de langste rij symbolen die ten opzichte van elkaar in beide strings in dezelfde volgorde staan.

De lengte van de LGD is altijd minimaal 1, want wanneer de volgorde van de strings volledig omgekeerd is, is ieder symbool een LGD. Daarom kunnen we in formule 3.15 op de pagina hiervoor de sprongafstand normaliseren tot een waarde van 0 tot en met 1 door hem te delen door de lengte van de strings minus 1.

In het onderhavige voorbeeld maken we van de naam $\alpha =$ "Peter Jan Wilhem" de string *abc*, waarna we het symbool voor de middelste naam weglaten omdat deze niet gekoppeld is. We houden $s_\alpha = ac$ over. Voor de naam $\beta =$ "Johannes Wilhelm Peter Gerard" make we de string *dcae*. We gebruiken hier voor de middelste twee namen symbolen als voor de namen van α waarmee deze gekoppeld zijn, ongeacht hoe goed deze koppelingen zijn. We laten de symbolen voor de ongekoppelde namen weer weg en houden over $s_\beta = ca$. Bij deze twee strings is de volgorde volledig omgekeerd. We verwachten dus een volgordeverschil van 1. Zowel a als c zijn LGD's van s_α en s_β en hebben lengte 1. Dit betekent dat $d^{\text{vvn}}(\alpha,\beta,k) = \frac{2-1}{2-1} = 1$.

3.3.4 Het totale verschil tussen voornamen

We hebben voor het verschil in voornamen nu drie waarden uitgerekend, één voor de mate waarin losse voornamen overeenkomen, één voor hoeveel losse namen overeenkomen en één voor de mate waarin overeenkomstige losse voornamen van volgorde veranderd zijn. We zullen deze drie waarden als volgt samennemen:

$$d^{\text{vn}}(a,b) = d^{\text{ovn}}(\alpha,\beta,k) + (1 - d^{\text{ovn}}(\alpha,\beta,k)) \left(1 - \frac{1 - d^{\text{gvn}}(\alpha,\beta,k)}{1 + w_{\text{vvn}} d^{\text{vvn}}(\alpha,\beta,k)}\right) \tag{3.16}$$

met

$$\alpha = a(\text{voornamen}) \quad \text{en} \quad \beta = b(\text{voornamen})$$

w_{vvn}**:** een gewicht om het belang van volgorde mee in te stellen

k**:** een partiële functie die de match beschrijft tussen de losse voornamen van α en β

De belangrijkste rol is toebedeeld aan d^{ovn}. Bij een gelijke volgorde ($d^{\text{vvn}} = 0$) is deze formule gelijk aan:

$$d^{\text{vn}}(a,b) = d^{\text{ovn}}(\alpha,\beta,k) + (1 - d^{\text{ovn}}(\alpha,\beta,k))\, d^{\text{gvn}}(\alpha,\beta,k) \tag{3.17}$$

Dit levert de waarde op die we zouden krijgen wanneer we alle ongematchte voornamen beschouwen als voornamen gematcht met een relatieve Levenshteinafstand van 1.

Wanneer de gematchte voornamen van volgorde veranderen wordt de matchkwaliteit gedeeld door een factor die afhangt van w_{vvn} en d^{vvn}. Met de matchkwaliteit bedoelen we het tegenovergestelde van het verschil, in dit geval $1 - d^{\text{lvn}}$. We hebben

in dit project voor w_{vvn} de waarde 1 genomen. Dit betekent dat de matchkwaliteit van de gematchte voornamen halveert wanneer de volgorde volledig verkeerd is.

Voor het voorbeeld van dit hoofdstuk met α = "Peter Jan Wilhem" en β = "Johannes Wilhelm Peter Gerard" hebben we de volgende waarden uitgerekend:

- $d^{\text{gvn}}(\alpha,\beta,k) = 0{,}101$
- $d^{\text{ovn}}(\alpha,\beta,k) = 0{,}429$
- $d^{\text{vvn}}(\alpha,\beta,k) = 1$

Het total verschil tussen α en β wordt:

$$d^{\text{vn}}(\alpha,\beta) = 0{,}429 + (1 - 0{,}429)\left(1 - \frac{1 - 0{,}101}{1 + 1 \times 1}\right) = 0{,}743 \tag{3.18}$$

Omdat we hier een voorbeeld gebruiken om alle mogelijke verschillen te illustreren is deze waarde niet erg goed. Over het algemeen zijn we geïnteresseerd in matches waarbij de voornamen een stuk meer op elkaar lijken dan deze twee.

We hanteren dan ook maxima voor d^{vn} en d^{an} van 0,5. Wanneer óf de achternamen óf de voornamen met meer dan deze waarde verschillen, beschouwen we een match als te slecht en stoppen met het vergelijken van de knopen. Met d^{vv} gaan we flexibeler om. Daar staan we grote verschillen toe.

In het voorbeeld van dit hoofdstuk nemen we twee namen die uiteindelijk niet erg overeenkomen. Eén van de oorzaken van de hoge d-waarde is de relatieve Levenshteinafstand tussen "Jan" en "Johannes". Omdat hij boven de 0,5 zit ronden we hem effectief af tot 1. Maar zijn "Jan" en "Johannes" wel zulke verschillende namen als de relatieve Levenshteinafstand suggereert? In het volgende hoofdstuk gaan we hier nader op in.

3.4 Het vergelijken van geboortedata

Het vergelijken van geboortedata is lastig, omdat we meestal van alleen de huwenden op een akte de geboortedatum of leeftijd weten. Bij de ouders staat dit niet op de akte, terwijl we ouders met huwenden of ouders onderling willen vergelijken. Het betekent echter niet dat we niets weten over de leeftijd van de ouders. Aan de hand van de geboortedata van hun kinderen kunnen we de mogelijke geboortedata van de ouders en tevens hun huwelijksdatum inschatten. We houden ons daarbij alleen bezig met geboorte- en huwelijksjaren, en niet met de exacte datum, omdat we meestal niet de geboortedatum kennen, maar de leeftijd in jaren.

We schatten voor alle persoonsknopen een periode waarin de persoon geboren moet zijn. We definiëren daarvoor voor iedere persoonsknoop een bereik $\text{gb} = [\text{gb}^{\text{min}}; \text{gb}^{\text{max}}]$ in hele jaren, waarbij gb^{min} het vroegst mogelijke jaar is en gb^{max} het laatst mogelijk jaar waarin de persoon geboren kan zijn. Hetzelfde doen we voor het huwelijksjaar van de gezinsknopen, namelijk het bereik $\text{hb} = [\text{hb}^{\text{min}}; \text{hb}^{\text{max}}]$ waarin het huwelijk voltrokken kan zijn. We voegen aan de tabel P de velden *gb-min* en *gb-max* toe en aan G *hb-min* en *hb-max* om dit bereik bij te houden. Voor records $p \in P$ en $h \in G$ zullen we met $p(\text{gb})$ en $h(\text{gb})$ naar deze bereiken verwijzen, als ware ze een soort samengesteld veld.

Voor de meeste huwenden is óf de leeftijd, óf de geboortedatum bekend. We kunnen gb dan ook op drie manieren invullen. In het makkelijkste geval is de geboortedatum gegeven; we vullen dan het geboortejaar in voor zowel gb^{min} als gb^{max}. In het tweede en meest voorkomende geval is alleen de leeftijd in jaren

constante	waarde	omschrijving
κ_h	$[15;115]$	de leeftijd waarop een persoon kan trouwen
κ_m	$[15;53]$	de leeftijd waarop een vrouw een moeder kan worden
κ_v	$[15;100]$	de leeftijd waarop een man vader kan worden

Tabel 3.2: De constanten voor het inperken van data

gegeven. Met behulp van de aktedatum kunnen we dan uitrekenen wanneer de persoon ongeveer geboren is. We moeten er echter rekening mee houden dat hij of zij ten tijde van het huwelijk zojuist de opgegeven leeftijd bereikt kan hebben, maar ook dat hij of zij bijna één jaar ouder kan zijn. We berekenen de nieuwe velden daarom als volgt:

$$\begin{array}{rcl} \text{gb}^{\min} & = & \text{jaar(aktedatum)} - \text{leeftijd} - 1 \\ \text{gb}^{\max} & = & \text{jaar(aktedatum)} - \text{leeftijd} \end{array} \tag{3.19}$$

Hierbij is jaar(aktedatum) het jaar waarin de aktedatum valt. In het derde geval is noch de geboortedatum noch de leeftijd vermeld. We zullen aan de hand van de aktedatum een periode bepalen waarbinnen de de persoon geboren moet zijn. We nemen aan dat de persoon niet jonger kan zijn dat 15 jaar en niet ouder dan 115. Dat bereik noemen we we $\kappa_h = [\kappa_h^{\min}; \kappa_h^{\max}] = [15;115]$. Hiermee kunnen we gb als volgt uitrekenen:

$$\begin{array}{rcl} \text{gb}^{\min} & = & \text{jaar(aktedatum)} - \kappa_h^{\max} \\ \text{gb}^{\max} & = & \text{jaar(aktedatum)} - \kappa_h^{\min} \end{array} \tag{3.20}$$

Met dit soort inperkingen kunnen we verschillende jaartallen inschatten. We gebruiken meer aannames dan alleen de minimale en maximale huwelijksleeftijd. We maken gebruik van de constanten in figuur 3.2. Hiermee schatten we niet alleen de geboortedata van de bruid en bruidegom in, maar ook die van hun ouders en de jaren waarin zij getrouwd kunnen zijn. Vooral κ_m geeft ons een goede inperking van jaren waarin we de ouders moeten zoeken. Vaak zal een bepaald bereik berekend kunnen worden op basis van meerdere constanten. In dat geval nemen we als bereik de overlapperiode. Soms zal er daarentegen geen overlap zijn. Dat duidt op inconsistente gegevens. De aktedatum heeft in dat geval prioriteit over de geboortedata of leeftijd van de huwenden. We berekenen de overlap als volgt:

$$r\,([a,b][c,d]) = \begin{cases} [\text{smax}(a,c),\text{smin}(b,d)] & , \quad \text{als } (a \le d \vee \text{isnull}(a) \vee \text{isnull}(d)) \wedge \\ & \quad\ (b \ge c \vee \text{isnull}(b) \vee \text{isnull}(c)) \\ [a,b] & , \quad \text{anders.} \end{cases} \tag{3.21}$$

met

$$\text{smax}(a,b) = \begin{cases} \max(a,b) & , \quad \text{als } \neg\text{isnull}(a) \wedge \neg\text{isnull}(b) \\ a & , \quad \text{als isnull}(b) \\ b & , \quad \text{anders.} \end{cases} \tag{3.22}$$

$$\text{smin}(a,b) = \begin{cases} \min(a,b) & , \quad \text{als } \neg\text{isnull}(a) \wedge \neg\text{isnull}(b) \\ a & , \quad \text{als isnull}(b) \\ b & , \quad \text{anders.} \end{cases} \tag{3.23}$$

Merk op dat de functie niet symmetrisch is. Wanneer er geen overlap is tussen $[a,b]$ en $[c,d]$ heeft het eerstgenoemde bereik prioriteit. Met de volgende functies kunnen we op basis van een leeftijdbereik a dat we reeds ingeschat hebben een tweede leeftijdsbereik b bepalen dat van a afhangt door middel van een κ-constante:

$$s(a,\kappa) = \left[a^{\min} - \kappa^{\max};\quad a^{\max} - \kappa^{\min}\right] \tag{3.24}$$

$$t(a,\kappa) = \left[a^{\min} + \kappa^{\min};\quad a^{\max} + \kappa^{\max}\right] \tag{3.25}$$

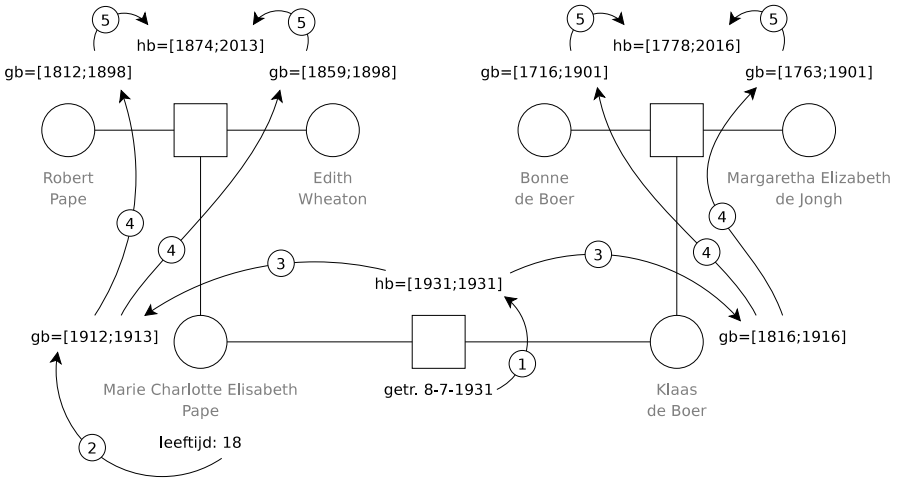
```
for each (h ∈ G : h(rol) = bruid&bruidegom) do
  h(hb) ← [jaar(h(aktedatum)); jaar(h(aktedatum))]                    (stap 1)
od
for each (p ∈ P : p(rol) ∈ {bruidegom; bruid}) do
  if ¬isnull(p(geboortedatum)) then                                   (stap 2)
    p(gb) ← [jaar(p(geboortedatum)); jaar(p(geboortedatum))]
  elseif ¬isnull(p(leeftijd)) then
    p(gb) ← [jaar(p(aktedatum)) − p(leeftijd) − 1;
             jaar(p(aktedatum)) − p(leeftijd)]
  else
    p(gb) ← [null; null]
  fi
od
for each (p ∈ P : p(rol) ∈ {bruidegom; bruid}) ⋈_partnergezin(p,h) (h ∈ G) do
  p(gb) ← r(s(h(hb),κ_h),p(gb))                                       (stap 3)
od
for each (p ∈ P) ⋈_kindgezin(p,q) (h ∈ G) =×_mangezin(v,h)(v ∈ P)
    =×_vrouwgezin(m,h)(m ∈ P) do
  h(hb) ← [null; null]
  if v ≠ ∅ then
    v(gb) ← s(p(gb),κ_v)                                              (stap 4)
    h(hb) ← t(v(gb),κ_h)                                              (stap 5)
  fi
  if m ≠ ∅ then
    m(gb) ← s(p(gb),κ_m)                                              (stap 4)
    h(hb) ← r(h(hb),t(m(gb),κ_h))                                     (stap 5)
  fi
od
```

Figuur 3.1: Een algoritme om jaartallen in te schatten

Het algoritme in figuur 3.1 rekent met behulp van deze functies alle jaarbereiken uit. Het begint met het invullen van het huwelijksjaarbereik van de gezinsknoop van de twee huwenden. Dit jaar is altijd gegeven, dus is dat makkelijk. Vervolgens wordt voor alle bruidegommen en bruiden het geboortejaarbereik bepaald aan de hand van leeftijd of geboortedatum. Dat bereik wordt in overeenstemming gebracht met het huwelijksjaarbereik, waarbij de laatste prioriteit heeft. Aan de hand van deze geboortejaarbereiken worden de mogelijke geboortejaren van de ouders bepaald en op basis daarvan hun mogelijke huwelijksjaren.

In figuur 3.2 op de volgende pagina staat een ons welbekende huwelijksakte. Voor deze akte worden door het schatalgoritme de jaarbereiken bepaald. De pijlen geven aan op welke gegevens de bereiken worden gebaseerd en de nummers geven aan in welke volgorde. Deze komen overeen met de stappen die in het algoritme worden genoemd. Als eerste wordt het gemakkelijk te bepalen huwelijksjaarbereik van het huwelijk van bruid en bruidegom bepaald. Dit beperkt zich tot 1931. Voor Marie Pape weten we hier de leeftijd, 18 jaar. Zij is dus geboren in 1912 of 1913. In stap 3 wordt dit bereik getoetst aan de hand van het huwelijksjaarbereik. Wat dat betreft moet ze tussen 1816 en 1916 geboren zijn. Het reeds bepaalde bereik voldoet hieraan, dus nemen we de overlap, waardoor we nog steeds het bereik [1912; 1913] hebben.

Voor Klaas de Boer — we gaan er hier van uit dat we zijn geboortedatum noch leeftijd kennen — is het lastiger. Bij stap 2 blijft zijn bereik op [null; null] staan



Figuur 3.2: De volgorde van het bepalen van jaarbereiken

en wordt in stap 3 ingeschat op $[1816; 1916]$ aan de hand van de huwelijksdatum. We kunnen zijn geboortejaar dus niet preciezer inschatten dan op 100 jaar. Voor zowel Marie Pape als Klaas de Boer gebruiken we hun geboortejaarbereik om die van de ouders in te schatten en hun mogelijke huwelijksjaren. Bij Klaas' ouders levert dit enorme periodes op: 185 jaar waarin zijn vader geboren kan zijn. De moeder van Marie Pape weten we met de meeste precisie: 39 jaar.

Gelukkig is het geval van Klaas de Boer uitzonderlijk en kunnen we het geboortejaar meestal op 1 of 2 jaar precies inschatten. De mogelijke geboortejaren van de moeder worden dan beperkt tot 38 à 39 jaar. Dat is het belangrijkste doel van het inschatten van de data. We hoeven nu, wanneer we bij een moederknoop een bruidsknoop proberen te matchen, maar in een periode van 40 jaar te kijken in plaats van meer dan 100 jaar. De vaderknoop en bruidegomsknoop hebben we er automatisch bij en hun geboortejaarbereik gebruiken we alleen ter controle.

We kunnen formule 3.2 op pagina 26 voor het verschil tussen personen uitbreiden tot:

$$d^{\mathrm{p}}(a,b) = \begin{cases} 1 & , \quad \text{als } d^{\mathrm{s}}(a,b) = 1 \vee d^{\mathrm{gb}}(a,b) = 1 \\ d^{\mathrm{n}}(a,b) & , \quad \text{anders.} \end{cases} \tag{3.26}$$

met

$$d^{\mathrm{gb}}(a,b) = \begin{cases} 0 & , \quad \text{als } a(\mathrm{gb}^{\min}) \leq b(\mathrm{gb}^{\max}) \wedge a(\mathrm{gb}^{\max}) \geq b(\mathrm{gb}^{\min}) \\ 1 & , \quad \text{anders.} \end{cases} \tag{3.27}$$

We kunnen nu ook de simpele functie geven voor d^{g}:

$$d^{\mathrm{g}}(c,d) = d^{\mathrm{hb}}(c,d) = \begin{cases} 0 & , \quad \text{als } c(\mathrm{hb}^{\min}) \leq d(\mathrm{hb}^{\max}) \wedge c(\mathrm{hb}^{\max}) \geq d(\mathrm{hb}^{\min}) \\ 1 & , \quad \text{anders.} \end{cases} \tag{3.28}$$

3.5 Een evaluatiealgoritme

In dit hoofdstuk hebben we formeel beschreven hoe aan een match een kwaliteit toegevoegd kan worden, door voor iedere relatie tussen persoonsknopen in de partiële bijectie (persoonsknoopmatch) en voor iedere relatie tussen gezinsknopen (gezinsknoopmatch) een d-waarde uit te rekenen. Gegeven een match kunnen we zonder problemen de totale d-waarde uitrekenen door over persoonsknoopmatches en gezinsknoopmatches in de match te itereren. In de praktijk bouwen we een match

```
matchGezinsknopen(g, h, Z) do
  d ← d^g(g,h)
  if d ≤ d^g_max then
    s ← waar
    C1 ← ∅
    if ∃p ∈ P : mangezin(p,g) ∧ ∃q ∈ P : mangezin(q,h) ∧ ¬∃(p,q,e) ∈ Z
    then
      C1 ← matchPersoonsknopen(p, q, Z ∪ {(g, h, d)})
      if C1 = ∅ then s ← onwaar
    fi
    C2 ← ∅
    if ∃p ∈ P : vrouwgezin(p,g)∧∃q ∈ P : vrouwgezin(q,h)∧¬∃(p, q, e)) ∈ Z∪C1
    then
      C2 ← matchPersoonsknopen(p, q, Z ∪ C1 ∪ {(g, h, d)})
      if C2 = ∅ then s ← onwaar
    fi
    C3 ← ∅
    K1 ← {k ∈ P : kindgezin(k, g)}
    if K1 ≠ ∅ then
      K2 ← {k ∈ P : kindgezin(k, h)}
      if K2 ≠ ∅ then
        C3 ← matchVeelKnopen(K1, K2, Z ∪ C1 ∪ C2 ∪ {(g, h, d)})
      fi
    fi
    if s then return C1 ∪ C2 ∪ C3 ∪ {(g,h,d)}
    else return ∅
  else
    return ∅
  fi
od
```

Figuur 3.3: De functie `matchGezinsknopen`

incrementeel op terwijl we hem evalueren. Zeker bij een grote match kan dat nodeloos rekenwerk schelen. Voor matchtype A en B is een dergelijke voorzichtigheid niet nodig. Ze bestaan altijd uit 1 familieknoopmatch en 2 persoonsknoopmatches. Maar met het oog op veelzijdigere toepassing presenteren we hier een algoritme dat voor echte matches van uiteenlopende vorm evaluatie uitvoert[2]. Het algoritme bestaat uit een aantal functies die elkaar recursief aanroepen. We starten het algoritme uit door de functie `matchGezinsknopen` voor twee te matchen gezinsknopen of `matchPersoonsknopen` voor twee persoonsknopen aan te roepen. Het algoritme zal langs kind- en huwelijkstakken lopen om een zo groot mogelijke match terug te geven. Als er geen goede en *echte* match te maken is, dan wordt een lege match teruggegeven.

Figuur 3.3 bevat de pseudocode voor de functie `matchGezinsknopen`. Deze functie accepteert als parameters de gezinsknopen g en h en een verzameling gematchte paren knopen Z. Die verzameling bevat knopen die al door het algoritme gematcht zijn en niet opnieuw overwogen hoeven worden. Deze houden we bij om te voorkomen dat het algoritme in cirkels loopt. Als eerste wordt in de functie het verschil tussen g en h opgeslagen in de variable d. Als deze niet te hoog is gaat de evaluatie verder. In de Boolese variable s wordt bijgehouden of de match succesvol kan zijn.

[2]Tijdens het schrijven wordt op bij de HSN op het IISG reeds gebruik gemaakt van de veelzijdigheid van het algoritme om matches te maken tussen huwelijksakten en geboorteakten.

```
matchPersoonsKnopen(p, q, Z) do
  d ← d^P(p,q)
  if d ≤ d^P_max then
    s ← waar
    C1 ← ∅
    if ∃g ∈ G : kindgezin(p,g) ∧ ∃h ∈ G : kindgezin(q,h) ∧ ¬∃(g,h,e) ∈ Z
    then
      C1 ← matchGezinssknopen(g, h, Z ∪ {(p,q,d)})
      if C1 = ∅ then s ← onwaar
    fi
    C2 ← ∅
    H1 ← {h ∈ G : partnergezin(p, h)}
    if H1 ≠ ∅ then
      H2 ← {h ∈ G : partnergezin(q, h)}
      if H2 ≠ ∅ then
        C2 ← matchVeelKnopen(H1, H2, Z ∪ C1 ∪ {(p,q,d)})
      fi
    fi
    if s then return C1 ∪ C2 ∪ {(p,q,d)}
    else return ∅
  else
    return ∅
  fi
od
```

Figuur 3.4: De functie `matchPersoonsknopen`

Dat is afhankelijk van het slagen van het matchen van de echtelieden. Als de echtgenoten matchen wordt een verzameling $C1$ ontvangen. Hierin staan niet alleen de echtgenootknopen zelf als tupel, maar mogelijk ook ieder paar voorouders of andere familieleden die zijn gematcht. Wanneer de echtgenoten niet voldoende matchen, wordt een lege verzameling teruggegeven. Dit betekent dat de match afgeblazen wordt. Het matchen van de echtgenotes gaat op dezelfde wijze en levert een verzameling $C2$ op. Bij beide aanroepen van de `matchPersoonsknopen`-functie wordt een derde parameter meegegeven, een verzameling van knopen die als reeds gematcht kunnen worden beschouwd. Tenslotte worden ook de kinderen van beide huwelijksknopen gematcht. Hiervoor wordt de functie `matchVeelKnopen` aangeroepen. Als dit geen matches oplevert, is dat niet erg.

Figuur 3.4 toont de pseudocode voor `matchPersoonsknopen`. Deze functie werkt analoog aan de eerste maar matcht persoonsknopen in plaats van gezinsknopen. Ook hier wordt eerst de d-waarde uitgerekend voor de twee knopen. Als deze voldoet, worden de oudergezinsknopen vergeleken. Als die niet matchen, faalt de match tussen de persoonknopen. Zo wel, dan gaan we verder met het vergelijken van de huwelijken van de ene persoonsknoop, opgeslagen in $H1$, met de huwelijken van de andere in $H2$. Ook hiervoor gebruiken we de functie `matchVeelKnopen`. Net als bij de kinderen in de vorige functie geldt hier dat het niet erg is als er geen matches tussen de huwelijken van beide personen wordt gevonden.

De functie `matchVeelKnopen`, figuur 3.5 op de rechter pagina, wordt aangeroepen voor twee verzamelingen A en B die beide óf persoonsknopen óf gezinsknopen bevatten. De derde parameter, verzameling Z, bevat ook hier reeds gemaakte matches. De functie geeft een aantal matches tussen knopen uit A en B terug in de verzameling C. Deze matches worden zo opgebouwd dat knopen nooit dubbel worden gematcht en dat geen knoop uit A meer één keer met een knoop uit B wordt

```
matchVeelKnopen(A, B, Z) do
  C ← ∅                                          definitieve matches
  S ← ∅                                          voorlopige matches
  U ← ∅                                          knopen uit A die reeds gematcht zijn
  V ← ∅                                          knopen uit B die reeds gematcht zijn
  g ← waar                                       wordt onwaar als we klaar zijn
  while g do
    for each a,b ∈ (A − U) × (B − V) do
      if a, b ∈ P then K ← matchPersoonsKnopen(a,b,Z ∪ C)
      else K ← matchGezinsKnopen(a,b,Z ∪ C)
      d ← d^M(K)
      s ← waar
      for each (m, n, e, L) ∈ S do
        if a = m ∨ b = n then
          if d < e then
            S = S − {(m, n, L, e, X)}
          else
            s ← onwaar
            leave for
          fi
        fi
      od
      if s then S = S ∪ {(a, b, d, K)}
    od
    g ← S ≠ ∅
    while S ≠ ∅ do
      (a,b,d,K) ← element van S met kleinste d
      if K ∩ C = ∅ then
        U ← U ∪ {a}
        V ← V ∪ {b}
        C ← C ∪ K
        S ← S − (a,b,d,K)
      else
        S ← ∅
        leave while
      fi
    od
  od
  return C
od
```

Figuur 3.5: De functie `matchVeelKnopen`

```
match(a, b) do
  if a, b ∈ P then
    C ← matchPersoonsKnopen(a, b, ∅)
  else
    C ← matchGezinsKnopen(a, b, ∅)
  fi
  if C ≠ ∅ then
    i = m(m-id) + 1 : m ∈ M ∧ ¬(∃n ∈ M : n(m-id) > m(m-id))
    for each {(p,q,d) ∈ C : p,q ∈ P} do
      PM ← PM ∪ {{(m-id,i), (p-id-paar,{p(p-id),q(p-id)}), (d,d), }}
    od
    for each {(g,h,d) ∈ C : g,h ∈ G} do
      GM ← GM ∪ {{(m-id,i), (g-id-paar,{g(g-id),h(g-id)}), (d,d), }}
    od
    d = d^M(C)
    p = |{(p,q,d) ∈ C : p,q ∈ P}|
    g = |{(g,h,d) ∈ C : g,h ∈ G}|
    M ← M ∪ {{(m-id, i), (d,d), (p,p), (g,g)}}
  fi
od
```

Figuur 3.6: De functie `match`

gematcht of andersom.

Allereerst worden alle mogelijke matches geprobeerd en opgeslagen in S. Wanneer een knoop meer dan één keer wordt gematcht, wordt alleen de beste match, met de laagste d-waarde, opgeslagen. Vervolgens wordt de beste match uit S opgenomen in C. De matches die daarna het best zijn komen alleen in C zolang er geen overlap bestaat tussen de matches. Wanneer er wel overlap onstaat, worden de desbetreffende match en alle overige weggegooid en wordt het proces herhaald voor de nog niet gematchte knopen. Zo wordt C gevuld met de gretig gevonden beste matches die onderling consistent zijn.

Het algoritme is niet erg efficiënt, maar wordt naar verwachting alleen aangeroepen voor kleine verzamelingen.

De laatste procedure, `match`, zie figuur 3.6, kunnen we aanroepen om een match te construeren aan de hand van twee gezins- of twee persoonsknopen. Hij slaat de gevonden match, zo er één is, op in drie tabellen, M, PM en GM. In de eerste tabel M wordt een match-*id* opgeslagen samen met de kwaliteit en grootte van de match. In PM worden de matches tussen persoonsknopen opgeslagen met de matchkwaliteit en een match-*id* om aan te geven bij welke match ze horen. In GM worden op dezelfde manier matches tussen gezinsknopen bewaard. We definiëren de tabellen als volgt:

$$M : (\begin{pmatrix} \text{m-id} \\ \text{d} \\ \text{p} \\ \text{g} \end{pmatrix}, (\text{m-id}), \emptyset, \emptyset) \tag{3.29}$$

Hier is m-id het nummer van de match, d de gemiddelde kwaliteit van de match en zijn p en g het respectievelijke aantal matches tussen persoonsknopen en tussen gezinsknopen waaruit de match bestaat. Deze worden afzonderlijk opgeslagen in tabellen PM en GM:

$$PM : (\begin{pmatrix} \text{m-id} \\ \text{p-id-paar} \\ \text{d} \end{pmatrix}, (\text{m-id,p-id-paar}), \emptyset, \emptyset) \tag{3.30}$$

$$GM : (\begin{pmatrix} \text{m-id} \\ \text{g-id-paar} \\ \text{d} \end{pmatrix}, (\text{m-id,g-id-paar}), \emptyset, \emptyset) \tag{3.31}$$

met

$$\text{p-id-paar} = \{\text{p1-id}, \text{p2-id}\} \tag{3.32}$$

$$\text{g-id-paar} = \{\text{g1-id}, \text{g2-id}\} \tag{3.33}$$

De records uit deze tabellen houden in hun m-id-veld bij bij welke match ze horen. De velden p1-id en p2-id geven aan op welke persoonsknopen de match van toepassing is en de velden g1-id en g2-id identificeren de gezinsknopen. Het d-veld geeft de kwaliteit van de match aan.

3.5.1 Velden met ongeordende gegevens

We gebruiken in de structuurdefinities van PM en GM velden met ongeordende gegevens: p-id-paar en g-id-paar. Bij een dergelijk veld maakt de volgorde niet uit. We stellen immers vast dat een match tussen persoonsknopen met p-id 10 en 20 ook een match is tussen knopen 20 en 10. Idem voor gezinsknopen. Door de twee id's ongeordend op te slaan voorkomen we dat we paren id's twee keer moeten op slaan.

In de praktijk van bestaande databanksystemen is dat niet mogelijk. Dergelijke problemen worden daar op één van twee manieren opgelost. Beide methoden maken gebruik van twee velden, bijvoorbeeld p1-id en p2-id, waarbij de twee waarden met een van te voren afgesproken volgorde worden opgeslagen in een record. Bij de eerste oplossing wordt er één record per paar gemaakt en staan de paren in een vaste volgorde. In dit voorbeeld zou p1-id altijd de kleinste id kunnen bevatten en p2-id de grootste. Wanneer we de tabel vervolgens doorzoeken op id-paren moeten we gezochte id's wel van te voren op volgorde zetten of twee keer zoeken. Wanneer we zoeken naar alle matches op één id zullen we altijd twee keer moeten zoeken, een keer in p1-id en een keer in p2-id. Als alternatief zouden we kunnen zoeken in een *view*, een virtuele logische tabel, die de reële tabel verenigt met een versie van zichzelf waarbij de id-velden omgedraaid zijn:

$$\begin{aligned} PM\text{-view} &= PM \cup \{(\text{m-id}, \text{p2-id}, \text{p1-id}, \text{d}) : \\ &\qquad (\text{m-id}, \text{p1-id}, \text{p2-id}, \text{d},) \in PM\} \qquad (3.34) \\ Q_i &= \{r \in PM\text{-view} : r(\text{p1-id}) = i\} \qquad (3.35) \end{aligned}$$

Hier is PM-view de virtuele tabel die gedefinieerd is op basis van PM, i de id die we zoeken en Q_i de verzameling van alle persoonsknoopmatches op i. Qua implementatie levert het gebruik van een dergelijke *view* geen voordelen op. De queries om op id te zoeken worden wel eleganter.

De tweede oplossing bestaat uit het toevoegen van dubbele records. Voor ieder record met id-paar (i,j) wordt ook een record met paar (j,i) toegevoegd, behalve als $i = j$. Dat heeft enkele nadelen. Ten eerste wordt de tabel potentieel twee keer zo groot en voor met mogelijk de helft gevuld met redundante data. Ten tweede moet er moeite worden gedaan om de tabel consistent te houden. De data bij (i,j) moeten wel hetzelfde blijven als die bij (j,i). Het voordeel is echter dat zoek-queries simpeler en eleganter worden en relatief snel, omdat geschikte indices op de data gemaakt kunnen worden.

Wat betreft implementatie hebben we gekozen voor de tweede optie. De tabellen waarbij we willen zoeken op paren of helften van paren worden veelvuldig doorzocht. Om de queries logisch en snel te houden verdubbelen we dus de data. In de formele beschrijving van de algoritmen en datastructuren van dit project willen we dit

compromis echter niet spiegelen en hanteren we de geïdealiseerde oplossing van een veld dat bestaat uit ongeordende waarden.

In de formules 3.30 op pagina 40 en 3.31 op de pagina hiervoor beschouwen de velden p-id-paar en g-id-paar ook nog eens als deel van de primaire index. We indexeren de ongeordende waarden voor iedere mogelijke volgorde van de waarden. Op die manier kunnen we het veld altijd doorzoeken op alle of slechts enkele van de waarden. De index wordt wel aanzienlijk groter. Het aantal mogelijke volgordes is immers gelijk aan de faculteit van de verschillende waarden in het veld.[3] Gelukkig zullen we geen velden gebruiken met meer dan twee ongeordende waarden. Bij de velden p-id-paar en g-id-paar kunnen we dus voor een gegeven m-id snel zoeken op zowel twee id's, dankzij de index, als op één id, dankzij de prefix van de index.

In de volgende hoofdstukken zullen we meer tabellen introduceren met velden met ongeordende waarden.

3.5.2 Het aanroepen van het algoritme

Matches van type A zullen we evalueren door de functie `match` aan te roepen voor één gezinsknoop van beide akten. Van de ene akte nemen we de gezinsknoop met de rol *huwenden* en van de andere met de rol *ouders bruidegom* of *ouders bruid*. In de voorbeeldmatch van figuur 2.3 op pagina 23 zouden we gezinsknopen s en r als parameters aan de `match`-functie meegeven. Deze functie zal vanzelf knopen g met e en h met f vergelijken. Knopen p en q en verder worden buiten beschouwing gelaten, omdat in de andere familiegraaf geen equivalente knopen bestaan. Hetzelfde geldt voor knoop k.

Voor matches van type B beginnen we de vergelijking met gezinsknopen van beide akten in de rol van óf *ouders bruidegom* óf *ouders bruid*. Voor de match in figuur 2.4 op pagina 24 betekent dit dat we `match` aanroepen voor p en s. Hierdoor worden p en s natuurlijk vergeleken en vervolgen a met g, en b met h. Daarnaast wordt ook de vergelijking e met k geprobeerd. Als die laatste niet slaagt, kan de rest van de match toch goed zijn. Slaagt hij daarentegen wel, dan kan de match zich ook nog verder uitbreiden over beide gehele akten.

Omdat we het liefste stelletjes matchen roepen we het algoritme over het algemeen altijd aan voor gezinsknopen. Op het zoeken naar de juiste gezinsknopen om de evaluatie mee te beginnen zullen we in hoofdstuk 5 nader ingaan.

[3] Dit geldt niet wanneer waarden meer dan één keer voorkomen. Als er n verschillende waarden in het veld staan, waarbij de i-de waarde x_i keer voorkomt, dan zijn er $\left(\sum_{i=1}^{n} x_i\right)! \left(\prod_{i=1}^{n} x_i!\right)^{-1}$ mogelijke volgorden. Zo is er bij twee dezelfde waarden inderdaad maar één mogelijke volgorde.

Hoofdstuk 4

Verwante voornamen

In het vorige hoofdstuk hebben we het verschil tussen voornamen bepaald aan de hand van een *edit-distance*, de Levenshteinafstand. We hebben daarbij aangenomen dat het verschil tussen namen klein is als het verschil in de schrijfwijze klein is. Zo kunnen we namen matchen met kleine spellingvariaties en compenseren voor spelfouten. Maar kan het zijn dat het verschil tussen namen klein is, terwijl het verschil in schrijfwijze toch groot is?

In het voorbeeld van het vorige hoofdstuk zagen we twee namen waarbij het verschil in schrijfwijze groot is, terwijl de namen eigenlijk heel verwant zijn, namelijk "Jan" en "Johannes". Een match tussen deze namen wordt op voorhand afgewezen omdat de relatieve Levenshteinafstand boven de 0,5 ligt, namelijk op 0,625. Maar de namen zijn verwant. De naam "Jan" is immers een afkorting van de naam "Johannes".

Zo zijn er meer namen die dicht bij elkaar liggen, maar heel anders geschreven worden. Het kan gaan om Nederlandse namen die van dezelfde stamnaam afstammen, zoals "Gerda" en "Trui", beide afkomstig van "Gertruida", maar ook om verwante namen in andere talen, zoals "Lodewijk" in het Nederlands, "Louis" in het Frans en "Ludwig" in het Duits. Een speciaal geval is de Latijnse variant van de naam. Van "Lodewijk" is dit "Ludovicus".

Over het algemeen geven mensen op verschillende akten dezelfde namen en kiezen niet iedere keer een andere variant van hun naam. Toch komt het voor dat naamsvarianten door elkaar worden gebruikt. Met name in de provincie Limburg worden Nederlandse, Franse, Latijnse en somse Duitse namen door elkaar gebruikt. Voor deze *verwante* namen willen we een andere verschilwaarde dan die de Levenshteinafstand oplevert.

We willen graag deze verwantschap tussen namen vaststellen en meetellen in de matchkwaliteit. In dit hoofdstuk zullen we hiervoor de voornaamverschilfunctie aanpassen. Daarnaast bekijken we een algoritme dat op basis van een collectie huwelijksakten deze verwantschappen kan opsporen.

4.1 Een nieuwe verschilmaat

Voor het verschil tussen losse voornamen stellen we een functie voor die de relatieve Levenshteinfunctie gebruikt wanneer namen niet verwant zijn en een andere waarde, als de $\mathcal{L}$-afstand groot is terwijl de namen verwant zijn:

$$d^{\text{lvn}}(\alpha,\beta) = \begin{cases} \mathcal{L}(\alpha,\beta) & , \text{ als } \{\alpha,\beta\} \notin V \vee \mathcal{L}(\alpha,\beta) < d_{\text{v}} \\ d_{\text{v}} & , \text{ als } \{\alpha,\beta\} \in V \wedge \mathcal{L}(\alpha,\beta) \geq d_{\text{v}} \end{cases} \tag{4.1}$$

α, β: losse voornamen

$\mathcal{L}$	Peter	Jan	Wilhem
Johannes	0,875	0,100	0,875
Wilhelm	0,857	1	0,143
Peter	0	1	0,833
Gerard	0,667	0,833	1

Tabel 4.1: De nieuwe afstanden tussen de twee sets voornamen

d_{v}**:** de d-waarde voor verwante voornamen die toch een grote $\mathcal{L}$-waarde hebben

V**:** de verzameling van paren namen die we als verwant beschouwen

Als we er vanuitgaan dat "Jan" en "Johannes" verwante namen zijn en dat dus {"Jan","Johannes"} $\in V$ en we nemen voor d_{v} de waarde 0,1, dan wordt de eindscore voor het voorbeeld in hoofdstuk 3 op pagina 25 een stuk beter. Tabel 4.1 bevat de nieuwe verschillen tussen de namen. Voorlopig nemen we aan dat {"Wilhem", "Wilhelm"} $\notin V$. Alle d-waarden zijn nu beter:

$$\begin{aligned} d^{\mathrm{lvn}}(\alpha,\beta) &= \sqrt{\tfrac{1}{3}\left(0^2 + 0{,}1^2 + 0{,}143^2\right)} = 0{,}101 \\ d^{\mathrm{ovn}}(\alpha,\beta) &= \tfrac{1}{7} = 0{,}143 \\ d^{\mathrm{vvn}}(\alpha,\beta) &= \tfrac{1}{2} = 0{,}500 \end{aligned}$$

De totale d^{vn} wordt nu:

$$d^{\mathrm{vn}}(\alpha,\beta) = 0{,}143 + (1 - 0{,}143)\left(1 - \frac{1 - 0{,}101}{1 + 1 \times 0{,}500}\right) = 0{,}486 \qquad (4.2)$$

Dit is al een stuk beter. Ondanks dat α en β behoorlijk verschillen komt de d^{vn}-waarde nu onder de $d^{\mathrm{vn}}_{\mathrm{max}} = 0{,}5$ uit.

In formule 4.1 op de vorige pagina maken we gebruik van V, de verzameling van paren namen die we als verwant beschouwen. Hoe bepalen we deze verzameling? We stellen een algoritme voor dat aan de hand van een collectie data als de huwelijksakten een dergelijke verzameling op kan bouwen.

4.2 Het verwantschapsalgoritme

Het *verwantschapsalgoritme* bestaat uit twee stappen. Beide stappen maken gebruik van een lijst waarin voor paren namen twee getallen, p en n, worden opgeslagen. Tijdens de eerste stap worden namen gezocht die onderling verwant zijn. Voor combinaties wordt het getal p (positief) ingevuld. Tijdens de tweede stap worden de n-waarden (negatief) ingevuld, voor namen die juist niet verwant lijken te zijn. Vele namencombinaties krijgen zowel een n- als een p-waarde.

Het algoritme analyseert een bestaande datacollectie. De resultaten hangen dan ook van deze dataset af en hoeven niet foutloos te zijn. Ook leveren verschillende datasets verschillende lijsten namen op. Die kunnen echter gemakkelijk gecombineerd worden.

Om de lijst op te slaan maken we de tabel LM (matches tussen losse voornamen). Deze heeft de volgende structuur.

$$LM = (\begin{pmatrix} \text{lvn-paar} \\ \text{vp} \\ \text{vn} \\ \text{v} \end{pmatrix}, (\text{lvn-paar}), \emptyset, \emptyset) \qquad (4.3)$$

met

$$\text{lvn-paar} = \{\text{voornaam1}, \text{voornaam2}\} \tag{4.4}$$

We zullen deze tabel in hoofdstuk 5 ook nog voor andere doeleinden gebruiken. Merk op dat het veld lvn-paar ongeordende gegevens bevat. Zie daarvoor paragraaf 3.5.1 op pagina 41.

4.2.1 De eerste stap

De eerste stap van het naamverwantschapsalgoritme bestaat uit het zoeken naar vermeldingen van dezelfde personen in een dataset waarbij de voornamen toch veranderen. Dat klinkt enigszins tegenstrijdig. Hoe weten we of twee vermeldingen dezelfde persoon betreffen, zeker als de voornamen veranderen? Hiervoor nemen we het criterium dat geen van de andere velden veranderd mag zijn en we beschouwen de velden van de partner als velden van de persoon in kwestie. We doen namelijk de volgende aannames:

- Wanneer van twee persoonsknopen voorvoegsel en achternaam hetzelfde zijn en de voornamen, voorvoegsel en achternaam van de partners ook, dan verwijzen de twee persoonsknopen waarschijnlijk naar dezelfde persoon, ongeacht verschil in voornamen.
- Als twee persoonsknopen naar dezelfde persoon verwijzen maar verschillende voornamen bevatten, zijn de voornamen waarschijnlijk onderling verwant.

Bijvoorbeeld, we treffen in een collectie huwelijksakten twee keer een meneer Willem (achternaam) aan met zijn vrouw Elizabeth Joanna de Swart. De voornaam van de eerste is “Justus”, van de tweede “Joost”. We gaan er nu van uit dat de kans groter is dat deze meneren Willem dezelfde persoon zijn, dan dat er twee keer een meneer Willem getrouwd is met een Elizabeth Joanna de Swart. Daarom beschouwen we de voornamen “Justus” en “Joost” als waarschijnlijk verwant.

Dat gaat natuurlijk niet altijd goed. Het voorbeeld kan best een zwagerhuwelijk[1] zijn geweest, of er zijn inderdaad twee meneren Willem met één of twee juffrouwen Elizabeth Joanna de Swart getrouwd. We doen er dus verstandig aan om rekening te houden met een foutmarge.

Per paar voornamen tellen we hoe vaak dit voorkomt. Dat getal p slaan we op in LM in het veld vp. Met het algoritme in figuur 4.1 op de volgende pagina voeren we stap 1 uit. We lopen in dit algoritme alle persoonsknopen van beide geslachten langs. De keuze van de velden kan zich natuurlijk uitstrekken tot alle soorten informatie die constant blijft ondanks voornaamsverandering. We beschouwen hier velden van de partner dan ook als identificerend voor de persoon in kwestie.

Merk op dat we de voornamen van α en β in volgorde koppelen, α_1 met β_1, α_2 met β_2 en zo voort, terwijl we in het vorige hoofdstuk grote moeite hebben gedaan om een intelligente koppeling te maken. Uit de praktijk is gebleken dat het koppelen van de losse voornamen aan de hand van bijvoorbeeld de Levenshteinafstand weinig verschil maakt in de uiteindelijke aantallen. Met beide methoden onstaat er een groot aantal fout-positieven die noodzaken tot stap 2 van het algoritme. Daarom kiezen we uit efficiëntieoverwegingen voor de eerste methode.

In stap 2 van het algoritme tellen we hoe vaak twee losse voornamen in één voornamenveld voorkomen. We nemen namelijk aan dat wanneer twee losse voornamen aan één persoon worden gegeven, ze waarschijnlijk niet onderling verwant zijn. Met

[1] Het zwagerhuwelijk of leviraat is het huwelijk tussen een weduwe en de broer van haar overleden man.

```
for each (a ∈ P) ⋈_{getrouwd(a,b)} (b ∈ P)
  ⋈ ⟨a(geslacht) = c(geslacht) ∧ a(voornamen) ≠ c(voornamen)
    ∧a(voorvoegsel) = c(voorvoegsel) ∧ a(achternaam) = c(achternaam)
    ∧b(geslacht) = d(geslacht) ∧ b(voornamen) = d(voornamen)
    ∧b(voorvoegsel) = d(voorvoegsel) ∧ b(achternaam) = d(achternaam)⟩
  ((c ∈ P) ⋈_{getrouwd(c,d)} (d ∈ P))
do
  α ← a(voornamen)
  β ← c(voornamen)
  for i ← 1 to min(|α|, |β|) do
    if ∃r ∈ LM : r(lvn-paar) = {α_i, β_i} then
      r(vp) ← r(vp) + 1
    else
      LM ← LM ∪ {{(lvn-paar, {α_i, β_i}), (vp, 1), (np, 0), (v, null)}}
    fi
  od
od
```

Figuur 4.1: Het naamverwantschapsalgoritme, stap 1

```
for each (p ∈ P) do
  α ← p(voornamen)
  for i ← 1 to |α| − 1 do
    for j ← 2 to |α| do
      if ∃r ∈ LM : r(lvn-paar) = {α_i, α_j} then
        r(np) ← r(np) + 1
      else
        LM ← LM ∪ {{(lvn-paar, {α_i, β_i}), (vp, 0), (np, 1), (v, null)}}
      fi
    od
  od
od
```

Figuur 4.2: Het naamverwantschapsalgoritme, stap 2

andere woorden, als een persoon voornamen α_1 tot en met α_n heeft, zijn al deze voornamen waarschijnlijk onderling niet verwant. We verwachten dus geen namen als "Piet Petrus" of "Wilhelmus Wil" in de dataset aan te treffen. Het algoritme in figuur 4.2 vult lijst LM met n-waarden.

Om te bepalen of twee voornamen verwant zijn, vergelijken we de getallen aan de hand van twee parameters, g en h. Die worden proefondervindelijk bepaald.

$$\{\alpha,\beta\} \in V \quad \longleftrightarrow \quad p_{\{\alpha,\beta\}} \geq g \quad \wedge \quad p_{\{\alpha,\beta\}} \geq h \cdot n_{\{\alpha,\beta\}} \tag{4.5}$$

Deze laatste stap wordt weergegeven in figuur 4.3 op de rechter pagina, waar het Boolese veld v aangeeft of het namenpaar verwant is. Naar aanleiding van de resultaten op één provincie, Limburg, hebben we g op 10 gesteld. Hiermee verwijderen een grote hoeveelheid fout-positieven die een n-telling van 0 hebben. We hebben h op 25 gezet. Paren voornamen die een zeer hoge p-telling hebben blijven dus positief als de n-telling laag genoeg is.

De g-parameter is afhankelijk van het aantal akten dat we gebruiken voor het naamverwantschapsalgoritme. De resultaten van van verschillende datasets kunnen worden gecombineerd door de aantallen in de lijsten per paar voornamen op te tellen. De g-waarde moet dan verhoogd worden omdat de incidentele p-waarden van niet verwante paren hoger zal zijn. Hoewel bij gelijksoortige data de verhoging van

```
for each (r ∈ LM) do
  if r(vp) ≥ g ∧ r(vp) ≥ h · r(np) then
    r(v) ← waar
  else
    r(v) ← onwaar
  fi
od
```

Figuur 4.3: Het naamverwantschapsalgoritme, stap 3

g-waarde proportioneel is aan de vehoging van het aantal akten, geldt dat niet in het algemeen. Bij ons project is gebleken dat bij het combineren van verwantschapstellingen die gebaseerd zijn op verschillende provincies, g en soms h aangepast moeten worden om de V zo goed mogelijk te krijgen.

Hoofdstuk 5

De heuristiek

In hoofdstukken 3 op pagina 25 en 4 op pagina 43 bespraken we hoe we kunnen bepalen of een match goed is. We kunnen nu voor iedere mogelijke match een d-waarde uitrekenen. Hoe dichter die bij 0 zit, hoe beter de match is. Als we nu alle mogelijke matches tussen alle akten zouden evalueren en degene met lage d-waarden op zouden slaan, zouden we alle goede matches vinden. Maar dit duurt veel te lang. Als we bijvoorbeeld de d-waarden voor alle mogelijke matches van type A zouden willen analyseren voor de collectie Zeeland, de kleinste met 164.498 aktes, zouden we 328.996 ouderparen moeten vergelijken met 164.498 bruidsparen. Dit zou 541.119.184.008 matches opleveren. Dat zijn er te veel. Als we tienduizend vergelijkingen per seconde zouden kunnen uitvoeren, zouden we iets meer dan twee maanden moeten wachten tot we klaar zijn. Dan blijven er nog vier grotere provincies over en we willen ze het liefst ook nog onderling matchen. Hoe kunnen we dan goede matches vinden? We zullen door middel van een heuristiek het grootste deel van de mogelijke matches van te voren moeten elimineren, zodat we slechts een klein aantal matches hoeven te evalueren.

Zoals we in paragraaf 3.5.2 op pagina 42 bespraken, zijn we op zoek naar paren gezinsknopen die een goede match opleveren. Voor matches van type A willen we paren waarbij de eerste gezinsknoop de rol *ouders bruid* of *ouders bruidegom* heeft, en de tweede de rol *huwenden*. Laten we een heuristiek opstellen die een verzameling Q van zulke paren oplevert die vele malen kleiner is dan de maximale verzameling van alle mogelijke paren van zulke knopen.

5.1 Perfecte matches

Laten we om te beginnen zoeken naar perfecte matches. Zoals in de inleiding besproken vinden we hiermee niet alle gewenste matches, maar wel een aanzienlijk deel. Een bijkomend voordeel is dat de heuristiek daarvoor niet alleen simpel en voor de hand liggend, maar ook nog eens snel is en weinig rekenkracht vergt.

variabele	aantal iteraties	aantal records dat voldoet
g	$\|G\|$	$\frac{2}{3}\|G\|$
a	$\|P\|$	1
b	$\|P\|$	1
c	$\|P\|$	k_1
h	$\|G\|$	1
d	$\|P\|$	k_2

Tabel 5.1: Aantal iteraties bij het naïef opbouwen van Q

We kunnen Q als volgt vullen:

$$
\begin{aligned}
Q \;=\; & \sigma_{(g,h)}\big((g \in G)\langle g(\text{rol}) \in \{\text{oudersbruidegom}, \text{oudersbruid}\}\rangle \\
& \bowtie (a \in P)\langle \text{vrouwgezin}(a, g)\rangle \\
& \bowtie (b \in P)\langle \text{mangezin}(b, g)\rangle \\
& \bowtie (c \in P)\langle a(\text{voornaam}) = c(\text{voornaam}) \\
& \qquad \wedge\, a(\text{voorvoegsel}) = c(\text{voorvoegsel}) \\
& \qquad \wedge\, a(\text{achternaam}) = c(\text{achternaam}) \\
& \qquad \wedge\, a(\text{gb-min}) \leq c(\text{gb-max}) \\
& \qquad \wedge\, a(\text{gb-max}) \geq c(\text{gb-min}) \\
& \qquad \wedge\, c(\text{rol}) = \text{bruid}\rangle \\
& \bowtie (h \in G)\langle \text{vrouwgezin}(c, h)\rangle \\
& \bowtie (d \in P)\langle b(\text{voornaam}) = d(\text{voornaam}) \\
& \qquad \wedge\, b(\text{voorvoegsel}) = d(\text{voorvoegsel}) \\
& \qquad \wedge\, b(\text{achternaam}) = d(\text{achternaam}) \\
& \qquad \wedge\, b(\text{gb-min}) \leq d(\text{gb-max}) \\
& \qquad \wedge\, b(\text{gb-max}) \geq d(\text{gb-min}) \\
& \qquad \wedge\, \text{mangezin}(d, h)\rangle\big)
\end{aligned}
\tag{5.1}
$$

a en b: de (knopen van de) moeder en vader van een bruid of bruidegom

g: de gezinsknoop die het huwelijk tussen a en b voorstelt

c en d: de (knopen van de) bruid en bruidegom van een andere akte (met dezelfde namen als a en b)

h: de gezinsknoop die het huwelijk tussen c en d voorstelt

P: de verzameling van persoonsknopen

G: de verzameling van gezinsknopen

In deze formule wordt de selectiefunctie σ gebruikt, die van een gegeven tabel slechts een aantal geselecteerde kolommen weergeeft. In formule 5.1 reduceert de σ-functie een tabel met tupels (g, a, b, c, h, d) tot een tabel met tupels (g, h).

Om te kunnen beoordelen of deze resultatenset snel en efficiënt opgebouwd kan worden, moeten we ons verdiepen in hoe hij stap voor stap wordt samengesteld. In Q zitten alle combinaties van zes records die aan de opgegeven eisen voldoen. Figuur 5.1 op de pagina hiernaast toont een naïeve wijze waarop we Q zouden kunnen vullen. Deze methode is nog minder efficiënt dan het worst case scenario aan het begin van het hoofdstuk. Hoe inefficiënt het is wordt inzichtelijk in tabel 5.1. Voor iedere loop geven we aan hoeveel records er doorlopen moeten worden en voor hoeveel van deze records we verwachten dat de *if*-voorwaarde waar is. Bij die records wordt de volgende loop ook gestart. Het totale aantal te bestuderen

```
Q ← ∅
for each g ∈ G do
  if g(rol) = oudersbruidegom ∨ g(rol) = oudersbruid then
    for each a ∈ P do
      if a(p-id) = g(vrouw-p-id) then
        for each b ∈ P do
          if b(p-id) = g(man-p-id) then
            for each c ∈ P do
              if c(voornaam) = a(voornaam)
                ∧ c(voorvoegsel) = a(voorvoegsel)
                ∧ c(achternaam) = a(achternaam)
                ∧ c(gb-min) ≤ a(gb-max)  ∧  c(gb-max) ≥ a(gbmin)
                ∧ c(rol) = bruid
              then
                for each h ∈ G do
                  if h(vrouw-p-id) = c(p-id) then
                    for each d ∈ P do
                      if d(p-id) = h(man-p-id)
                        ∧ d(voornaam) = b(voornaam)
                        ∧ d(voorvoegsel) = b(voorvoegsel)
                        ∧ d(achternaam) = b(achternaam)
                        ∧ d(gb-min) ≤ b(gb-max)  ∧  d(gb-max) ≥ b(gbmin)
                        ∧ d(rol) = bruidegom
                      then
                        Q ← Q ∪ {(g, h)}
                    fi od
                fi od
            fi od
        fi od
    fi od
fi od
```

Figuur 5.1: Een naïeve manier om Q te bepalen

combinaties is dus:

$$|G| + \frac{2}{3}|G|(|P| + 1(|P| + 1(|P| + k_1(|G| + 1(|P| + k_2))))) \tag{5.2}$$

Hoeveel records c en d zullen voldoen aan de uitgebreidere *if*-voorwaarden (identieke namen, de juiste rol en het juiste geboortejaarbereik) is niet precies te voorspellen en verschilt ook per record. Zo zal "Klaas de Boer" vaker voorkomen dan "Marie Charlotte Elisabeth Pape". De waarden k_1 en k_2 stellen het gemiddelde aantal vondsten voor, waarbij in acht wordt genomen dat k_2 mede wordt bepaald doordat er via c en h al een subselectie is gemaakt. Bij Zeeland vinden we voor 34% van de akten een perfecte match. Hier geldt dus dat $k_1 k_2 = 0{,}34$. Laten we voor het gemak aannemen dat $k_1 = k_2 = \sqrt{0{,}34}$. Het volgen van deze implementatie voor Zeeland zou dan ongeveer 1,257,457,178,374 (1.3 biljoen) iteraties vergen. Gelukkig kan het een stuk sneller.

Als we bovenstaande implementatie zouden volgen, zouden we voor iedere gezinsknoop g alle persoonsknopen a aflopen om degene te vinden met de juiste p-id-waarde. Het veld p-id is in tabel P echter de primaire sleutel. De databank zal daar dankbaar gebruik van maken om aan de hand van een gegeven p-id vrijwel onmiddelijk het gezochte record te kunnen vinden. Dit zorgt ervoor dat we voor a, b en d niet de gehele tabel hoeven te doorzoeken. Ook h kunnen we snel vinden. We weten weliswaar niet de primaire sleutelwaarde (g-id), maar wel de waarde van vrouw-p-id. Op deze kolom hebben we tabel G geïndexeerd en daarom kunnen we records in G aan de hand van een vrouw-p-id ook onmiddelijk vinden.

Deze standaard databankmechanismen zorgen ervoor dat we onszelf niet nodeloos vertragen en veel moeite moeten doen om de in persoonsknopen en gezinsknopen gesplitste akten weer bij elkaar te zoeken. Het vinden van c kost echter nog veel tijd. Ook dat kan echter snel verholpen worden door de volgende index toe te voegen aan P: (voornaam, voorvoegsel, achternaam, gb-min, gb-max, rol). Het zal de databank wat tijd kosten om een sortering op deze velden te maken en op te slaan als index, maar omdat sorteren efficiënt kan worden uitgevoerd in tijd in orde van $n \log n$, waarbij n de hoeveelheid te sorteren gegevens is, duurt dit niet lang. Overigens zullen de meeste prefixen van de voorgestelde index ook volstaan zoals de voor de hand liggende (voornaam, voorvoegsel, achternaam). Per combinatie van waarden in deze velden zullen over het algemeen zo weinig records worden aangetroffen dat het maar de vraag is of verder indiceren de overhead waard is.

Als we toch bezig zijn met indices toevoegen, maken we ook meteen een index in G voor het veld rol. Dat scheelt ons werk in de buitenste loop. De implementatie van Q kan nu een stuk efficiënter. Om aan te geven dat we voor een voorwaarde om twee tabellen te koppelen een index kunnen gebruiken, zullen we die voorwaarden niet als *if*-voorwaarde implementeren, maar opnemen in de *for each*-uitdrukking. Figuur 5.2 op de pagina hiernaast laat zien hoe we Q kunnen bepalen door gebruik te maken van indices. De *for each*-uitdrukkingen bij a, b, h en d zijn symbolisch geworden, omdat er maximaal 1 record aan de eisen kan voldoen. Merk op dat het bij d geen zin heeft om de extra gedefinieerde index te gebruiken. Van de gezochte record kennen we immers de waarde van de primaire sleutel en kunnen we er direct naartoe springen en de record testen.

In tabel 5.2 op de rechter pagina zien we dat we bij het opbouwen van Q nu voor ieder record een index kunnen gebruiken. Het totale aantal te bekijken records is nu $\frac{2}{3}|G|k_1$. Voor Zeeland zou dit met dezelfde aannames uitkomen op 120.132 records. Dat is goed te doen. In de praktijk is het iets meer werk. We doen nu alsof het zoeken van records met behulp van indices geen tijd kost, maar in werkelijkheid kost het tijd in de orde van $\log n$ of minder, waarbij n het aantal records in de tabel is. Daarnaast is er nog wat overhead die veroorzaakt wordt door de hardware, zoals geheugenbeperkingen en zoektijden van harde schijven. Daarover later meer. Hier

```
Q ← ∅
for each g ∈ G : g(rol) = oudersbruidegom ∨ g(rol) = oudersbruid) do
  for each a ∈ P : a(p-id) = g(vrouw-p-id) do
    for each b ∈ P : b(p-id) = g(man-p-id) do
      for each c ∈ P : c(voornaam) = a(voornaam)
        ∧ c(voorvoegsel) = a(voorvoegsel)
        ∧ c(achternaam) = a(achternaam)
        ∧ c(gb-min) ≤ a(gb-max)  ∧  c(gb-max) ≥ a(gbmin)
        ∧ c(rol) = bruid
      do
        for each h ∈ G : h(vrouw-p-id) = c(p-id) do
          for each d ∈ P : d(p-id) = h(man-p-id) do
            if d(voornaam) = b(voornaam)
              ∧ d(voorvoegsel) = b(voorvoegsel)
              ∧ d(achternaam) = b(achternaam)
              ∧ d(gb-min) ≤ b(gb-max)  ∧  d(gb-max) ≥ b(gbmin)
              ∧ d(rol) = bruidegom
            then
              Q ← Q ∪ {(g, h)}
            fi
          od
        od
      od
    od
  od
od
```

Figuur 5.2: Het bepalen van Q met behulp van indices

variabele	benutte index	aantal iteraties
g	(rol)	$\frac{2}{3}\|G\|$
a	(p-id)	1
b	(p-id)	1
c	(voornaam, voorvoegsel, achternaam, gb-min, gb-max, rol)	k_1
h	(vrouw-p-id)	1
d	(p-id)	1

Tabel 5.2: Aantal iteraties bij opbouwen van Q met behulp van indices

geeft $\frac{2}{3}|G|k_1$ een goede indicatie dat de hoeveelheid werk die bij de bepaling van Q komt kijken niet al te groot is.

5.1.1 De verwerking van Q

Wanneer we Q eenmaal uitgerekend hebben, rest ons nog de taak om de aangeboden matches op te slaan in de tabellen M, PM en GM[1]. We volgen daarbij een systematiek die we vaker zullen hanteren. Die vloeit voort uit het feit dat we enerzijds een databank gebruiken om de grote hoeveelheden gegevens in op te slaan en anderzijds een cliëntapplicatie om bepaalde operaties mee uit te voeren die normaal gesproken niet efficiënt door een database uitgevoerd kunnen worden, zoals het uitrekenen van de Levenshteinafstand. We verdelen het werk als volgt:

In de database:
`Voer een zoekvraag of heuristiek uit.`
`Geef de resultaten aan de cliëntapplicatie als verzameling` V`.`

In de cliëntapplicatie:
`for each` $v \in V$ `do`
`  Voer een (complexe) operatie uit op` v`.`
`  Sla eventuele resultaten op in de database.`
`od`

Deze aanpak heeft geleid tot de volgende simpele implementatie van de verwerking van Q:

Q `wordt volgens SQL-specificatie gevuld door de database.`
`for each` $(g, h) \in Q$ `do`
`  match`(g, h)
`od`

De procedure `match` hebben we gedefinieerd in 3.5 op pagina 36. Deze procedure maakt een echte match tussen g en h en slaat de resultaten op in de database. Bij het maken van de match controleert de procedure alle verschilwaarden en wijst de match af als hij te slecht blijkt. Dat is in dit geval nooit nodig, omdat we Q zo gespecificeerd hebben dat het matchen van g en h altijd tot een perfecte match zal leiden. We kunnen echter bovenstaand algoritme gebruiken voor alle heuristieken en voor meer dan alleen huwelijksakten. De flexibiliteit van de `match`-procedure zorgt ervoor dat het niet uitmaakt hoe we aan de beginknopen van een match zijn gekomen; hij zal vanuit deze knopen een zo groot en zo goed mogelijke match opbouwen. Hierna zullen we een heuristiek R opbouwen die imperfecte maar goede matches oplevert. Deze zullen we verwerken aan de hand van bovenstaand algoritme door alleen de SQL-specificatie te veranderen in die van R. Daarnaast zullen we de systematiek van het zoeken met behulp van de database en het verwerken met de cliëntapplicatie ook toepassen op enkele problemen die we bij de opbouw van R zullen tegenkomen.

5.2 Imperfecte matches

Al met al is het zoeken van perfecte matches gemakkelijk te doen. De moeilijkheden duiken op wanneer we ook imperfecte matches zoeken. We zoeken een verzameling R van tupels gezinsknopen, die, wanneer we ze aan het evaluatiealgoritme geven,

[1] Zie pagina 40 voor de definitie van deze tabellen.

provincie	persoonsknopen	verschillende achternamen
Groningen	1.248.811	36.449
Overijssel	1.319.628	51.806
Gelderland	1.950.929	76.017
Zeeland	986.714	43.328
Limburg	1.130.325	65.393

Tabel 5.3: Aantal verschillende achternamen per provincie

een goede d^{M} opleveren, maar niet noodzakelijk één die 0 is. We kunnen de ideale R makkelijk definiëren door formule 5.1 van Q te nemen en deze aan te passen bij voornaam, voorvoegsel en achternaam:

$$
\begin{aligned}
R \;=\; & \sigma_{(g,h)}\Big((g \in G)\langle g(\mathrm{rol}) \in \{\mathrm{oudersbruidegom}, \mathrm{oudersbruid}\}\rangle \\
& \bowtie (a \in P)\langle \mathrm{vrouwgezin}(a, g)\rangle \\
& \bowtie (b \in P)\langle \mathrm{mangezin}(b, g)\rangle \\
& \bowtie (c \in P)\Big\langle d^{\mathrm{vn}}\big(a(\mathrm{voornaam}), c(\mathrm{voornaam})\big) \leq 0.5 \\
& \qquad \wedge\, \mathcal{L}\big(a(\mathrm{voorvoegsel}), c(\mathrm{voorvoegsel})\big) \leq 0.5 \\
& \qquad \wedge\, \mathcal{L}\big(a(\mathrm{achternaam}), c(\mathrm{achternaam})\big) \leq 0.5 \\
& \qquad \wedge\, a(\text{gb-min}) \leq c(\text{gb-max}) \\
& \qquad \wedge\, a(\text{gb-max}) \geq c(\text{gb-min}) \\
& \qquad \wedge\, c(\mathrm{rol}) = \mathrm{bruid}\Big\rangle \\
& \bowtie (h \in G)\langle \mathrm{vrouwgezin}(c, h)\rangle \\
& \bowtie (d \in P)\Big\langle d^{\mathrm{vn}}\big(b(\mathrm{voornaam}), d(\mathrm{voornaam})\big) \leq 0.5 \\
& \qquad \wedge\, \mathcal{L}\big(b(\mathrm{voorvoegsel}), d(\mathrm{voorvoegsel})\big) \leq 0.5 \\
& \qquad \wedge\, \mathcal{L}\big(b(\mathrm{achternaam}), d(\mathrm{achternaam})\big) \leq 0.5 \\
& \qquad \wedge\, b(\text{gb-min}) \leq d(\text{gb-max}) \\
& \qquad \wedge\, b(\text{gb-max}) \geq d(\text{gb-min}) \\
& \qquad \wedge\, \mathrm{mangezin}(d, h)\Big\rangle\Big)
\end{aligned}
\tag{5.3}
$$

De moeilijkheid zit hem natuurlijk in de d^{vn}- en $\mathcal{L}$-functies. Naamvelden zijn niet zo te sorteren dat namen met een kleine Levenshteinafstand snel te vinden zijn, laat staan namen met een kleine d^{vn}-waarde. Om records c te vinden zullen we dus een groot deel van P af moeten zoeken. We kunnen dit voorkomen door meer voorwerk te doen.

5.3 Het prematchen van achternamen

Het zoeken naar achternamen met kleine onderlinge Levenshteinafstanden kan wel in een databankomgeving als we de afstanden van te voren uitrekenen en opslaan. Om dit te doen voor ieder paar persoonsknopen levert natuurlijk geen tijdwinst op, maar we kunnen het wel doen voor paren achternamen. Het aantal verschillende achternamen is namelijk aanzienlijk kleiner dan het aantal persoonsknopen. In tabel 5.3 staat per provincie hoeveel persoonknopen en hoeveel verschillende achternamen er op de akten staan. Dit reduceert het probleem met meer dan een factor 20.[2] Maar nog steeds hebben we te maken met grote aantallen paren achternamen. Het aantal verschillende paren waarvoor we de Levenshteinafstand moeten uitrekenen is immers $\binom{n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$, waabij n het aantal verschillende achternamen is.

[2]Bij Limburg is het relatieve aantal verschillende achternamen opvallend groot. Een mogelijke verklaring hiervoor is een relatief groot personenverkeer tussen Limburg en de omringende buurlanden, waar Duits en Frans wordt gesproken.

eerste letter	aantal
a	1127
b	4506
c	1909
d	2428
e	1022
f	1236
g	2187
h	3484
i	272
j	586
k	2740
l	2510
m	2805
n	937
o	785
p	1850
q	118
r	2178
s	4016
t	1277
u	159
v	2492
w	2098
x	5
y	13
z	578
overig	10

Tabel 5.4: Aantal achternamen per eerste letter in Zeeland. De categorie *overig* bestaat uit namen met rare eerste letters als ! # $ % & ´ ( ) * + , . - / 0 1 2 5 8 ; < ?] Ö ˆ_ ` §\, © enzovoorts.

Dit betekent voor Zeeland met het minste aantal achternamen 939 miljoen paren en voor Gelderland met het grootste aantal bijna 3 miljard. We moeten nog meer paren elimineren om het een beetje efficiënt te krijgen. Overigens hebben we wel het voordeel dat er een grote overlap is in het aantal verschillende achternamen. Als we voor Zeeland alle Levenshteinafstanden hebben uitgerekend, kunnen we bij Gelderland veel namenparen overslaan die al in Zeeland voorkwamen.

Om het aantal paren te beperken, bekijken we alleen achternamen met dezelfde eerste letter. Dat houdt in dat we geen matches zullen gaan vinden tussen persoonsknopen wier achternamen met verschillende letters beginnen. We accepteren dit nadeel ten behoeve van de hoeveelheid werk die dit scheelt. In tabel 5.4 zien we hoeveel achternamen er per letter voorkomen in Zeeland. Als we voor iedere achternaam de Levenshteinafstand tot alle andere achternamen met dezelfde voorletter willen uitrekenen, moeten we deze afstand voor 55.574.041 namenparen uitrekenen. Dat is een groot aantal, maar het uitrekenen van de Levenshteinafstand gaat redelijk snel en uit de praktijk blijkt dat het een paar uur kost. We rekenen de afstand niet uit van namen tot zichzelf, die is immers altijd nul, of voor namen waarbij de kortste meer dan twee keer zo kort is als de langste. We doen echter wel wat extra werk om het nadeel van deze heuristiek te beperken. Zo vergelijken we namen uit de categorie *overig* met alle andere namen en vergelijken we namen met de voorletters

$d \in$	aantal
$[0{,}00; 0{,}05\rangle$	196.484
$[0{,}05; 0{,}10\rangle$	17.405
$[0{,}10; 0{,}15\rangle$	194.083
$[0{,}15; 0{,}20\rangle$	137.014
$[0{,}20; 0{,}25\rangle$	309.270
$[0{,}25; 0{,}30\rangle$	1.324.845
$[0{,}30; 0{,}35\rangle$	2.302.075
$[0{,}35; 0{,}40\rangle$	2.833.766
$[0{,}40; 0{,}45\rangle$	3.840.863
$[0{,}45; 0{,}50\rangle$	287.700
$\{0{,}50\}$	5.184.291

Tabel 5.5: Aantal paren achternamen per d-waarde voor alle vijf provincies.

s en z met elkaar, evenals f en t, c en k, en i met y. Die laatste vergelijken we met name om paren als IJzinga en Yzinga mee te nemen.

We slaan alle unieke achternamen op in een tabel A waarin we tevens bijhouden of de naam al met de andere namen gematcht is of niet. Wanneer we nieuwe namen uit een andere provincie toevoegen, kunnen we deze aan de hand van het veld *nieuw* apart houden, matchen met de namen die we al hadden en ook onderling. In de tabel AM slaan we paren achternamen op en de verschilwaarde.

$$A = (\begin{pmatrix} \text{achternaam} \\ \text{nieuw} \end{pmatrix}, (\text{achternaam}), \emptyset, \{(\text{nieuw})\}) \tag{5.4}$$

$$AM = (\begin{pmatrix} \text{an-paar} \\ \text{l} \\ \text{d} \end{pmatrix}, (\text{an-paar}), \emptyset, \{(\text{achternaam}, \text{d})\}) \tag{5.5}$$

met

$$\text{an-paar} = \{\text{achternaam1}, \text{achternaam2}\} \tag{5.6}$$

In het veld l slaan we de Levenshteinafstand op en in d de relatieve afstand. Op dit veld en op achternaam maken we een index, zodat we aan de hand van een achternaam alle gerelateerde achternamen kunnen vinden met een bepaalde d-waarde in gedachten. Omdat het veld *an-paar* ongeordend is, indiceert deze index op beide achternamen. Het veld {(an-paar, {Boer,Pape}),(l, 4), (d, 1)} zal dan ook zowel onder (Boer, 1) als onder (Pape, 1) terug te vinden zijn in de index. We zullen echter geen namen opslaan met een d-waarde van boven de 0,5. Dit scheelt wat rekenwerk. Niet alleen kunnen we, zoals vermeld, vergelijkingen overslaan tussen namen die meer dan een factor 2 in lengte verschillen, maar ook kunnen we aan het Levenshteinalgoritme een maximumwaarde meegeven, gelijk aan de helft van de lengte van het langste woord, als afbreekwaarde. Op het moment dat het Levenshteinalgoritme ziet dat de afstand groter wordt dan deze waarde breekt het voortijdig af en doet dus geen onnodig werk. De grootste winst is echter te behalen in ruimte. In tabel 5.5 zien we hoeveel paren achternamen voorkomen per d-waarde. Het aantal matches neemt sterk toe naarmate het verschil groter wordt. Het heeft geen zin om de meeste opslagcapaciteit te gebruiken voor de slechtste matches. Boven de 0,5 slaan we dan ook niets meer op.

De toename in tabel 5.5 lijkt enigzins grillig plaats te vinden. Dit komt doordat de d-waarde wordt berekend met een deling van twee gehele getallen. De meeste paren hebben dan ook een "ronde breuk" als verschilwaarde, een getal dat wordt verkregen door relatief kleine gehele getallen te delen. Zie tabel 5.6. Een speciale

d	$\tilde{d}$	aantal
0	0,000	196.193
$1/11$	0,091	9.236
$1/10$	0,100	18.303
$1/9$	0,111	32.050
$1/8$	0,125	54.683
$1/7$	0,143	87.886
$1/6$	0,167	106.621
$2/11$	0,182	25.592
$1/5$	0,200	135.853
$2/9$	0,222	159.589
$3/13$	0,231	8.999
$1/4$	0,250	441.681
$3/11$	0,273	80.080
$2/7$	0,286	797.229
$3/10$	0,300	282.301
$4/13$	0,308	24.521
$1/3$	0,333	1.990.160
$5/14$	0,357	30.497
$4/11$	0,364	301.451
$3/8$	0,375	2.425.601
$5/13$	0,385	65.019
$2/5$	0,400	2.069.931
$3/7$	0,429	708.599
$4/9$	0,444	972.589
$1/2$	0,500	5.184.291

Tabel 5.6: Aantal paren achternamen per d-waarde, die als "ronde breuk" uitgedrukt kan worden, voor alle vijf provincies.

groep matches is de eerste, van $d \in [0{,}00; 0{,}05\rangle$. Deze bestaat voor het grootste deel uit identiteitsmatches. De vijf provincies kennen immers 196.193 verschillende achternamen. Deze matchen natuurlijk met zichzelf met een d-waarde van 0.

Hetzelfde proces, het prematchen van de achternaam, kan ook worden uitgevoerd voor de voorvoegsels. Ook hier is de afstand gelijk aan de Levenshteinafstand. In dit project hebben we echter veel experimenten gedaan waarbij w_{vv} (het belang van verschillen in voorvoegsel) gelijk was aan 0. Om deze reden, en omdat het effect vergeleken met het prematchen op achternaam klein is, hebben we het prematchen van voorvoegsels niet geïmplementeerd.

5.4 Het prematchen van voornamen

Ook bij voornamen zouden we de verschilwaarde van te voren kunnen berekenen. De verschilfunctie voor voornamen is echter ingewikkelder, waardoor we niet dezelfde heuristiek kunnen toepassen als bij achternamen. De behoefte aan een goede heuristiek is echter nog groter. In tabel 5.7 op de pagina hiernaast zien we namelijk dat er nog meer verschillende voornamen zijn dan achternamen.

We kunnen niet de heuristiek van dezelfde eerste letter gebruiken, omdat we bij voornamen ook geïnteresseerd zijn in paren die niet dezelfde beginletter hebben. Zo kennen we het voornamenpaar "Frederik Rippert Nicolaas" en "Nicolaas Frederik Rippert" een relatief goede d-waarde toe van $1/3$. Bij de eerste voornamen staat echter een "F" vooraan en bij de tweede een "N". Hetzelfde geldt voor "Teun" en "Antheunis", al zou die combinatie door het verwantschapsalgoritme gevonden

provincie	persoonsknopen	verschillende voornamen
Groningen	1.248.811	135.546
Overijssel	1.319.628	69.781
Gelderland	1.950.929	106.353
Zeeland	986.714	57.900
Limburg	1.130.325	73.629

Tabel 5.7: Aantal verschillende voornamen per provincie

geslacht	verschillende voornamen
alleen mannelijk	176.909
alleen vrouwelijk	195.075
mannelijk en vrouwelijk	2.013
totaal	373.997

Tabel 5.8: Aantal verschillende voornamen per geslacht

kunnen worden. We moeten op zoek naar een andere heuristiek.

5.4.1 Het ontleden van voornamen

Hoewel we de heuristiek van de achternamen niet kunnen gebruiken voor voornamen, kunnen we hem wel gebruiken voor een deel van het probleem. We delen alle voornamen op in losse voornamen, net als in paragraaf 3.3 op pagina 28, en zullen deze gaan matchen. Om te beginnen maken we een tabel V voor unieke voornamen. Die kunnen in dit stadium nog bestaan uit meerdere losse voornamen.

$$V = (\begin{pmatrix} \text{voornamen} \\ \text{man} \\ \text{vrouw} \\ \text{aantallos} \end{pmatrix}, (\text{voornamen}), \emptyset, \emptyset) \tag{5.7}$$

voornamen: een set voornamen die voorkomt in een voornamenveld van tabel P

man: een Boolese waarde die waar is desda er een mannelijke persoonsknoop is met deze voornamen

vrouw: een Boolese waarde die waar is desda er een vrouwelijk persoonsknoop is met deze voornamen

aantallos: het aantal voornamen waar deze set voornamen uit bestaat

We zullen deze tabel vullen door te itereren over de persoonsknopen en de voornamen toevoegen als deze niet al aanwezig zijn in de tabel. De velden *man* en *vrouw* zijn Boolese waarden. Als de persoonsknoop een man betreft, worden zijn voornamen opgeslagen met het veld *man* als *waar* en de waarde *onwaar* in *vrouw*. Als de persoonsknoop een vrouw betreft, worden de waarden in *man* en *vrouw* andersom opgeslagen. Wanneer een persoonsknoop aan de beurt is wiens voornamen al in V staan, kijken we alleen naar de velden *man* en *vrouw*. Afhankelijk van het geslacht van de persoonsknoop zetten we het juiste veld op *waar* en laten het andere ongemoeid. Zo kan het voorkomen dat een naam die ingevoerd is als jongensnaam, ook de waarde *waar* kan krijgen in het veld *vrouw*. In tabel 5.8 zien we dat de overlap tussen jongensnamen en meisjesnamen klein is. We kunnen ons dan ook

werk besparen door alleen namen die één van de twee geslachten beide hebben te vergelijken. Eerst zullen we verder gaan door de voornamen te splitsen in losse voornamen. Daarvoor maken we een tabel L, waarin we de losse voornamen opslaan, en een tabel S waarin we de relatie tussen L en V bijhouden:

$$L = (\begin{pmatrix} \text{voornaam} \\ \text{man} \\ \text{vrouw} \end{pmatrix}, (\text{voornaam}), \emptyset, \emptyset) \tag{5.8}$$

$$S = (\begin{pmatrix} \text{voornamen} \\ \text{voornaam} \\ \text{positie} \end{pmatrix}, (\text{voornamen, voornaam}), \emptyset, \emptyset) \tag{5.9}$$

In L slaan we alle unieke losse voornamen op. Ook hier houden we bij of de losse voornaam alleen een jongensnaam, alleen een meisjesnaam of beide is. Dit baseren we direct op de waarden van *man* en *vrouw* van alle voornamen uit V waar de losse voornaam deel van uitmaakt. In S houden we de samenstelling van de voornamen bij en daarmee de relatie tussen tabellen V en L. Wanneer we een nieuwe voornaam uit V ontleden in losse voornamen en deze toevoegen aan L, maken we per losse voornaam een relatierecord tussen de voornaam in V en de voornamen in L, waarbij we opslaan welke plaats de losse voornaam in de gehele voornaam heeft. Daarnaast houden we in V bij uit hoeveel voornamen de gehele voornaam bestaat.

We zullen nu de losse voornamen gaan prematchen, net zoals we dat bij de achternamen hebben gedaan. Ook hier besluiten we alleen namen te vergelijken met dezelfde beginletter en maken daarbij dezelfde uitzonderingen (s met z, f met t, c met k en i met y). In tabel 5.9 op de rechter pagina staan de aantallen losse voornamen per letter en geslacht. De resultaten van het prematchen zullen we opslaan in de tabel LM. Deze hebben we al eerder gebruikt in paragraaf 4.2 op pagina 44 om de verwantschapsrelatie tussen losse voornamen op te slaan. We zullen de de tabel als volgt uitbreiden:

$$LM = (\begin{pmatrix} \text{lvn-paar} \\ \text{l} \\ \text{vp} \\ \text{vn} \\ \text{v} \\ \text{dl} \\ \text{d} \end{pmatrix}, (\text{lvn-paar}), \emptyset, \emptyset) \tag{5.10}$$

met

$$\text{lvn-paar} = \{\text{voornaam1}, \text{voornaam2}\} \tag{5.11}$$

De velden *vp*, *vn* en *v* worden gebruikt voor het verwantschapsalgoritme van hoofdstuk 4. Hier wordt in het Boolese veld *v* uiteindelijk bijgehouden of de voornamen wel of geen verwantschapsrelatie hebben. Bij het prematchen slaan we in *l* de Levenshteinafstand op en in *dl* de relatieve Levenshteinafstand. Aan de hand van formule 4.1 op pagina 43 kunnen we de uiteindelijke verschilwaarde uitrekenen tussen de twee namen en we slaan deze op in veld *d*. We moeten er echter rekening mee houden dat vele naamcombinaties die we niet prematchen wel in de tabel zullen komen als we het verwantschapsalgoritme gebruiken. Daarbij geven we de velden *l* en *dl* de waarde `null` en *d* de waarde 1. Tabel 5.10 op de pagina hiernaast laat zien hoeveel matches we hebben per verschilwaarde, als we het verwantschapsalgoritme buiten beschouwing laten. Net als bij de achternamen zien we hier een grillige toename in het aantal matches naarmate de verschilwaarde stijgt. Dezelfde effecten treden op.

eerste letter	aantal mannelijk	aantal vrouwelijk	totaal
a	2.567	3.721	5.830
b	1.943	2.119	3.728
c	1.387	1.786	2.962
d	1.161	1.419	2.390
e	1.717	2.665	3.996
f	1.253	1.817	2.824
g	2.364	2.898	4.907
h	2.708	3.091	5.233
i	636	860	1.336
j	1.880	2.465	3.946
k	771	1.060	1.646
l	1.813	2.389	3.818
m	1.803	3.005	4.450
n	602	753	1.249
o	487	505	887
p	897	1.044	1.801
q	55	29	79
r	1.925	2.000	3.542
s	1.875	2.463	3.948
t	1.435	2.148	3.240
u	239	239	426
v	361	437	748
w	1.713	1.911	3.324
x	31	10	39
y	80	90	148
z	254	362	579
overig	233	218	411
totaal	32.190	41.504	67.487

Tabel 5.9: Losse voornamen per voorletter en geslacht voor de vijf provincies

$d \in$	aantal
$[0{,}00; 0{,}05\rangle$	67.490
$[0{,}05; 0{,}10\rangle$	4.275
$[0{,}10; 0{,}15\rangle$	75.042
$[0{,}15; 0{,}20\rangle$	51.087
$[0{,}20; 0{,}25\rangle$	159.257
$[0{,}25; 0{,}30\rangle$	352.374
$[0{,}30; 0{,}35\rangle$	620.366
$[0{,}35; 0{,}40\rangle$	551.388
$[0{,}40; 0{,}45\rangle$	2.078.729
$[0{,}45; 0{,}50\rangle$	204.628
$\{0{,}50\}$	3.765.585

Tabel 5.10: Aantal paren losse voornamen per d-waarde voor alle vijf provincies.

5.4.2 Een heuristiek voor gehele voornamen

Nu we een tabel LM voorhanden hebben met relaties tussen losse voornamen en een tabel S met relaties tussen gehele en losse voornamen, kunnen we deze combineren. De functie $d^{\text{ovn}}(\alpha, \beta)$ geeft een verschilwaarde tussen twee gehele voornamen die afhangt van het aantal koppelingen tussen de losse voornamen. In paragraaf 3.3.4 op pagina 32 zagen we hoe de uiteindelijke verschilwaarde tussen twee voornamen alleen onder de 0,5 kan komen als minimaal de helft van de losse voornamen gekoppeld kan worden. We kunnen de heuristiek voor het prematchen van gehele voornamen hierop als volgt baseren. We prematchen twee gehele voornamen α en β dan en slechts dan als er tussen de losse voornamen van α en de losse voornamen van β evenveel of meer relatierecords bestaan in LM dan $\frac{(|\alpha|+|\beta|)}{2\cdot 2}$ en het aantal losse voornamen niet meer verschilt dan een factor 2. We delen een tweede keer door 2 omdat iedere match tussen losse voornamen potentieel twee losse voornamen verbindt. Overigens biedt deze heuristiek geen garantie dat d^{ovn} onder de 0,5 komt. Het kan immers zo zijn dat de meeste van de relaties in LM voorkomen tussen slechts enkele van de losse voornamen van beide namen. Maar andersom geldt wel dat als d^{ovn} onder de 0,5 komt, het namenpaar meegenomen wordt. We kunnen de heuristiek als volgt uitdrukken:

$$
\begin{aligned}
W \quad = \quad & (a \in V) \bowtie (b \in V) \\
& \Big\langle \tfrac{1}{2} \leq \tfrac{a(\text{aantallos})}{b(\text{aantallos})} \leq 2 \wedge \tfrac{a(\text{aantallos})+b(\text{aantallos})}{2\cdot 2} \leq \\
& \quad \Big| (s \in S)\langle s(\text{voornamen}) = a(\text{voornamen})\rangle \\
& \quad \bowtie (m \in LM)\langle m(\text{d}) \leq 0{,}5 \wedge s(\text{voornaam}) \in m(\text{lvn-paar})\rangle \\
& \quad \bowtie (t \in S)\langle t(\text{voornaam}) \in m(\text{lvn-paar}) \\
& \quad \wedge\, t(\text{voornamen}) = b(\text{voornamen})\rangle \Big| \quad \Big\rangle
\end{aligned}
\tag{5.12}
$$

Laten we daar een mogelijke implementatie aan toevoegen:

```
for each a ∈ V do
  T ← ∅
  for each s ∈ S : s(voornamen) = a(voornamen) do
    for each m ∈ LM : s(voornaam) ∈ m(lvn-paar) ∧ m(d) ≤ 0,5 do
      for each t ∈ S : t(voornaam) ∈ m(lvn-paar) do
        for each b ∈ V : b(voornamen) = t(voornamen)
             ∧ ((a(man) ∧ b(man)) ∨ (a(vrouw) ∧ b(vrouw))) do
          if 0,5 ≤ a(aantallos)/b(aantallos) ≤ 2 then
            if ∃(b, n) ∈ W then
              T ← T − (b, n) ∪ {(b, n + 1)}
            else
              T ← T ∪ {(b, 1)}
            fi
          fi
        od
      od
    od
  od
  for each (b, n) ∈ T do
    if n < (a(aantallos) + b(aantallos))/4 then T ← T − (b, n)
  od
  W ← {(a, b) : ∃(b, n) ∈ T}
od
```

Merk op dat we gebruik maken van een tijdelijke tabel T en dat we er reeds vanuit gaan dat we het gebruik van dit algoritme voorbereiden met het maken van relevante indices.

5.4.3 Problemen bij implementatie

Hoewel de heuristiek voor voornamen ons veel werk bespaart, is het nog steeds een grote klus. We kunnen vrij eenvoudig een ruwe schatting maken van de hoeveelheid werk. De drie for-loops a, s en m doen eigenlijk niets anders dan op hiërarchische wijze de gehele tabel LM doorlopen. Deze is na de verwerking van vijf provincies 8.021.151 records groot. Per gerelateerde losse voornaam uit m(lvn-paar) lopen we vervolgens alle samenstellingsrecords af. Daar $|S| = 822.469$ en $|L| = 67.487$, zijn er per losse voornaam gemiddeld ongeveer 12 samenstellingrecords en dus gehele voornamen. De uiteindelijke hoeveelheid uit te rekenen Levenshteinafstanden komt dan uit op $12 \times 8.021.151 = 96.253.812$. In de praktijk zal dit getal echter variëren aan de hand van de precieze relaties tussen individuele voornamen. Hoewel deze hoeveelheid werk op zich te doen is hebben we er in dit project vanaf gezien om het uit te voeren.

Om deze heuristiek toe te passen hebben we er aanvankelijk voor gekozen om dezelfde systematiek te gebruiken als we bij de totale heuristiek Q hebben gedaan. Hierbij programmeren we verzameling W als SQL-query en laten deze uitvoeren door de databaseserver. Als deze query uitgevoerd is, halen we de resultaten op uit de database en lopen we de voornamenparen in W langs met behulp van een cliëntapplicatie. Hiermee rekenen voor ieder paar d^{vn} uit. Als deze waarde onder de 0,5 uitkomt, slaan we het paar op in een tabel voor matches tussen gehele voornamen VM. Deze kunnen we tenslotte gebruiken in in de heuristiek Q.

In de praktijk is echter gebleken dat dit proces te lang duurt. Met deze opzet zou het prematchen van de voornamen meerdere weken in beslag hebben genomen. We hebben ervoor gekozen om niet over te stappen op het voor de hand liggende alternatief, alle gegevens uit de database kopiëren naar het geheugen van de clientapplicatie en daar het werk doen. In plaats daarvan hebben we de vergelijking op voornaam uit heuristiek Q verwijderd en de gereduceerde heuristiek getest. Omdat die qua effectiviteit volstond hebben we hem als zodanig gehandhaafd en gebruikt.

5.4.4 Een gereduceerde heuristiek

De heuristiek R die we uiteindelijk hebben toegepast vergelijkt de persoonsknopen van echtparen alleen op achternaam, naast geboortejaartal en geslacht. We voeren de vergelijking op achternaam uit door gebruik te maken van de koppeltabel AM, waarin combinaties achternamen staan die een redelijk goede d^{an}-waarde hebben.

De uiteindelijke specificatie van R is als volgt:

$$
\begin{aligned}
R \;=\; & \sigma_{(g,h)}\Big((g \in G)\langle g(\text{rol}) \in \{\text{oudersbruidegom}, \text{oudersbruid}\}\rangle \\
& \bowtie (a \in P)\langle \text{vrouwgezin}(a, g)\rangle \\
& \bowtie (b \in P)\langle \text{mangezin}(b, g)\rangle \\
& \bowtie (m \in AM)\langle a(\text{achternaam}) \in m(\text{an-paar})\rangle \\
& \bowtie (c \in P)\big\langle (c(\text{achternaam}) \in m(\text{an-paar}) \\
& \qquad \wedge\, a(\text{gb-min}) \leq c(\text{gb-max}) \\
& \qquad \wedge\, a(\text{gb-max}) \geq c(\text{gb-min}) \\
& \qquad \wedge\, c(\text{rol}) = \text{bruid}\big\rangle \\
& \bowtie (h \in G)\langle \text{vrouwgezin}(c, h)\rangle \\
& \bowtie (d \in P)\big\langle b(\text{gb-min}) \leq d(\text{gb-max}) \\
& \qquad \wedge\, b(\text{gb-max}) \geq d(\text{gb-min}) \\
& \qquad \wedge\, \text{mangezin}(d, h)\big\rangle \\
& \bowtie (n \in AM)\langle \{b(\text{achternaam}), d(\text{achternaam})\} = n(\text{an-paar})\rangle\Big)
\end{aligned}
\tag{5.13}
$$

Ook hier zullen we een inschatting maken van de hoeveelheid werk die het uitvoeren van deze heuristiek oplevert. Als we aannemen dat iedere soort zoekvraag die de database moet uitvoeren voorbereid is met een index, komen we op het volgende uit. Voor g moet langs $\frac{2}{3}|G|$ records geïtereerd worden. Records a en b worden hier uniek door bepaald en leveren dus maar 1 iteratie op. Hoeveel records m wordt gevonden zal verschillen per achternaam. Na de verwerking van vijf provincies zijn er 196.107 verschillende achternamen en 16.627.796 matches ertussen met een d-waarde van 0,5 of minder. Dat zijn gemiddeld ongeveer 85 gerelateerde achternamen per achternaam. Het gemiddelde aantal records c dat deze achternaam en de rol *bruid* heeft is $1/6 \times \frac{|P|}{|A|}$. Aan de hand van de bruid c kunnen gezin h en bruidegom d meteen gevonden worden. Aan de hand van de achternamen van b en d kunnen we weer direct via een index op LM zien of er een match bestaat voor hun achternamen. Zo houden we h, d, en n op 1 iteratie. We laten hier de eliminatie van de bruidsrecords c op basis van geboortejaar buiten beschouwing. Laten we allereerst van de provincie Zeeland uitgaan. Hiervoor geldt:

$$
\begin{aligned}
|P| &= 986.714 \\
|G| &= 493.452 \\
|A| &= 43.328 \\
\frac{|AM|}{|A|} &\approx 85
\end{aligned}
$$

Met deze cijfers komen we uit op:

$$
\begin{aligned}
\text{aantal iteraties} &= 2/3|G| \times \frac{|AM|}{|A|} \times 1/6\frac{|P|}{|A|} \\
&= \frac{|G||AM||P|}{9|A|^2} \\
&= \frac{85|G||P|}{9|A|} \\
&= \frac{85 \times 493.452 \times 986.714}{9 \times 43.328} \\
&\approx 106.131.420
\end{aligned}
$$

Dit getal is hoog, maar niet te hoog, in tegenstelling tot bij het prematchen van de voornamen. Al dit werk kan namelijk door de database worden uitgevoerd en de uiteindelijke resultatenset die aan de cliëntapplicatie gegeven wordt voor verwerking is

veel kleiner. Bij Zeeland bestond deze set uit 315.384 kandidaatmatches. De database klaart deze klus in een uur of acht, waarna de cliëntapplicatie de daadwerkelijke matches nog moet maken. Dit duurt ook nog een paar uur. Deze precieze tijden zijn tijdens het project niet bijgehouden en zijn afhankelijk van de implementatie, de hardware en andere factoren.

5.5 Het verwijderen van dubbele matches

Na het matchingsproces hebben we matches tussen ouderparen en bruidsparen. Vaak komt het voor dat matches elkaar uitsluiten. Zo nemen we aan dat wanneer één ouderpaar matcht met twee bruidsparen op verschillende akten, dit meestal niet klopt. Vaak is één van de twee matches beter dan de andere. We willen dan alleen de goede match bewaren en de andere elimineren. Ook komt het voor dat we meerdere goede matches hebben. Dit kan verschillende oorzaken hebben. Mogelijkerwijs is één van de akten geen huwelijksakte, maar bijvoorbeeld een echtscheidingsakte (zie paragraaf 1.1.3 op pagina 11). Soms lijken twee echtparen daadwerkelijk erg op elkaar. Om dergelijke conflicten te behandelen hanteren we de volgende logica:

$$x = a \wedge x = b \quad \longrightarrow \quad a = b \tag{5.14}$$

We beschouwen een ouderpaar x. Deze matcht met bruidspaar a en met bruidspaar b. Als deze matches kloppen moeten a en b ook naar dezelfde personen refereren. Als dit niet het geval is, dan is x niet a of niet b:

$$a \neq b \quad \longrightarrow \quad x \neq a \vee x \neq b \tag{5.15}$$

Deze logica kunnen we doorvoeren op de matchrelaties. Als we een match tussen x en a hebben, en één tussen x en b, proberen we een match te maken tussen a en b. Als deze goed genoeg is, handhaven we alle drie de matches. Zo niet, dan bewaren we alleen de beste match, als het verschil niet te klein is. Als het verschil te klein is elimineren we beide matches. Dit levert twee parameters op waarover we na moeten denken. De eerste is de d-waarde waaronder we de controlematch tussen a en b goed genoeg vinden. De tweede is de mate van verschil tussen de matches waarboven we één ten gunste van de andere elimineren. We noemen deze parameters hier m (maximum) en v (verschil). Met het volgende algoritme bekijken we alle dubbele matches nader:

```
D ← {(a,{x,y}) : match (a,x) bestaat en match (a,y) bestaat
  en match (x,y) bestaat niet}
for each (a,{x,y}) ∈ D do
  if d(a,x) ≤ m ∧ d(a,y) ≤ m then
    bereken d(x,y)
    if d(x,y) < m then
      sla match (x,y) op in database
    else
      elimineer match (x,y)
      if d(a,x) + v < d(a,y) then
        elimineer match (a,y)
      elseif d(a,y) + v < d(a,x) then
        elimineer match (a,x)
      fi
    fi
  else
    if d(a,x) > m then elimineer match (a,x)
```

```
    if d(a, y) > m then elimineer match (a, y)
  fi
od
```

Proefondervindelijk hebben we bepaald dat v op 0,05 en m op 0,3 acceptabele resultaten oplevert.

Naast de dubbele matches die we hierboven bespraken, een ouderpaar dat met meerdere bruidsparen matcht, hebben we natuurlijk ook bruidsparen die met meerdere ouderparen matchen. Dit vormt geen probleem. Vaak komen echtelieden eerst voor op hun huwelijksakte, om daarna terug te komen als ouders op de huwelijksakten van hun kinderen. Bovenstaand algoritme zal deze dubbelen in de meeste gevallen niet elimineren, maar wel matches toevoegen tussen de ouderparen onderling en zo de set van matches van type A verrijken met matches van type B.

Merk op dat het algoritme meestal onderscheid zal maken tussen beide soorten dubbelen op basis van geboortejaar. Bij de eerste soort dubbelen zal het algoritme de matches (a, x) en (a, y) controleren door de match (x, y) te proberen. Omdat x en y in dit geval beide bruidsparen zijn en doorgaans van de bruidsparen het geboortejaar bekend is, zal de match falen als deze niet overeenkomen. Dit zal meestal het geval zijn. Bij de tweede soort dubbelen stellen x en y ouderparen voor. De grootste precisie waarmee we hun geboortejaren kennen is doorgaans 38 jaar bij de moederknoop. De kans op overlap is dus veel groter en de controlematch zal veel vaker het stadium van naamsvergelijking bereiken.

Hoofdstuk 6

Resultaten

Tijdens dit project hebben we de huwelijksakten van de provincies Groningen, Overijssel, Gelderland, Zeeland en Limburg gematcht. Het gaat natuurlijk te ver om iedere match in dit document op te nemen, maar we kunnen wel kijken naar de kwantitatieve resultaten en die zo goed mogelijk kwalitatief analyseren.

6.1 Aantallen matches

We kunnen het grote aantal matches op vele manieren bekijken. We beginnen met de simpelste en voor de hand liggendste: we bekijken het aantal matches per d^{M}-waarde. In tabel 6.1 op de volgende pagina staat een overzicht van aantallen matches per provincie. In de kolom *maximum* staat het aantal huwelijksakten van de provincie maal twee. Dat is een zwakke bovengrens voor het aantal matches. Iedere akte biedt immers twee echtparen aan, de ouders, die gematcht kunnen worden met een huwend paar op een andere akte. Deze bovengrens wordt normaal nooit gehaald. Er zijn altijd ouders in de collectie die zelf niet in dezelfde provincie zijn getrouwd. Ook kunnen de akten van minimaal de eerste 15 jaar niet gematcht worden. Overigens kunnen, zeker als we hogere d^{M}-waarden toestaan, ouderparen met meer dan één huwend paar gematcht worden. Zulke dubbelen kunnen in theorie het aantal gevonden matches boven het maximum uit laten stijgen. Zo zal bij een d^{M}-waarde van 1,0 of lager alles met alles kunnen matchen en deze bovengrens niet van toepassing zijn. We laten matches slechter dan 0,5 echter weg en komen niet dichtbij de bovengrens. Onder ieder aantal staat een percentage dat de verhouding aangeeft tussen het aantal matches en het maximum voor de provincie.

De kolom $d^{\mathrm{M}} = 0$ geeft de exacte matches aan. Bij deze matches zijn de namen identiek. Ook komen natuurlijk alle jaartallen overeen. Het lage percentage van Limburg valt op. Dit zou verklaard kunnen worden door een relatief groot personenverkeer met de omliggende streken en door inconsistenter gebruik van verwante voornamen. In de andere kolommen staat het aantal matches dat verkregen wordt door te matchen zonder het verwantschapsalgoritme. We zien dat het aantal matches het sterkst toeneemt in het begin. Er worden grote sprongen gemaakt van $d^{\mathrm{M}} = 0$ naar $d^{\mathrm{M}} \leq 0{,}1$ en van $d^{\mathrm{M}} \leq 0{,}1$ naar $d^{\mathrm{M}} \leq 0{,}2$. Daarna worden de sprongen kleiner.

Tabel 6.2 op de pagina hierna toont dezelfde cijfers voor het matchen met behulp van het verwantschapsalgoritme. We zien dat de aantallen op dezelfde manier met d^{M}toenemen. Vooral wanneer we ons beperken tot lage d^{M}-waarden hebben we iets grotere aantallen matches. Dit is het best zichtbaar bij Limburg, waar we bij $d^{\mathrm{M}} \leq 0{,}1$ van 104.396 naar 128.047 matches gaan. Hier is de relatieve toename het grootst, 23%. We moeten echter concluderen dat de achterstand van Limburg op

provincie	maximum	$d^{\mathrm{M}} = 0$	$d^{\mathrm{M}} \leq 0{,}1$	$d^{\mathrm{M}} \leq 0{,}2$	$d^{\mathrm{M}} \leq 0{,}3$	$d^{\mathrm{M}} \leq 0{,}4$	$d^{\mathrm{M}} \leq 0{,}5$
Groningen	416.446	144.232	206.675	248.743	258.653	264.658	269.785
		35%	50%	60%	62%	64%	65%
Overijssel	439.046	169.296	218.161	244.497	249.934	255.986	258.424
		39%	50%	56%	57%	58%	59%
Gelderland	732.090	214.873	307.698	349.806	358.131	368.307	372.172
		29%	42%	48%	49%	50%	51%
Zeeland	328.996	112.598	165.900	191.500	196.750	200.944	202.003
		34%	50%	58%	60%	61%	61%
Limburg	379.358	63.778	104.396	154.585	179.234	192.424	198.253
		17%	28%	41%	47%	51%	52%

Tabel 6.1: Matches zonder het verwantschapsalgoritme

provincie	maximaal	$d^{\mathrm{M}} = 0$	$d^{\mathrm{M}} \leq 0{,}1$	$d^{\mathrm{M}} \leq 0{,}2$	$d^{\mathrm{M}} \leq 0{,}3$	$d^{\mathrm{M}} \leq 0{,}4$	$d^{\mathrm{M}} \leq 0{,}5$
Groningen	416.446	144.232	220.129	257.754	268.317	274.554	278.697
		35%	53%	62%	64%	66%	67%
Overijssel	439.046	169.296	232.077	253.055	257.883	260.091	260.654
		39%	53%	58%	59%	59%	59%
Gelderland	732.090	214.873	334.624	365.728	371.586	373.812	374.340
		29%	46%	50%	51%	51%	51%
Zeeland	328.996	112.598	177.684	198.378	201.508	202.059	202.171
		34%	54%	60%	61%	61%	61%
Limburg	379.358	63.778	128.047	176.407	191.938	198.058	199.753
		17%	34%	47%	51%	52%	53%

Tabel 6.2: Matches met het verwantschapsalgoritme

ouderpaar	huwend paar				
	Groningen	Overijssel	Gelderland	Zeeland	Limburg
Groningen	144.232	1.965	1.518	252	215
Overijssel	950	169.296	8.409	251	270
Gelderland	663	11.518	214.873	457	1261
Zeeland	237	360	918	112.598	220
Limburg	59	156	888	111	63.778

Tabel 6.3: Matches tussen provincies met $d^{\mathrm{M}} = 0$

ouderpaar	huwend paar				
	Groningen	Overijssel	Gelderland	Zeeland	Limburg
Groningen	206.675	2.535	2.102	370	326
Overijssel	1.364	218.161	11.871	349	413
Gelderland	942	16.334	307.698	710	2.022
Zeeland	347	538	1.408	165.900	373
Limburg	98	242	1.577	169	104.396

Tabel 6.4: Matches tussen provincies met $d^{\mathrm{M}} \leq 0{,}1$

de andere provincies nog niet is ingelopen.

6.1.1 Matches tussen provincies

Tabellen 6.1 en 6.2 tellen alleen matches tussen akten uit dezelfde provincie. Twee doorsneden van de matches tussen akten uit verschillende provincies zijn weergegeven in tabellen 6.3 en tabel 6.4. Deze tonen aantallen matches met $d^{\mathrm{M}} = 0$ en $d^{\mathrm{M}} \leq 0{,}1$ respectievelijk. We zien vooral veel matches tussen de enige twee provincies die aan elkaar grenzen, Overijssel en Gelderland.

6.2 Kwalitatief oordeel

Idealiter zouden we de historische onderzoekers die gebruik maken van de uitkomsten van dit project alleen kloppende matches willen aanbieden. Dat is echter onmogelijk. Ook tussen de matches met $d^{\mathrm{M}} = 0$ vinden we nog onterechte matches. Om precies te bepalen welke dat zijn moeten we ons tot andere bronnen wenden dan de huwelijksakten, maar we kunnen aan conflicten tussen dubbele matches zien dat ze bestaan.

Als voorbeeld nemen we akte nummer 6 uit Aduard, Groningen in 1915. Als ouders van de bruid worden hier vermeld Enne Blaauw (geboortejaarbereik [1782–1867]) en Doetje van der Nald (geboortejaarbereik [1829–1867]). Wanneer we op zoek gaan naar hun eigen huwelijksakte, vinden we twee kandidaten. Enerzijds vinden we akte nummer 10 uit dezelfde gemeente in 1880. De namen komen perfect overeen, al zijn er geen leeftijden of geboortejaren opgegeven. Dat een dochter van Enne en Doetje 35 jaar na het huwelijk van haar ouders trouwt is goed denkbaar. Anderzijds treffen we een bruidspaar aan met precies dezelfde namen op een akte uit Grootegast, 1870, nummer 3. Hierop staan wel geboortedata: 1821 voor Enne en 1839 voor Doetje. Ook deze data kunnen goed overeenstemmen met de eerste akte. Hoewel voor beide matches een d^{M}-waarde van 0 geldt, is toch minimaal één van de matches fout.

Om een goede d^{M}-grenswaarde te bepalen hebben we een testset nodig: een

verzameling huwelijksakten waarvan alle kloppende matches bekend zijn. We kunnen ons matchproces dan loslaten op deze collectie en onze eigen matches maken. Door die te vergelijken met de van te voren bekende goede matches kunnen we de d^{M}-grenswaarde bepalen waaronder onze matches het meest met de testset overeenkomen. Daarnaast kunnen we goede waarden bepalen voor de vele gewichten en parameters die de d^{M}-functie rijk is. We hebben echter geen beschikking over een dergelijke ideale testset. Het construeren van een set die groot genoeg is voor dit doel is een zeer tijdrovende en moeilijke bezigheid. We zijn daarom op een andere methode aangewezen om een grenswaarde te bepalen.

6.2.1 Dubbele matches

Om een oordeel te vellen over de grenswaarden kunnen we kijken naar het relatieve aantal conflicterende dubbele matches. Wanneer we 0 als bovengrens voor d^{M} hanteren is de verhouding van het aantal dubbelen tot het totale aantal matches laag. Bij $d^{\mathrm{M}} \leq 1$ zullen alle matches dubbelen zijn. We kunnen het relatieve aantal dubbelen bij de tussenliggende grenswaarden bekijken en een oordeel vormen aan de hand van hoe deze aantallen toenemen.

In de ideale situatie levert het verhogen van de grenswaarde alleen maar nieuwe goede matches op en geen foute matches. In dat geval zouden we überhaupt geen dubbelen verwachten. Namen zijn echter niet uniek en omdat we bij $d^{\mathrm{M}} = 0$ al dubbelen zien, moeten we ook bij de verruiming van perfecte matchcriteria extra dubbelen zien ontstaan in dezelfde verhouding. In werkelijkheid levert het verruimen van criteria natuurlijk ook slechte matches op, die een grote kans hebben om met een andere match te conflicteren. We kunnen de mate waarin dit gebeurt gebruiken om de grenswaarden te beoordelen.

Tabel 6.5 op de rechter pagina toont het aandeel dubbele matches voor verschillende d^{M}-waarden. We nemen de aantallen bij $d^{\mathrm{M}} = 0$ als uitgangspunt. Bij $d^{\mathrm{M}} = 0{,}1$ is het aantal dubbelen rond de twee keer zo groot. Bij $d^{\mathrm{M}} = 0{,}2$ zien we een grote sprong. Bij een aantal provincies wordt het aantal dubbelen tien keer zo groot. Daarna blijft het aantal dubbelen stijgen. In combinatie met de cijfers van tabel 6.1 op pagina 68 komt $d^{\mathrm{M}} \leq 0{,}1$ goed uit de verf: bij deze grenswaarde behalen we een groot deel van de winst in het aantal extra matches terwijl we een relatief kleine toename in dubbelen zien.

Bij het gebruik van het verwantschaps algoritme, tabel 6.6 op de rechter pagina, zien we heel andere aantallen dubbelen. De aantallen zijn veel hoger, soms een factor 10, dan wanneer het verwantschapsalgoritme niet wordt gebruikt. Alleen bij Limburg is deze stijging kleiner. Dit kan verschillende oorzaken hebben. Om te beginnen kunnen de cijfers erop duiden dat de naamsverwantschapslijsten die door het algoritme worden gebruikt niet zuiver genoeg zijn. Ten tweede kan de waarde 0,1 die we gebruiken voor d_{v} (de verschilwaarde die we aan verwante namen toekennen) te laag zijn. Eventueel leidt het gebruik van verwante namen sowieso tot meer dubbelen. Tenslotte is er de mogelijkheid dat de getallen beïnvloed zijn door de volgorde waarin de provincies zijn gematcht. De eerste provincie waarop het verwantschapalgoritme is gebruikt is Limburg. Daarna is naamverwantschapsinformatie van Limburg uitgebreid met die van Zeeland en zijn de aktes van Zeeland met behulp van de totale lijst gematcht. Zo zijn daarna de matches van Gelderland, Overijssel en Groningen met een steeds uitgebreidere verwantschapslijst behandeld. We hebben voor deze aanpak gekozen omdat we verwachtten dat de kwaliteit van de lijst zou toenemen, met name door de snelle toename van negatieve verwantschapsinformatie. De cijfers van tabel 6.6 zouden erop kunnen duiden dat de lijst, naarmate deze de informatie van meer provincies bevatte, in kwaliteit achteruit ging. In de volgorde waarin de lijst groeide, zien we immers het aantal dubbelen ook toenemen. In dat geval zouden nieuwe matchrondes gedraaid moeten worden

provincie	$d^{\mathrm{M}} = 0$	$d^{\mathrm{M}} \leq 0{,}1$	$d^{\mathrm{M}} \leq 0{,}2$	$d^{\mathrm{M}} \leq 0{,}3$	$d^{\mathrm{M}} \leq 0{,}4$	$d^{\mathrm{M}} \leq 0{,}5$
Groningen	0,058%	0,061%	0,211%	0,822%	2,034%	3,542%
Overijssel	0,024%	0,047%	0,262%	0,561%	1,484%	2,021%
Gelderland	0,022%	0,052%	0,262%	0,587%	1,307%	1,729%
Zeeland	0,163%	0,266%	0,441%	0,573%	0,864%	0,970%
Limburg	0,039%	0,078%	0,353%	1,325%	2,971%	4,035%

Tabel 6.5: Het aantal dubbele matches ten opzichte van het totale aantal matches zonder het verwantschapsalgoritme

provincie	$d^{\mathrm{M}} = 0$	$d^{\mathrm{M}} \leq 0{,}1$	$d^{\mathrm{M}} \leq 0{,}2$	$d^{\mathrm{M}} \leq 0{,}3$	$d^{\mathrm{M}} \leq 0{,}4$	$d^{\mathrm{M}} \leq 0{,}5$
Groningen	0,058%	0,666%	1,433%	3,228%	5,009%	6,248%
Overijssel	0,024%	0,487%	1,066%	1,855%	2,523%	2,693%
Gelderland	0,022%	0,482%	0,958%	1,595%	2,054%	2,156%
Zeeland	0,163%	0,475%	0,694%	0,829%	0,968%	1,010%
Limburg	0,039%	0,144%	0,565%	1,910%	3,869%	4,500%

Tabel 6.6: Het aantal dubbele matches ten opzichte van het totale aantal matches met het verwantschapsalgoritme

met nieuwe verwachtschapsinformatie in de hoop dat het relatieve aantal dubbelen dichter in de buurt komt van dat van Limburg.

Conclusie

In dit project hebben we ons bezig gehouden met het matchen van huwelijksakten van Genlias. Zo verkregen we longitudinale informatie voor historisch onderzoek. Het matchen zelf is slechts een van vele werkzaamheden die aan bod zijn gekomen. Zo heb ik databestanden opgeschoond en genormaliseerd die in uiteenlopende vormen waren aangeleverd. Ik heb een methodiek opgebouwd om de genealogische verbanden op uniforme wijze te representeren, om informatie van huwelijksakten, geboorteakten en andere bronnen door elkaar te kunnen gebruiken. Verder heb ik een algoritme geschreven voor het matchen van deze genormaliseerde genealogieën. Ook heb ik te maken gehad met het optimaliseren van databaseprocessen en client-databasecommunicatie. Daarnaast heb ik een visuele interface gebouwd voor het besturen van deze processen. Tenslotte heb ik de resultaten van de verschillende matchronden moeten analyseren. Dit alles maakt het project bijzonder multidisciplinair.

Ook al waren ze tijdrovend, bepaalde aspecten van het project, zoals de datainvoer en de visuele client, waren minder geschikt om in detail op te nemen in deze scriptie. Appendix B biedt nog een blik op de clientapplicatie, Certilink.

Hoewel de veelzijdigheid het project bijzonder interessant maakte, werd het daardoor ook moeilijk om diep op alle aspecten van het project in te gaan die dit verdienden. Toch hoop ik een goede balans tussen verschillende aandachtsgebieden te hebben gevonden.

Dankzij de uniforme genealogiestructuur, die gedurende het project alleen gebruikt is voor huwelijksakten, zijn inmiddels matches tussen huwelijksakten en geboorteakten gemaakt, waardoor de databestanden van de HSN gekoppeld zijn met die van Genlias.

Het naamverwantschapsalgoritme heeft, door automatisch verwante voornamen te herkennen, het aantal gevonden matches van in ieder geval Limburg flink vergroot. Hoewel het ook tot extra matches heeft geleid bij andere provincies, moet nader worden geanalyseerd wat de kwaliteit van deze matches is. We hebben immers gezien dat er veel conflicterende dubbelen ontstaan, wat een indicatie kan zijn voor slechte matches.

De software die ik heb geschreven om de matchronden uit te voeren maakt gebruik van een database, zowel voor de opslag van de grote hoeveelheden data als voor het doorzoeken van deze hoeveelheden tijdens het matchproces. Het was nodig om deze zoekprocessen zo te optimaliseren dat de matchronden binnen redelijke tijd uit te voeren waren. Voor vervolgonderzoek beveel ik aan het zoekwerk zoveel mogelijk in het programmageheugen uit te voeren en communicatie met een database te beperken tot het ophalen en wegschrijven aan het begin en einde van het proces. Desondanks is het mogelijk om met de huidige software binnen redelijke tijd experimenten uit te voeren.

Tot slot is het van grote waarde om de in hoofdstuk 6 genoemde testset aan te leggen of te construeren uit bestaand werk, zoals de HSN-collectie. Met een dergelijke set kan de verschilfunctie afgesteld worden om de verzameling van goede matches preciezer te identificeren.

Zelfs met een dergelijke set blijft een fundamentele onzekerheid bestaan over welke matches wel en welke niet kloppen. Maar het is juist die onzekerheid die het werken met historische data zo interessant en waardevol maakt.

Bijlage A

Matches

In deze scriptie hebben we ons met name bezig gehouden met de methodologie van het matchen en niet met de akten of de matches zelf. Om toch wat van het materiaal zelf te laten zien toont deze appendix een aantal willekeurig gekozen matches. Op de volgende pagina's staan tien matches uit de categorie $d^{\mathrm{M}} \in [0{,}0; 0{,}1\rangle$ en uit de categorieën $[0{,}1; 0{,}2\rangle$, $[0{,}2; 0{,}3\rangle$, $[0{,}3; 0{,}4\rangle$ en $[0{,}4; 0{,}5\rangle$. De matches zijn gesorteerd op afnemende kwaliteit.

Per match is veel informatie over de brondata en de matchoverwegingen beschikbaar. We beschouwen hier alleen een selectie van de gegevens. Per match zien we de volgende informatie:

- de akte waarop de gematchte personen als ouders staan:
  - de rol van de ouders op de akte
  - aktenummer, akteplaats en aktejaar
  - de namen van de ouders
  - de geboortejaarbereiken van de ouders
- de akte waarop de gematchte personen als huwenden staan:
  - aktenummer, akteplaats en aktejaar
  - de namen van de huwenden
  - de geboortejaarbereiken van de huwenden
- voor beide persoonsmatches een aantal matchwaarden:
  - d^{gvn}: de verschilmaat tussen de gematchte voornamen
  - d^{vvn}: de mate van verandering in volgorde van gematchte voornamen
  - d^{ovn}: de mate waarin de voornamen onderling ongematcht bleven
  - d^{vn}: de totale verschilmaat voor de voornamen
  - d^{vn}: de verschilmaat voor de achternamen
- de verschilmaat voor de gehele match: d^{M}

Matches met $d^{\mathrm{M}} \in [0{,}0; 0{,}1\rangle$:

ouders van bruid van akte 22, Oude Pekela, 1899

Folmer Kamminga × Mijntje Bos
geboren: 1777 – 1862 geboren: 1824 – 1862

huwenden van akte 10, Meeden, 1863

Folmer Kamminga × Mijntje Bos
geboren: 1837 geboren: 1833

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$
$d^{\mathrm{M}} = 0{,}00$

---

ouders van bruid van akte 12, Enschede, 1868

Jan Heutink × Willemina Heuw
geboren: 1734 – 1819 geboren: 1781 – 1819

huwenden van akte 27, Lonneker, 1832

Jan Heutink × Willemina ten Heuw
geboren: 1809 geboren: 1806

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$
$d^{\mathrm{M}} = 0{,}00$

---

ouders van bruidegom van akte 134, Ede, 1917

Cornelis van den Dikkenberg × Dirkje Hendrijntje van de Voort
geboren: 1797 – 1882 geboren: 1844 – 1882

huwenden van akte 4, Ede, 1895

Cornelis van den Dikkenberg × Dirkje Hendrijntje van de Voort
geboren: 1864 geboren: 1869

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$
$d^{\mathrm{M}} = 0{,}00$

---

ouders van bruidegom van akte 71, Voorst, 1916

Cornelis Nijkamp × Louiza Gerrigje Thijssen
geboren: 1792 – 1877 geboren: 1839 – 1877

huwenden van akte 42, Epe, 1887

Cornelis Nijkamp × Louiza Gerrigje Thijssen
geboren: 1855 geboren: 1855

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$
$d^{\mathrm{M}} = 0{,}00$

ouders van bruidegom van akte 8, Tubbergen, 1922

Gerhardus Lambertus Wolters × Aleida Hommels
geboren: 1794 – 1879 geboren: 1841 – 1879

huwenden van akte 21, Tubbergen, 1887

Gerhardus Lambertus Wolters × Aleida Hommels
geboren: 1861 geboren: 1857

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{M}} = 0{,}00$

ouders van bruid van akte 2, Hattem, 1908

Hendrik van Welsum × Arendje Veldkamp
geboren: 1786 – 1871 geboren: 1833 – 1871

huwenden van akte 27, Oldebroek, 1883

Hendrik van Welsum × Arendje Veldkamp
geboren: 1859 geboren: 1863

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{M}} = 0{,}00$

ouders van bruid van akte 8, Sint Jansteen, 1861

Antonius Peeters × Paulina van Remoortere
geboren: 1737 – 1822 geboren: 1784 – 1822

huwenden van akte 3, Sint Jansteen, 1833

Antonius Peeters × Paulina van Remoortere
geboren: 1801 geboren: 1806

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{M}} = 0{,}00$

ouders van bruid van akte 6, Herwijnen, 1848

Hendricus van Arendonk × Jacoba Adriana Vervoorn
geboren: 1729 – 1814 geboren: 1776 – 1814

huwenden van akte 6, Herwijnen, 1828

Hendrikus van Arendonk × Jacoba Adriana Vervoorn
geboren: 1799 geboren: 1804

$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}10$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{M}} = 0{,}05$

ouders van bruidegom van akte 25, Bellingwolde, 1859

Willem Harms Kruize × Anna Jakobs Tipkers
geboren: 1736 – 1821 geboren: 1783 – 1821

huwenden van akte 2, Wedde, 1836

Willem Harms Kruize × Anna Jacobs Tipker
geboren: 1807 geboren: 1812

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{gvn}} = 0{,}07$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}07$, $d^{\mathrm{ln}} = 0{,}14$

$d^{\mathrm{M}} = 0{,}08$

ouders van bruidegom van akte 12, Dinxperlo, 1843

Ruth Vlaswinkel	×	Christina Essink
geboren: 1718 – 1803		geboren: 1765 – 1803

huwenden van akte 7, Dinxperlo, 1815

Rutger Vlaswinkel	×	Christina Essing
geboren: 1794		geboren: 1792

$d^{\text{gvn}} = 0{,}10$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{vn}} = 0{,}10$, $d^{\text{ln}} = 0{,}00$

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}17$

$d^{\text{M}} = 0{,}10$

Matches met $d^{\mathrm{M}} \in [0{,}1; 0{,}2\rangle$:

ouders van bruidegom van akte 44, Rheden, 1902

Gerrit Jan Meijer	×	Gerritje van den Beldt
geboren: 1780 – 1865		geboren: 1827 – 1865

huwenden van akte 91, Arnhem, 1878

Gerrit Jan Meijer	×	Gerritje van de Beld
geboren: 1831		geboren: 1839

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$	$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$	$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}20$

$d^{\mathrm{M}} = 0{,}10$

---

ouders van bruidegom van akte 129, Middelburg, 1917

Jan van Dien	×	Katelina Heijboer
geboren: 1794 – 1879		geboren: 1841 – 1879

huwenden van akte 10, Wissenkerke (NB), 1884

Jan van Dien	×	Kathalina Heijboer
geboren: 1854		geboren: 1861

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$	$d^{\mathrm{gvn}} = 0{,}22$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$	$d^{\mathrm{vn}} = 0{,}22$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{M}} = 0{,}11$

---

ouders van bruidegom van akte 32, Steenwijkerwold, 1906

Jan de Jonge	×	Klaasje de Jong
geboren: 1783 – 1868		geboren: 1830 – 1868

huwenden van akte 3, Avereest, 1877

Jan de Jonge	×	Jantje de Jonge
geboren: 1855		geboren: 1858

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$	$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$	$d^{\mathrm{vn}} = 0{,}10$, $d^{\mathrm{ln}} = 0{,}20$

$d^{\mathrm{M}} = 0{,}11$

---

ouders van bruid van akte 141, Arnhem, 1891

Arent Kruk	×	Katrientje Rijnders
geboren: 1767 – 1852		geboren: 1814 – 1852

huwenden van akte 45, Rheden, 1864

Arent Kruk	×	Katrientje Reinders
geboren: 1832		geboren: 1833

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$	$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$	$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}25$

$d^{\mathrm{M}} = 0{,}13$

ouders van bruidegom van akte 9, Brummen, 1882

Jan Berends × Johanna Ambrosius
geboren: 1756 – 1841 geboren: 1803 – 1841

huwenden van akte 2, Heteren, 1851

Jan Berns × Johanna Ambrosius
geboren: 1825 geboren: 1826

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}29$ $d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}00$
$d^{\text{M}} = 0{,}14$

ouders van bruid van akte 3, Oude Pekela, 1860

Harm Strobos × Janna Wessels Dorgeloos
geboren: 1735 – 1820 geboren: 1782 – 1820

huwenden van akte 12, Wedde, 1818

Harm Hindriks Strobos × Janna Wessels Dorgeloos
geboren: 1791 geboren: 1799

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}00$ $d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}00$
$d^{\text{M}} = 0{,}17$

ouders van bruid van akte 10, Rossum, 1896

Gerardus van Heel × Clasina van Os
geboren: 1759 – 1844 geboren: 1806 – 1844

huwenden van akte 7, Driel, 1843

Gerardus van Heel × Clasina van Oss
geboren: 1810 geboren: 1819

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}00$ $d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}33$
$d^{\text{M}} = 0{,}17$

ouders van bruidegom van akte 14, Delfzijl, 1917

Casper Rodenhuis × Hillechien Frieling
geboren: 1792 – 1877 geboren: 1839 – 1877

huwenden van akte 41, Appingedam, 1890

Casper Adam Rodenhuis × Hillegien Frieling
geboren: 1857 geboren: 1864

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$ $d^{\text{gvn}} = 0{,}10$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}00$ $d^{\text{vn}} = 0{,}10$, $d^{\text{ln}} = 0{,}00$
$d^{\text{M}} = 0{,}17$

ouders van bruidegom van akte 3, Middelstum, 1858

Cornelis Jans Dijkema × Geertje Wiggers Rijpema
geboren: 1731 – 1816 geboren: 1778 – 1816

huwenden van akte 9, Stedum, 1822

Kornellis Jans Dijkema × Geertje Wichers Rijpinga
geboren: 1797 geboren: 1800

$d^{\text{gvn}} = 0{,}07$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{gvn}} = 0{,}07$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}07$, $d^{\text{ln}} = 0{,}00$ $d^{\text{vn}} = 0{,}07$, $d^{\text{ln}} = 0{,}38$
$d^{\text{M}} = 0{,}19$

ouders van bruidegom van akte 42, Zwollerkerspel, 1853

Willem Frederik Looman	×	Reintjen Engbers
geboren: 1733 – 1818		geboren: 1780 – 1818

huwenden van akte 31, Deventer, 1830

Willem Frederik Looman	×	Rensien Engberts
geboren: 1801		geboren: 1801

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{gvn}} = 0{,}38$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}38$, $d^{\mathrm{ln}} = 0{,}13$

$d^{\mathrm{M}} = 0{,}20$

Matches met $d^{\mathrm{M}} \in [0{,}2; 0{,}3\rangle$:

ouders van bruidegom van akte 3, Grubbenvorst, 1921

Jacob Hovens × Johanna Ambaum
geboren: 1795 – 1880 geboren: 1842 – 1880

huwenden van akte 3, Belfeld, 1879

Jacobus Hubertus Hovens × Johanna Ambaum
geboren: 1846 geboren: 1846

$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}40$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$
$d^{\mathrm{M}} = 0{,}20$

---

ouders van bruidegom van akte 29, Sappemeer, 1880

Harm Schothorst × Hendrikje Huisman
geboren: 1759 – 1844 geboren: 1806 – 1844

huwenden van akte 12, Noordbroek, 1850

Harm Schothorst × Hinderkje Alberts Huisman
geboren: 1827 geboren: 1828

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$
$d^{\mathrm{vn}} = 0{,}00$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}40$, $d^{\mathrm{ln}} = 0{,}00$
$d^{\mathrm{M}} = 0{,}20$

---

ouders van bruidegom van akte 51, Maastricht, 1846

Paulus Preghter × Maria Gertrudis Ceulenberger
geboren: 1710 – 1795 geboren: 1757 – 1795

huwenden van akte 262, Maastricht, 1808

Paul Franois Preghter × Marie Gertrude Culenberger
geboren: 1785 geboren: 1785

$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}40$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}10$, $d^{\mathrm{ln}} = 0{,}08$
$d^{\mathrm{M}} = 0{,}21$

---

ouders van bruid van akte 18, Meerssen, 1865

Louis Gulikers × Gertruid Peusen
geboren: 1732 – 1817 geboren: 1779 – 1817

huwenden van akte 44, Meerssen, 1829

Lodewijk Gulikers × Maria Gertruid Peussens
geboren: 1802 geboren: 1802

$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$
$d^{\mathrm{vn}} = 0{,}10$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}33$, $d^{\mathrm{ln}} = 0{,}25$
$d^{\mathrm{M}} = 0{,}21$

ouders van bruid van akte 8, Simpelveld, 1906

Paulus Joseph Bosch × Juliana Lennarts
geboren: 1777 – 1862 geboren: 1824 – 1862

huwenden van akte 7, Wittem, 1875

Pualus Josephus Bosch × Juliana Lennens
geboren: 1845 geboren: 1845

$d^{\text{gvn}} = 0{,}25$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}25$, $d^{\text{ln}} = 0{,}00$ $d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}38$
$d^{\text{M}} = 0{,}22$

ouders van bruidegom van akte 1, Delfzijl, 1880

Harm Jonk × Epke Brons
geboren: 1758 – 1843 geboren: 1805 – 1843

huwenden van akte 30, Delfzijl, 1842

Harm Jans Jonk × Epke Wessels Brons
geboren: 1814 geboren: 1819

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$
$d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}00$ $d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}00$
$d^{\text{M}} = 0{,}24$

ouders van bruidegom van akte 23, Wissenkerke (NB), 1896

Hendrik van der Maas × Jacoba Leonora Bakker
geboren: 1775 – 1860 geboren: 1822 – 1860

huwenden van akte 26, Wissenkerke (NB), 1859

Hendrik van der Maas × Jacoba Heleonora Bakker
geboren: 1839 geboren: 1837

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}50$
$d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}00$ $d^{\text{vn}} = 0{,}50$, $d^{\text{ln}} = 0{,}00$
$d^{\text{M}} = 0{,}25$

ouders van bruidegom van akte 1, Est en Opijnen, 1868

Hendrik Verhoeks × Maria van den Berg
geboren: 1739 – 1824 geboren: 1786 – 1824

huwenden van akte 1, Waardenburg, 1815

Hendrik Verbeek × Maria Magdalena van den Berg
geboren: 1794 geboren: 1792

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$
$d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}38$ $d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}00$
$d^{\text{M}} = 0{,}25$

ouders van bruid van akte 1, Oostburg, 1872

Leendert de Nijs × Sara Judith Wensch
geboren: 1746 – 1831 geboren: 1793 – 1831

huwenden van akte 3, Retranchement, 1838

Lindert de Nijz × Sara Wens
geboren: 1807 geboren: 1813

$d^{\text{gvn}} = 0{,}10$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$
$d^{\text{vn}} = 0{,}10$, $d^{\text{ln}} = 0{,}25$ $d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}33$
$d^{\text{M}} = 0{,}27$

ouders van bruidegom van akte 15, Delfzijl, 1872

Ente Nijhof	×	Kunje Kiwiet
geboren: 1741 – 1826		geboren: 1788 – 1826

huwenden van akte 13, Delfzijl, 1834

Ente Hindriks Niehoff	×	Kunje Lammerts Kiewiet
geboren: 1803		geboren: 1811
$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$		$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$
$d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}29$		$d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}14$

$d^{\text{M}} = 0{,}28$

Matches met $d^{\mathrm{M}} \in [0{,}3; 0{,}4\rangle$:

ouders van bruidegom van akte 2, Gulpen, 1922
Pieter Koymans × Anna Elissen
geboren: 1772 – 1857 geboren: 1819 – 1857

huwenden van akte 4, Gulpen, 1866
Peter Koeijmans × Maria Anna Hubertina Elissen
geboren: 1839 geboren: 1839

$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$
$d^{\mathrm{vn}} = 0{,}10$, $d^{\mathrm{ln}} = 0{,}33$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}00$
$d^{\mathrm{M}} = 0{,}30$

ouders van bruidegom van akte 45, Wildervank, 1847
Klaas Hindriks Kruithof × Tunisje Arends Paus
geboren: 1725 – 1810 geboren: 1772 – 1810

huwenden van akte 15, Wildervank, 1822
Kars Hindriks Kruithof × Teuntien Arends Paas
geboren: 1797 geboren: 1790

$d^{\mathrm{gvn}} = 0{,}28$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$
$d^{\mathrm{vn}} = 0{,}28$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}25$
$d^{\mathrm{M}} = 0{,}31$

ouders van bruid van akte 21, Steenwijkerwold, 1831
Dirk Hendriks Hogenkamp × Jantje Egbers
geboren: 1710 – 1795 geboren: 1757 – 1795

huwenden van akte 16, Enschede, 1813
Hendrik Horstkamp × Janna Evers
geboren: 1781 geboren: 1786

$d^{\mathrm{gvn}} = 0{,}13$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}42$, $d^{\mathrm{ln}} = 0{,}33$ $d^{\mathrm{vn}} = 0{,}10$, $d^{\mathrm{ln}} = 0{,}33$
$d^{\mathrm{M}} = 0{,}32$

ouders van bruidegom van akte 25, Laren, 1905
Hendrik Jan Bielderman × Hendrika Nijenhuis
geboren: 1771 – 1856 geboren: 1818 – 1856

huwenden van akte 14, Hengelo, 1862
Gerrit Jan Beltman × Hendrica Nijhuis
geboren: 1835 geboren: 1841

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$ $d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}40$ $d^{\mathrm{vn}} = 0{,}10$, $d^{\mathrm{ln}} = 0{,}22$
$d^{\mathrm{M}} = 0{,}34$

ouders van bruidegom van akte 8, Beek, 1890

Wilhelmus Thijssen × Elisabeth Heijnen
geboren: 1754 – 1839 geboren: 1801 – 1839

huwenden van akte 9, Spaubeek, 1852

Joannes Wilhelmus Tyssen × Catharina Elysabeth Heynen
geboren: 1807 geboren: 1807

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$ $d^{\text{gvn}} = 0{,}11$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$
$d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}38$ $d^{\text{vn}} = 0{,}41$, $d^{\text{ln}} = 0{,}29$
$d^{\text{M}} = 0{,}35$

ouders van bruidegom van akte 437, Groningen, 1905

Johannes Bernardus de Boer × Johanna Christina Jans
geboren: 1775 – 1860 geboren: 1822 – 1860

huwenden van akte 15, Raalte, 1875

Johannes Boers × Johanna Maria Jansen
geboren: 1848 geboren: 1848

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}50$
$d^{\text{vn}} = 0{,}33$, $d^{\text{ln}} = 0{,}20$ $d^{\text{vn}} = 0{,}50$, $d^{\text{ln}} = 0{,}33$
$d^{\text{M}} = 0{,}36$

ouders van bruid van akte 3, Markelo, 1916

Berend Jan Leferink × Jenneken Wibbelink
geboren: 1794 – 1879 geboren: 1841 – 1879

huwenden van akte 4, Markelo, 1891

Hendrik Jan Lubberdink × Jenneken Wolberink
geboren: 1856 geboren: 1842

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}50$ $d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}50$, $d^{\text{ln}} = 0{,}40$ $d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}33$
$d^{\text{M}} = 0{,}36$

ouders van bruid van akte 22, Geldermalsen, 1897

Hendrikus Verbeek × Anthonia Verbeek
geboren: 1765 – 1850 geboren: 1812 – 1850

huwenden van akte 6, Driel, 1837

Hendrikus Versteeg × Anna Verhoekx
geboren: 1815 geboren: 1812

$d^{\text{gvn}} = 0{,}00$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$ $d^{\text{gvn}} = 0{,}50$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}00$, $d^{\text{ln}} = 0{,}38$ $d^{\text{vn}} = 0{,}50$, $d^{\text{ln}} = 0{,}38$
$d^{\text{M}} = 0{,}36$

ouders van bruid van akte 71, Roermond, 1910

Johannes Andreas Janssen × Johanna Maria Brouwers
geboren: 1785 – 1870 geboren: 1832 – 1870

huwenden van akte 29, Raalte, 1886

Joannes Jansman × Joanna Broeks
geboren: 1859 geboren: 1856

$d^{\text{gvn}} = 0{,}10$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$ $d^{\text{gvn}} = 0{,}10$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}33$
$d^{\text{vn}} = 0{,}40$, $d^{\text{ln}} = 0{,}29$ $d^{\text{vn}} = 0{,}40$, $d^{\text{ln}} = 0{,}38$
$d^{\text{M}} = 0{,}37$

ouders van bruidegom van akte 11, Haren, 1881

Jans Smit × Jantien Westerhof
geboren: 1756 – 1841 geboren: 1803 – 1841

huwenden van akte 5, Staphorst, 1830

Jan van Spil × Jentje Alberts Westerhuis
geboren: 1803 geboren: 1809

$d^{\mathrm{gvn}} = 0{,}25$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}25$, $d^{\mathrm{ln}} = 0{,}50$

$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{vn}} = 0{,}40$, $d^{\mathrm{ln}} = 0{,}30$

$d^{\mathrm{M}} = 0{,}38$

Matches met $d^{\mathrm{M}} \in [0{,}4; 0{,}5\rangle$:

ouders van bruidegom van akte 1, Thorn, 1846

Michiel Cuypers × Anna Maria Hermans
geboren: 1723 – 1808 geboren: 1770 – 1808

huwenden van akte 29, Meerlo, 1825

Matthies Kuijpers × Maria Margriet Hermans
geboren: 1795 geboren: 1795

$d^{\mathrm{gvn}} = 0{,}50$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$
$d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}38$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}00$

$d^{\mathrm{M}} = 0{,}40$

---

ouders van bruidegom van akte 3, Beesd, 1895

Dirk van Goor × Teuntje Willemina van Oijen
geboren: 1761 – 1846 geboren: 1808 – 1846

huwenden van akte 1, Maurik, 1842

Gerrit Dirk van de Geer × Arreke Willemina van Ojen
geboren: 1812 geboren: 1820

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$
$d^{\mathrm{vn}} = 0{,}33$, $d^{\mathrm{ln}} = 0{,}50$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}20$

$d^{\mathrm{M}} = 0{,}40$

---

ouders van bruid van akte 55, Hellendoorn, 1900

Hendrikus Hiddink × Evertdina Hendrika Broekmate
geboren: 1777 – 1862 geboren: 1824 – 1862

huwenden van akte 44, Laren, 1862

Hendrik Jan Haijtink × Bernarda Hendrika ten Broeke
geboren: 1835 geboren: 1841

$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$
$d^{\mathrm{vn}} = 0{,}40$, $d^{\mathrm{ln}} = 0{,}38$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}33$

$d^{\mathrm{M}} = 0{,}41$

---

ouders van bruidegom van akte 102, Deventer, 1876

Jan Schoens × Anna Maria van der Logt
geboren: 1749 – 1834 geboren: 1796 – 1834

huwenden van akte 1, Wijlre, 1837

Jan Egidius Schijns × Maria Gertrudis van Loo
geboren: 1804 geboren: 1804

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$
$d^{\mathrm{vn}} = 0{,}33$, $d^{\mathrm{ln}} = 0{,}29$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}50$

$d^{\mathrm{M}} = 0{,}42$

ouders van bruidegom van akte 12, Heerde, 1902

Berent Jan Westenberg × Gerritdina Stein
geboren: 1764 – 1849 geboren: 1811 – 1849

huwenden van akte 86, Deventer, 1863

Jan Willem Wittenberg × Geertjen Sterk
geboren: 1810 geboren: 1825

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$ $d^{\mathrm{gvn}} = 0{,}50$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}00$
$d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}20$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}40$
$d^{\mathrm{M}} = 0{,}42$

ouders van bruid van akte 85, Zwolle, 1870

Johan Christoffel Staal × Sara Helena Maria Staal
geboren: 1747 – 1832 geboren: 1794 – 1832

huwenden van akte 1, Duivendijke, 1846

Johannes Stoel × Sara Stoel
geboren: 1820 geboren: 1820

$d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$
$d^{\mathrm{vn}} = 0{,}40$, $d^{\mathrm{ln}} = 0{,}40$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}40$
$d^{\mathrm{M}} = 0{,}43$

ouders van bruid van akte 4, Stedum, 1868

Pieter Jakobs Vink × Geeske Pieters Kiel
geboren: 1744 – 1829 geboren: 1791 – 1829

huwenden van akte 18, Ten Boer, 1834

Pieter Kornelis Vink × Tjaaktje Pieters Kol
geboren: 1805 geboren: 1818

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$ $d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$
$d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}00$ $d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}50$
$d^{\mathrm{M}} = 0{,}43$

ouders van bruidegom van akte 12, Denekamp, 1893

Assuerus Hendriks Brandsma × Jourica Hendriks van der Werf
geboren: 1764 – 1849 geboren: 1811 – 1849

huwenden van akte 8, Staphorst, 1851

Hendrik Brand × Hendrikje Wolf
geboren: 1828 geboren: 1829

$d^{\mathrm{gvn}} = 0{,}13$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$ $d^{\mathrm{gvn}} = 0{,}22$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$
$d^{\mathrm{vn}} = 0{,}42$, $d^{\mathrm{ln}} = 0{,}38$ $d^{\mathrm{vn}} = 0{,}48$, $d^{\mathrm{ln}} = 0{,}50$
$d^{\mathrm{M}} = 0{,}45$

ouders van bruid van akte 36, Avereest, 1866

Jan Hendriks Wind × Jacobje Meesters
geboren: 1744 – 1829 geboren: 1791 – 1829

huwenden van akte 89, Middelburg, 1826

Jan Dirk de Wit × Helena Johanna Messer
geboren: 1798 geboren: 1802

$d^{\mathrm{gvn}} = 0{,}00$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}50$ $d^{\mathrm{gvn}} = 0{,}10$, $d^{\mathrm{vvn}} = 0{,}00$, $d^{\mathrm{ovn}} = 0{,}33$
$d^{\mathrm{vn}} = 0{,}50$, $d^{\mathrm{ln}} = 0{,}50$ $d^{\mathrm{vn}} = 0{,}40$, $d^{\mathrm{ln}} = 0{,}38$
$d^{\mathrm{M}} = 0{,}45$

ouders van bruidegom van akte 9, Oude Pekela, 1916

Berend Kok	×	Maria Bos
geboren: 1791 – 1876		geboren: 1838 – 1876

huwenden van akte 9, Uithuizermeeden, 1877

Berteld Kooi	×	Martje de Boer
geboren: 1852		geboren: 1851

$d^{\text{gvn}} = 0{,}29$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}29$, $d^{\text{ln}} = 0{,}50$

$d^{\text{gvn}} = 0{,}50$, $d^{\text{vvn}} = 0{,}00$, $d^{\text{ovn}} = 0{,}00$
$d^{\text{vn}} = 0{,}50$, $d^{\text{ln}} = 0{,}50$

$d^{\text{M}} = 0{,}46$

Bijlage B

Certilink

Het hart van de software die ik voor dit project heb geschreven is Certilink. Deze applicatie ontleent z'n naam aan de samentrekking van "Certificate Linker", een omschrijving van de taak van het programma. De software is geschreven in Borland Delphi 2005 voor Windows en communiceert met een MySQL-database. Voor deze combinatie is gekozen op verzoek van het IISG. De werking van Certilink komt grotendeels overeen met de algoritmen die besproken zijn in deze scriptie. Deze bijlage beperkt zich dan ook tot een korte beschrijvende handleiding.

Installatie

De applicatie bestaat uit twee bestanden: `certilink.exe` en `settings.dat`. Het eerste is de daadwerkelijke gecompileerde applicatie en het tweede is een bestandje waarin de laatste instellingen van het programma bewaard worden. Dit bestandje wordt gelezen en gewijzigd door Certilink, maar het kan ook worden geopend met iedere ASCII-editor, zoals Notepad.

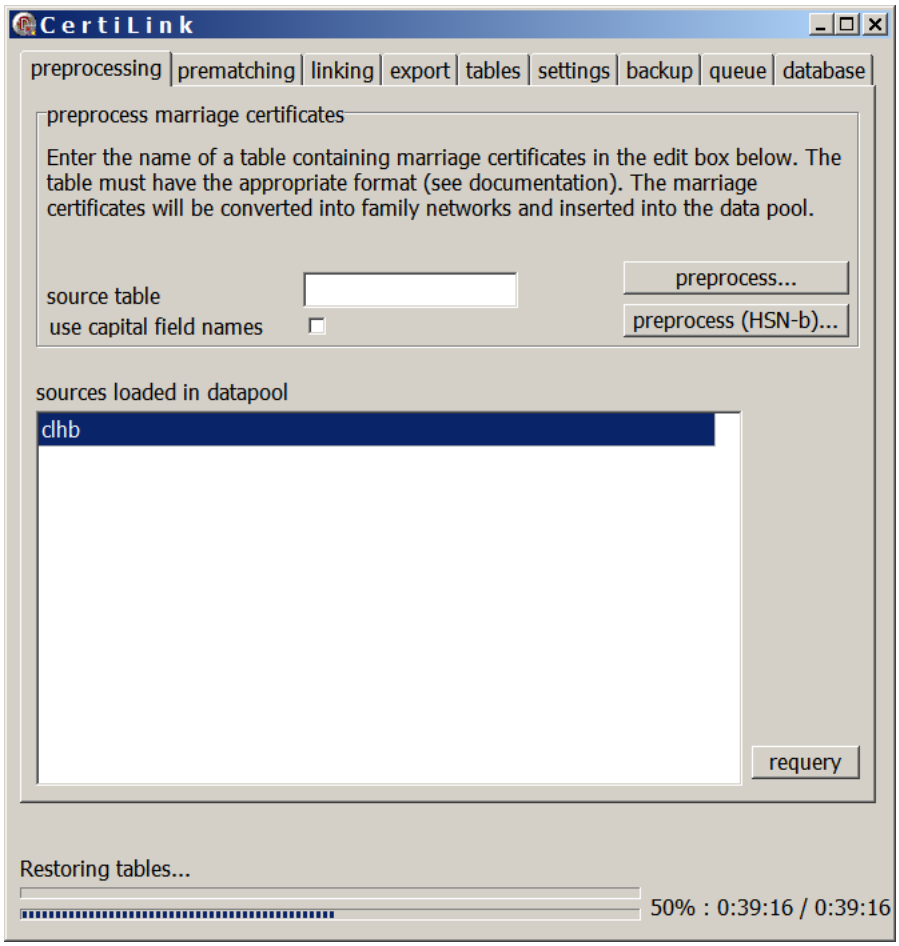
Naast deze twee bestanden moet Certilink via een ODBC-koppeling kunnen communiceren met een MySQL-database. Hiervoor moet de MySQL-driver voor ODBC geïnstalleerd zijn. Daarnaast moet de database zelf natuurlijk klaar staan voor gebruik.

Tabelstructuur

Certilink verdeelt gegevens in drie categorieën. De eerste wordt de *datapool* genoemd. Dit zijn de genormaliseerde genealogische data waarmee matches worden gemaakt. In de praktijk komt dit neer op de huwelijksakten van één of meer provincies die moeten worden gematcht en eventueel geboorteakten. De tabellen P en G worden hier geïmplementeerd als `clpernode` en `clfamnode` en vormen de datapool.

Daarnaast is er de *matchpool*. Hierin komen alle matches terecht die gemaakt worden met de genealogische informatie in de datapool. Deze pool bestaat uit tabellen de tabellen PM, GM en M, geïmplementeerd als `clpermatch`, `clfammatch` en `clmatch`.

Tenlotte is er de *prematchpool*. Hierin wordt prematchinformatie opgeslagen zoals een cumulatieve lijst van unieke voornamen, unieke achternamen en naammatches, die onafhankelijk zijn van de datapool. Tot deze pool behoren de tabellen A (unieke achternamen) geïmplementeerd als `clln`, V (unieke gehele voornamen) als `clfn`, L (unieke losse voornamen) als `clsfn`, S (relaties tussen V en L) als `clcomp`, AM (matches tussen achternamen) als `cllnmatch`, VM (matches tussen



Figuur B.1: Preprocessing

gehele voornamen) als `clfnmatch` en *LM* (matches tussen losse voornamen) geïmplementeerd als `clsfnmatch`.

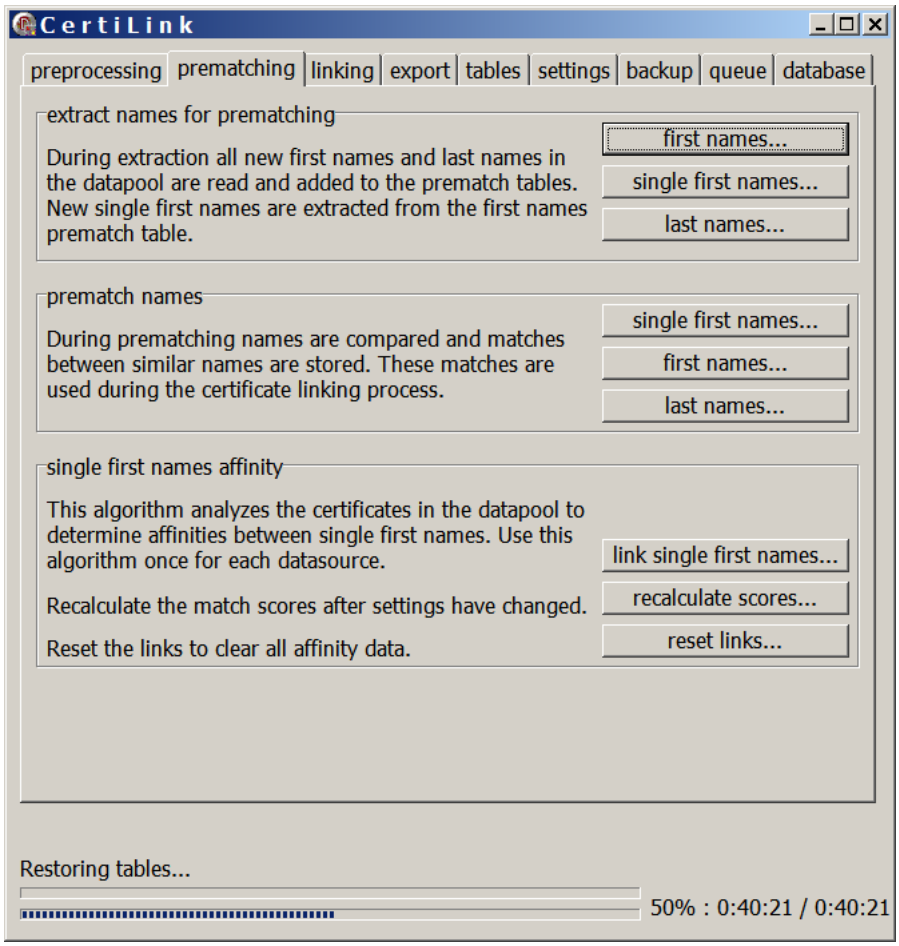
Preprocessing

De eerste stap in het matchproces is het inlezen van de brongegevens. Dit kan worden aangestuurd vanuit het tabblad *preprocessing* in figuur B.1. Voordien moet een tabel met brongegevens gekopieerd worden naar de database. Daarin voorziet Certilink niet. Dat moet dus met de hand gebeuren. Daarna kan het programma de gegevens uit deze tabel inlezen, normaliseren en toevoegen aan de datapoel. Hiervoor zijn de volgende elementen aanwezig:

sourcetable: Nadat een brontabel op de database is gezet kan de naam van de tabel hier worden ingevuld voor het invoerproces.

preprocess: Deze knop start het inlezen van de gegevens van de opgegeven brontabel en voegt die toe aan de datapoel. Deze aktie is alleen geschikt voor tabellen met huwelijksakten zoals die door Genlias zijn aangeleverd. Die moeten minstens de velden bevatten die vermeld staan in tabel B.1 op pagina 94.

use capital field names: Vink deze optie aan om oudere tabellen van Genlias in te lezen. Deze tabellen voldoen aan dezelfde formaateisen, maar



Figuur B.2: Prematching

hebben alle velden in hoofdletters en het veld “IDNR” in plaats van het veld “id”.

preprocess (HSB-b): Met deze knop wordt de invoer van HSN-geboorteakten gestart. Een brontabel met geboorteakten heeft een ander formaat dan één met huwelijksakten. Tabel B.2 op pagina 95 toont de minimale velden.

sources loaded in datapool: Hier kun je zien welke gegevens op dit moment al in de datapoel aanwezig zijn.

requery: Na een wijziging in de datapoel kan met deze knop de database worden geraadpleegd om de informatie over de gegevens in de datapoel te verversen.

Prematching

Wanneer de data ingelezen zijn, genormaliseerd en toegevoegd aan de datapoel zijn we nog niet klaar om te matchen. Eerst moet een aantal prematch-handelingen worden verricht, het prematchen van de achternamen en van de losse voornamen. Deze activiteiten kun je in gang zetten op de tab *prematching*, figuur B.2. Hier kunnen de volgende taken worden gestart:

veldnaam	datatype	*NULL*?	omschrijving
id	int	nee	primaire sleutel
gmsdcn	char(3)	nee	aktenummer
gmsdcd	int	nee	aktedatum: dag in maand
gmsdcm	int	nee	aktedatum: maand
gmsdcy	int	nee	aktedatum: jaar
gmsdcp	char(24)	nee	akteplaats
gmsdat	char(30)	ja	aktetype
gmgrfn	char(60)	ja	voornamen bruidegom
gmgrpf	char(20)	ja	voorvoegsel achternaam bruidegom
gmgrln	char(50)	ja	achternaam bruidegom
gmgroc	char(70)	ja	beroep bruidegom
gmgrbl	char(36)	ja	geboorteplaats bruidegom
gmgrbd	int	ja	geboortedatum bruidegom: dag in maand
gmgrbm	int	ja	geboortedatum bruidegom: maand
gmgrby	int	ja	geboortedatum bruidegom: jaar
gmgray	int	ja	leeftijd bruidegom in jaren
gmgffn	char(60)	ja	voornamen vader bruidegom
gmgfpf	char(20)	ja	voorvoegsel achternaam vader bruidegom
gmgfln	char(50)	ja	achternaam vader bruidegom
gmgfoc	char(70)	ja	beroep vader bruidegom
gmgmfn	char(60)	ja	voornamen moeder bruidegom
gmgmpf	char(20)	ja	voorvoegsel achternaam moeder bruidegom
gmgmln	char(50)	ja	achternaam moeder bruidegom
gmgmoc	char(70)	ja	beroep moeder bruidegom
gmbrfn	char(60)	ja	voornamen bruid
gmbrpf	char(20)	ja	voorvoegsel achternaam bruid
gmbrln	char(50)	ja	achternaam bruid
gmbroc	char(70)	ja	beroep bruid
gmbrbl	char(36)	ja	geboorteplaats bruid
gmbrbd	int	ja	geboortedatum bruid: dag in maand
gmbrbm	int	ja	geboortedatum bruid: maand
gmbrby	int	ja	geboortedatum bruid: jaar
gmbray	int	ja	leeftijd bruid in jaren
gmbffn	char(60)	ja	voornamen vader bruid
gmbfpf	char(20)	ja	voorvoegsel achternaam vader bruid
gmbfln	char(50)	ja	achternaam vader bruid
gmbfoc	char(70)	ja	beroep vader bruid
gmbmfn	char(60)	ja	voornamen moeder bruid
gmbmpf	char(20)	ja	voorvoegsel achternaam moeder bruid
gmbmln	char(50)	ja	achternaam moeder bruid
gmbmoc	char(70)	ja	beroep moeder bruid

Tabel B.1: Het vereiste formaat voor brontabellen met huwelijksakten.

veldnaam	datatype	*NULL*?	omschrijving
id	int	nee	primaire sleutel
hbsdcn	int	ja	aktenummer
hbsdcp	char(24)	ja	akteplaats
hbsdcd	int	ja	aktedatum: dag in maand
hbsdcm	int	ja	aktedatum: maand
hbsdcy	int	ja	aktedatum: jaar
hbchfn	char(60)	ja	voornamen kind
hbchpf	char(20)	ja	voorvoegsel achternaam kind
hbchln	char(50)	ja	achternaam kind
hbchsx	char(1)	ja	geslacht kind ("m" of "f")
hbchbl	char(40)	ja	geboorteplaats kind
hbchbd	int	ja	geboortedatum kind: dag in maand
hbchbm	int	ja	geboortedatum kind: maand
hbchby	int	ja	geboortedatum kind: jaar
hbcffn	char(60)	ja	voornamen vader
hbcfpf	char(20)	ja	voorvoegsel achternaam vader
hbcfln	char(50)	ja	achternaam vader
hbcmfn	char(60)	ja	voornamen moeder
hbcmpf	char(20)	ja	voorvoegsel achternaam moeder
hbcmln	char(50)	ja	achternaam moeder

Tabel B.2: Het vereiste formaat voor brontabellen met geboorteakten.

extract names for prematching:

first names: Met deze knop worden nieuwe voornamen in de datapoel toegevoegd aan V.

single first names: Deze taak bestaat uit het splitsen van nieuwe voornamen in V in losse voornamen en deze wanneer nodig toevoegen aan L.

last names: Deze knop start het toevoegen van nieuwe achternamen in de datapoel aan A.

prematch names: Deze drie knoppen starten het prematchen van losse voornamen in L, voornamen in V en achternamen in A. Bij alle drie de taken worden alleen de nieuwe namen vergeleken met alle namen in de tabel. De taak *first names* kan nog wel uitgevoerd worden, maar duurt erg lang en is niet meer nodig (zie paragraaf 5.4).

single first names

first names

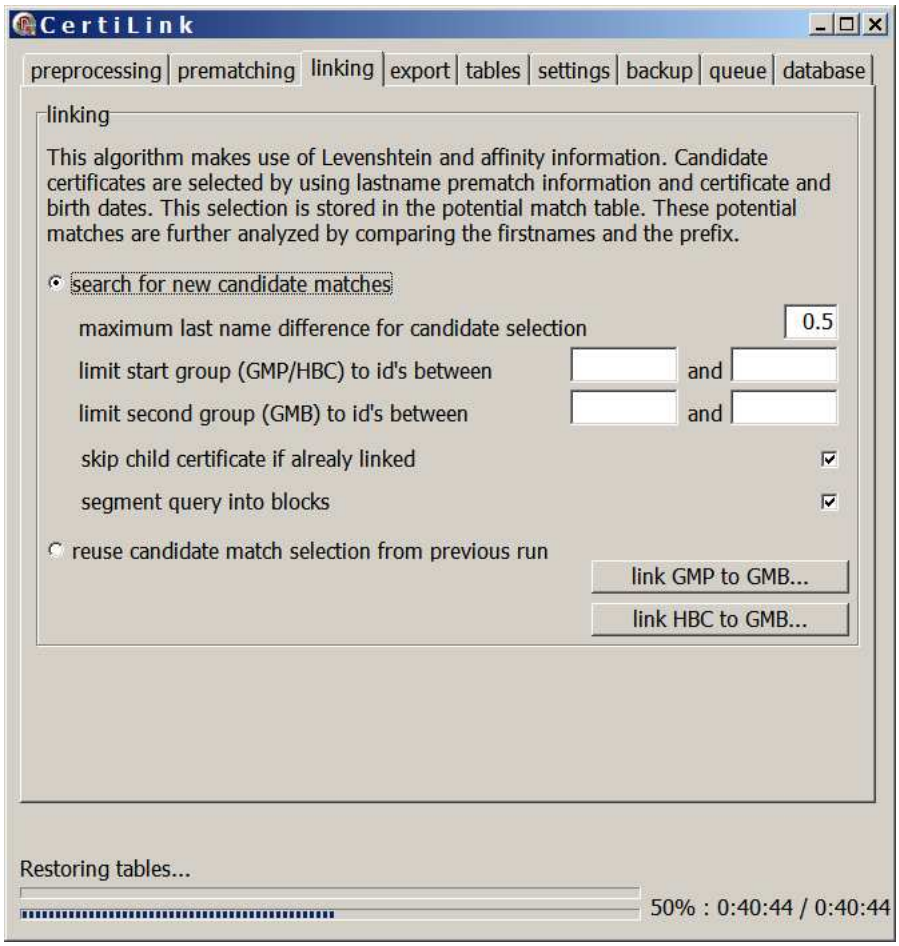
last names

single first names affinity: Het verwantschapsalgoritme maakt extra koppelingen tussen vermoedelijk verwante losse voornamen. Dit gebeurt aan de hand van de brondata (zie hoofdstuk 4). Dit hoeft maar één keer te gebeuren per nieuwe dataset, nadat de nieuwe losse voornamen zijn toegevoegd.

link single first names: Met deze knop wordt het zoeken naar koppelingen gestart op basis van de dataset die op dit moment in de datapoel zit.

recalculate scores: Of twee losse voornamen als verwant worden beschouwd hangt af van het gevonden aantal koppelingen en een aantal instellingen (zie *settings*). Wanneer deze instellingen veranderen kan met de taak *recalculate scores* opnieuw worden bepaald welke namen verwant zijn.

reset links: Met deze taak wordt alle verwantsschapsinformatie gewist. Er zal alleen worden gematcht met Levenshteinafstand totdat de taak *link single first names* weer is uitgevoerd op een bepaalde dataset.



Figuur B.3: Linking

Linking

Het daadwerkelijke matchen kan gestart worden op de tab *linking*, weergegeven in figuur B.3. Het programma zoekt hiervoor naar paren akten die op basis van prematchinformatie mogelijke goede matches opleveren. Wanneer deze kandidaten gevonden zijn worden de paren één voor één afgelopen om de matchkwaliteit te beoordelen. Eventueel kan het zoeken naar kandidaten overgeslagen worden en een nieuwe matchronde uitgevoerd worden op basis van de kandidatenlijst die bij de vorige matchronde is opgebouwd.

Op deze tab worden de afkortingen GMB, GMP en HBC gebruikt. Die staan voor de categorieën te matchen persoonsknopen. GMP duidt de ouders op de huwelijksakten aan (Genlias Marriage certificate Parents). Deze willen we matchen met GMB, de huwenden op de huwelijksakten (Genlias Marriage certificate Bride & bridegroom). Daarnaast hebben we ook HBC, de kinderen van de geboorteakten (HSN Birth certificate Children). Deze groep kunnen we matchen met de GMB-categorie. Deze processen kunnen we als volgt instellen:

search for new candidate matches: Kies deze optie om een nieuwe lijst met kandidaatparen akten te maken voor de matchronde. Het matchproces begint met het doorzoeken van de database om deze lijst samen te stellen. Hierbij kan het volgende ingesteld worden:

maximum last name difference for candidate selection: Certilink zoekt alleen naar kandidaatmatches die achternamen hebben met een d^{an} die kleiner of gelijk is aan deze waarde. Door deze lager dan 0,5 in te stellen, wordt een kleinere maar snellere kandidatenlijst opgebouwd.

limit start group (GMP/HBC) to id's between: De kandidaatmatches worden gezocht uit combinaties van alle ouderparen en huwende paren. Wanneer HSN-geboorteakten worden gematcht met Genlias-huwelijksakten wordt gezocht uit combinaties van alle kinderen en huwenden. Door hier een minimum- en maximum-id in te vullen worden niet alle ouderparen of kinderen meegenomen, maar alleen die uit het opgegeven id-bereik. Deze beperkte selectie wordt wel vergeleken met alle huwenden, behalve als gebruik gemaakt wordt van de volgende optie.

limit second group (GMB) to id's between: De kandidaatmatches worden gezocht uit combinaties van alle ouderparen en huwende paren. Door hier een minimum- en maximum-id in te vullen worden niet alle huwende paren meegenomen, maar alleen die uit het opgegeven id-bereik. Hetzelfde geldt wanneer gematcht wordt tussen kinderen en huwenden. In dat geval worden alleen huwenden bekeken in het opgegeven bereik. Deze worden wel vergeleken met alle ouderparen of kinderen, behalve als de vorige optie wordt benut.

skip child certificate if already linked: Deze optie is alleen beschikbaar voor het matchen van huwelijksakten onderling, niet voor het matchen van huwelijksakten met geboorteakten. Wanneer dit is aangevinkt wordt voor ouderparen op de huwelijksakte van hun kind (het "child certificate") geen kandidaatmatch gezocht, als er voor dat paar al een kandidaatmatch in de matchpoel aanwezig is. Op die manier kan een matchpoel worden uitgebreid zonder werk over te doen.

segment query into blocks: Deze optie hakt het zoeken naar kandidaatparen in stukken. Op deze manier krijgt de database niet één grote query maar vele kleine te verwerken. Op de beschikbare MySQL-database van het IISG heeft dit tot betere resultaten geleid.

reuse candidate selection from previous run: Met deze optie wordt de kandidatenlijst van de vorige matchronde gebruikt in plaats van een nieuwe op te bouwen.

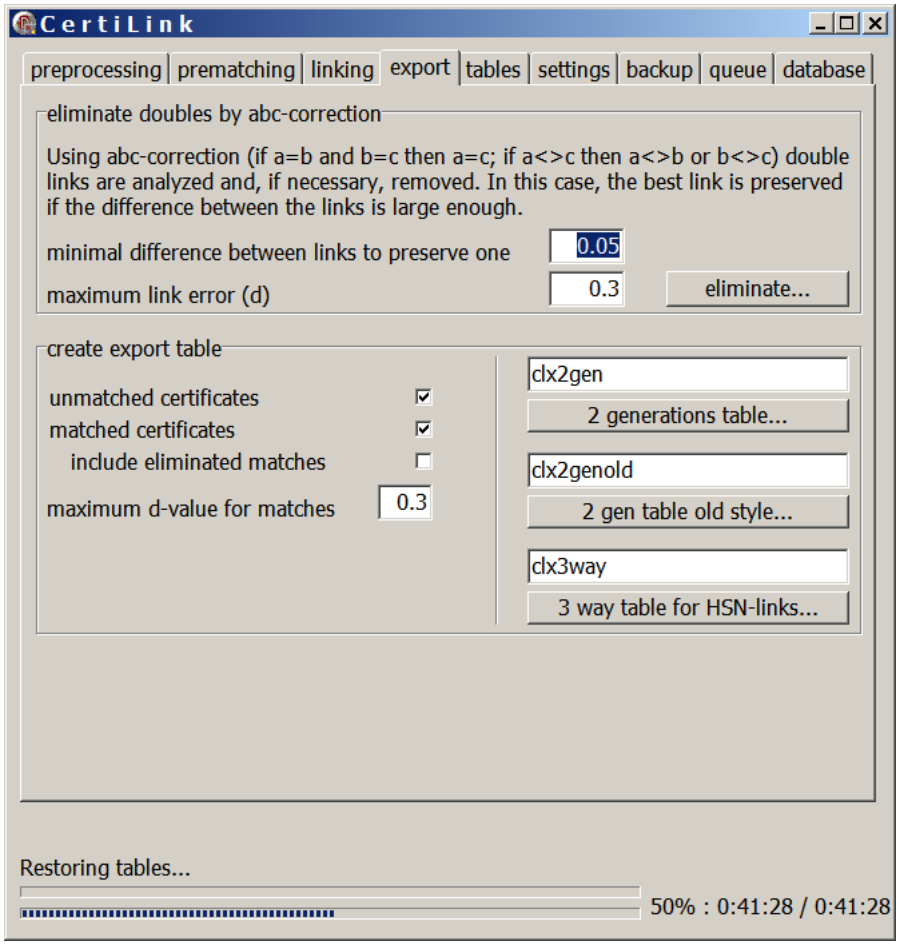
link GMP to GMB: Deze knop start het matchen van ouders op huwelijksakten met huwenden op huwelijksakten.

link HBC to GMB: Deze knop start het matchen van kinderen op geboorteakten met huwenden op huwelijksakten (Genlias Marriage Brides & bridegrooms). Hierbij worden de ouders van de kinderen ook met de ouders van de bruid of bruidegom gematcht.

Export

Op deze tab, getoond in figuur B.4 op de volgende pagina, kan de output van het matchalgoritme opgeschoond en geëxporteerd worden. Ook deze taken zijn instelbaar:

eliminate doubles by abc-correction: Met dit proces worden conflicterende dubbele matches vergeleken en wordt de slechtste afgekeurd. Afgekeurde matches blijven in de database als "eliminated matches". Zie voor meer informatie over het elimineren van conflicten paragraaf 5.5.



Figuur B.4: Export

minimal difference between links to preserve one: Wanneer het verschil tussen de kwaliteit van twee matches kleiner is dan deze waarde, wordt geen keuze tussen de matches gemaakt, maar worden ze beide geëlimineerd. Dit is parameter v uit paragraaf 5.5.

maximum link error (d): Bij conflicterende matches wordt een controlematch gemaakt om te kijken of beide matches tegelijk goed kunnen zijn. Wanneer de controlematch gelijk aan of onder de hier ingevulde waarde is, worden de matches goedgekeurd. Dit is parameter m uit de genoemde paragraaf.

eliminate Deze knop start het proces van het elimineren van dubbele matches.

create export table: De matches zijn in de matchpoel opgeslagen als numerieke verwijzingen naar akten. Met deze opties kan een voor de mens leesbare exporttabel worden gemaakt.

unmatched certificates: Vink deze optie aan om de niet gematchte akten op te nemen in de export tabel. De velden die de match en de andere akte beschrijven blijven dan voor deze akten leeg.

matched certificates: Vink deze optie aan om de gematchte akten op te nemen in de exporttabel.

include eliminated matches: Met deze optie worden de verwijderde dubbele matches toch in de exporttabel opgenomen.

maximum d-value for matches: Vul hier de bovengrens voor de matchkwaliteit in. De exporttabel zal alleen matches met een even grote of betere kwaliteit bevatten.

2 generations table: Hier kan de naam van de exporttabel ingevuld worden en met de knop wordt de tabel gemaakt. Per regel biedt deze tabel informatie over de huwende, zijn of haar gematchte ouders, de informatie van de eigen akte, informatie over de ouders van de gematchte akte waarop ze zelf trouwen en gegevens over de matchkwaliteit. Deze exporttabel is alleen geschikt voor matches tussen huwelijksakten.

2 generations old style: Hier kan een tabel worden gemaakt die vergelijkbaar is met de vorige functie. Het formaat van deze tabel is iets anders.

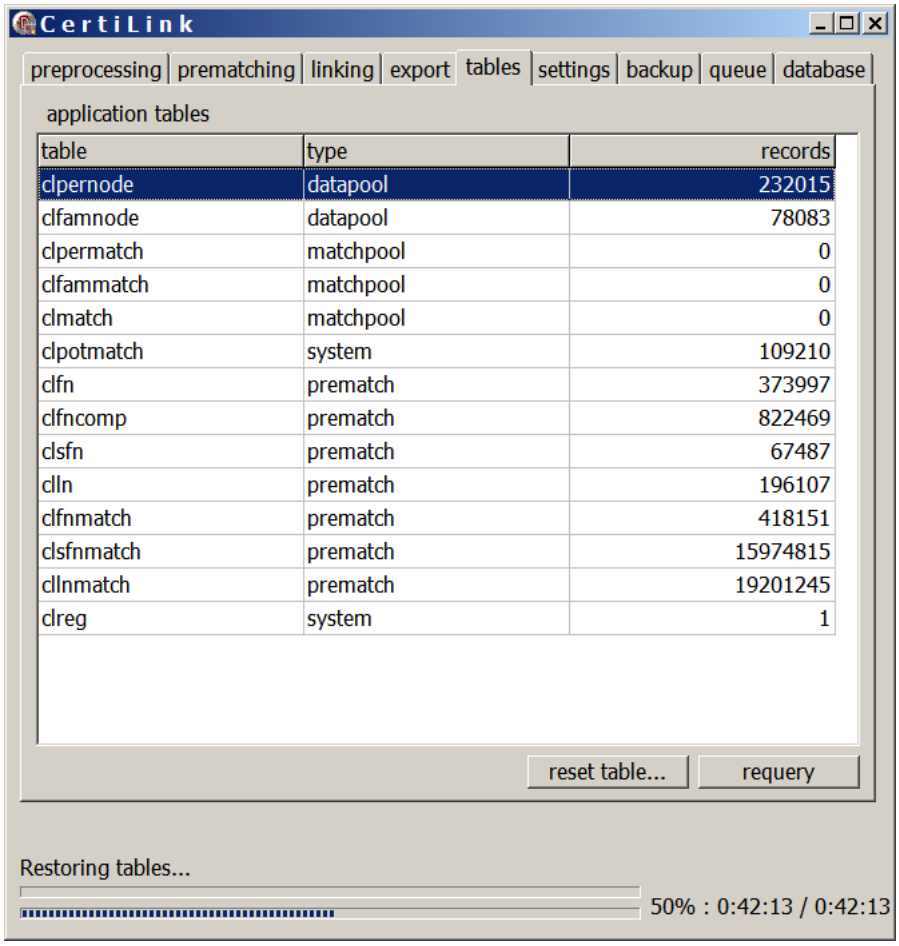
3 way table for HSN-links: Deze exporttabel is geschikt voor matches tussen geboorteakten en huwelijksakten. Per regel staat in de tabel het geboren kind en zijn of haar ouders en de gematchte huwende en diens ouders. Ook heeft ieder regel informatie over de matchkwaliteit.

Tables

De tab *tables* (figuur B.5 op de pagina hierna) stelt ons in staat om enkele elementaire taken met betrekking tot de tabellen uit te voeren. We kunnen zien hoeveel records ieder tabel bevat. Eveneens kunnen we een tabel *resetten*. Hiermee wordt de tabel geleegd. Wanneer Certilink op een nieuwe database aangesloten is, of er ontbreken om een andere reden tabellen, kan met de resetknop een tabel opnieuw worden aangemaakt. De resetknop heeft alleen effect op de geselecteerde tabel. Gebruik de knop *requery* om het overzicht van tabellen te verversen.

Alleen tabellen die voor Certilink van direct belang zijn staan in dit overzicht. Dit zijn:

clpernode: Deze tabel bevat de persoonsknopen. (P)



Figuur B.5: Tables

clfamnode: Deze tabel bevat de gezinsknopen. (G)

clpermatch: In deze tabel staan matches tussen persoonsknopen. (PM)

clfammatch: In deze tabel staan matches tussen gezinsknopen. (GM)

clmatch: In deze tabel staan matches. (M)

clpotmach: In deze tabel wordt de kandidatenlijst voor het matchproces opgeslagen (zie *linking*).

clfn: Dit is de tabel van unieke gehele voornamen. (V)

clfncomp: Dit is de koppeltabel tussen gehele voornamen en de losse voornamen waaruit ze bestaan. (S)

clsfn: Dit is de tabel van unieke losse voornamen. (L)

clln: Dit is de tabel van unieke achternamen. (A)

clfnmatch: Dit is de tabel van de matches tussen voornamen. (VM)

clsfnmatch: Dit is de tabel van matches tussen losse voornamen. Ook de verwantschapsinformatie staat in deze tabel. (LM)

cllnmatch: Dit is de tabel van matches tussen achternamen. (AM)

clreg: In deze tabel wordt systeeminformatie van Certilink bijgehouden.

Settings

Certilink en de algoritmen die het gebruikt kennen veel instellingen en parameters. Deze kunnen worden ingesteld op de tab *settings* (zie B.6 op de volgende pagina). Alle parameters en hun waarden worden gelezen uit en weggeschreven naar `settings.dat`. De waarden kunnen in dit scherm veranderd worden om het gedrag van Certilink aan te passen. De knop *save* schrijft de gewijzigde instellingen naar het bestandje. De wijzigingen blijven dan bewaard na het afsluiten van het programma. Tabellen B.3 en B.4 op pagina's 103 en 104 tonen alle parameters met een korte beschrijving.

Backup

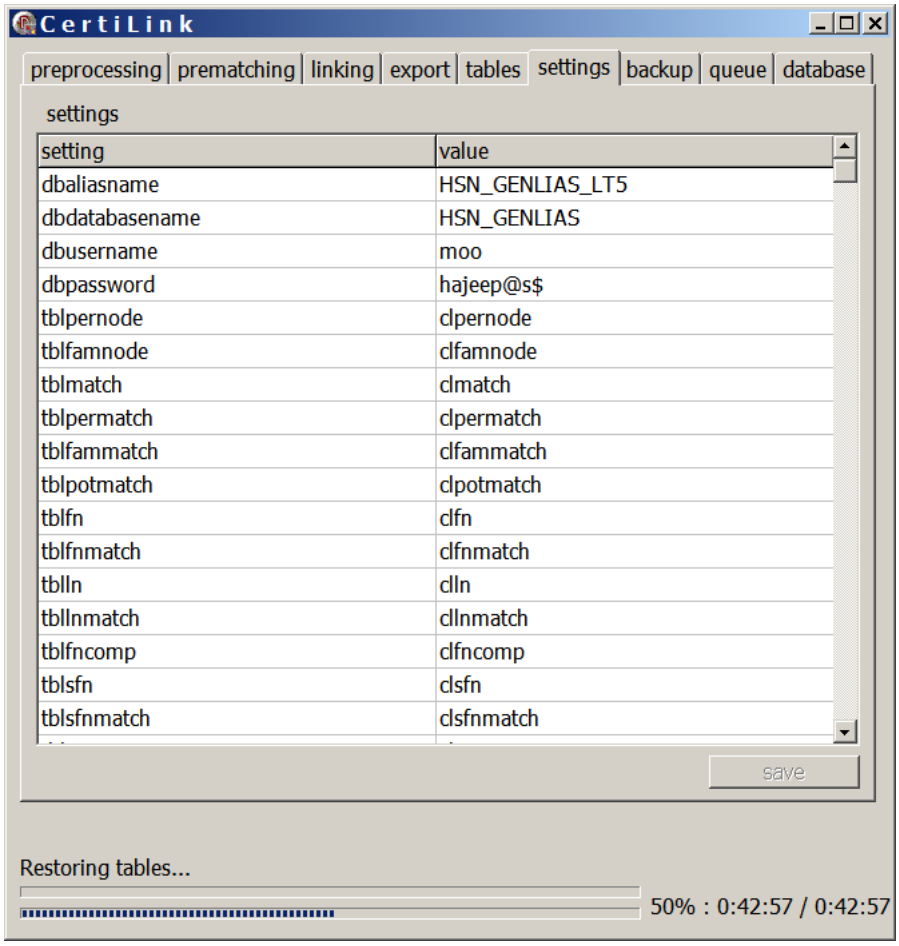
De *backup*-tab, getoond in figuur B.7 op pagina 105, maakt het ons mogelijk om van de datapoel- en matchpoeltabellen evenals de kandidaatmatchtabel en de exporttabellen backups te maken. De tabellen worden gekopieerd op de database en onder een andere naam bewaard. De tabellen kunnen ook worden teruggehaald. De volgende functies staan op dit tabblad:

datapool: Zet deze optie aan om tabellen van de datapoel als backup weg te schrijven of terug te halen.

matchpool: Deze optie selecteert de tabellen van de matchpoel.

potential match table: Hiermee kan de kandidaatmatchestabel worden geselecteerd.

export tables: Met deze optie worden de exporttabellen gekozen. Alle tabellen waarvan de naam met "clx" begint worden beschouwd als exporttabellen.



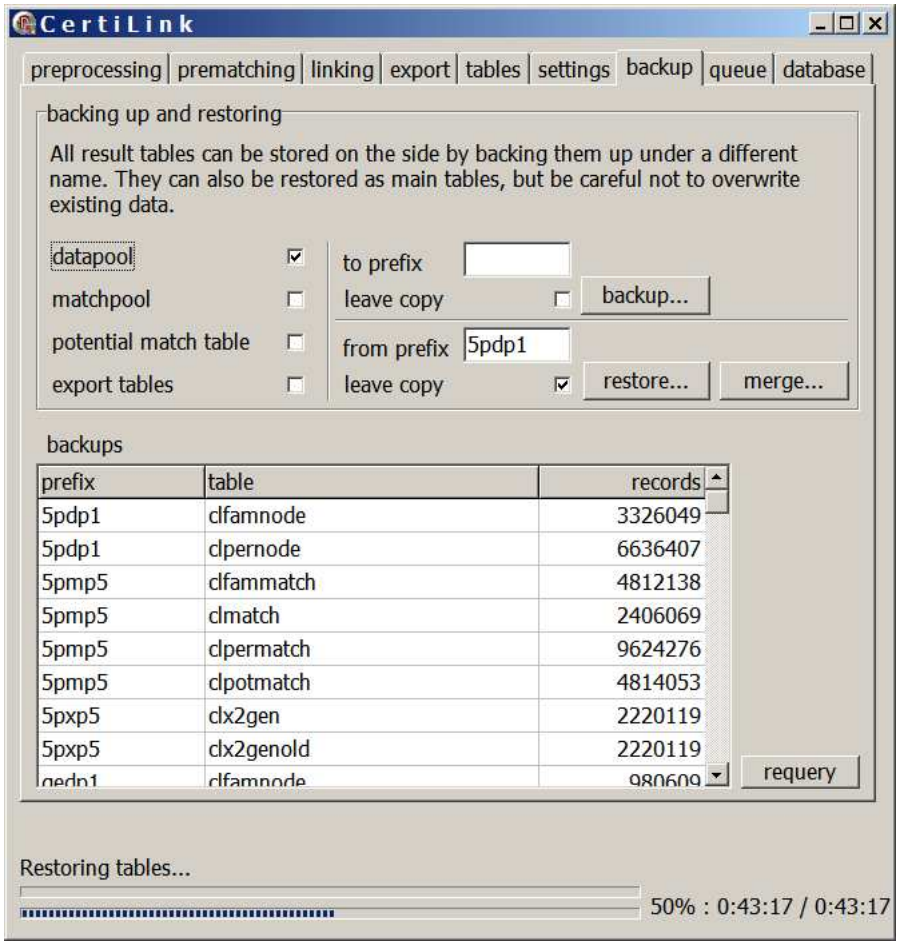
Figuur B.6: Settings

parameter	standaard	omschrijving
dbaliasname		de ODBC-DSN naam van de databaseverbinding
dbdatabasename		de naam van de database op de server
dbusername		de username waarmee op de database ingelogd kan worden
dbpassword		het wachtwoord waarmee op de database ingelogd kan worden
tblpernode	clpernode	de naam van de persoonsknooptabel
tblfamnode	clfamnode	de naam van de gezinsknooptabel
tblmatch	clmatch	de naam van de matchtabel
tblpermatch	clpermatch	de naam van de persoonsknoopmatchtabel
tblfammatch	clfammatch	de naam van de gezinsknoopmatchtabel
tblpotmatch	clpotmatch	de naam van de tabel voor kandidaatmatches
tblfn	clfn	de naam van de voornamentabel
tblfnmatch	clfnmatch	de naam van voornaammatchtabel
tblln	clln	de naam van de achternaamtabel
tbllnmatch	cllnmatch	de naam van de achternaammatchtabel
tblfncomp	clfncomp	de naam van losse voornaamkoppeltabel
tblsfn	clsfn	de naam van de losse voornaamtabel
tblsfnmatch	clsfnmatch	de naam van de losse voornaammatchtabel
tblreg	clreg	de naam van de systeemtabel
linkmaxperd	0.5	de maximale verschilwaarde tussen persoonsknopen voor een match
linkmaxpere	0.5	de maximale onvergelijkbaarheid tussen persoonsknopen voor een match (wordt niet meer gebruikt)
linkmaxfamd	0.5	de maximale verschilwaarde tussen gezinsknopen voor een match
linkmaxfame	0.5	de maximal onvergelijkbaarheid tussen gezinsknopen voor een match (wordt niet meer gebruikt)
linkblocksize	100000	de blockgrootte bij gesegmenteerd matchen (zie *linking*)
pmcombfirstletters	sz,ft,ck,iy	Bij het prematchen van namen worden alleen namen vergeleken met dezelfde eerste letter. Namen waarvan de eerste letters hier samen staan worden ook vergeleken.
pmmaxsfnlevdis	-1	de maximale Levenshteinafstand bij het prematchen van losse voornamen (bij -1 is dit maximum uitgeschakeld)
pmmaxsfndlev	0.5	de maximale relatieve Levenshteinafstand bij het prematchen van losse voornamen
pmaffd	0.1	de verschilwaarde die wordt toegekend aan verwante voornamen
pmaffminassoc	50	de minimale positieve telling voor voornaamverwantschap
pmaffdissocfactor	25	de factor waarmee de positieve telling groter moet zijn dan de negatieve telling voor voornaamverwantschap
pmmaxfndsfn	0.5	het maximale verschil tussen losse voornamen voor een match
pmmaxfndshuffle	1	de maximale verandering in volgorde van losse voornamen voor een match
pmmaxfndunmatched	0.5	het maximum aandeel losse voornamen dat niet gekoppeld kan worden voor een match
pmmaxfnd	1	het maximale totale verschil tussen voornamen voor een match
pmmaxpflevdis	-1	de maximale Levenshteinafstand tussen prefices voor een match (uitgeschakeld)
pmmaxpfdlev	-1	de maximale relatieve Levenshteinafstand tussen prefices voor een match (uitgeschakeld)
pmmaxlnlevdis	-1	de maximale Levenshteinafstand tussen achternamen voor een match (uitgeschakeld)
pmmaxlndlev	0.4	de maximale relatieve Levenshteinafstand tussen achternamen voor een match
pmmaxlnxlevdis	2	de bovenstaande maxima voor achternamen gelden niet wanneer de Levenshteinafstand gelijk aan of onder deze limiet blijft
pmmaxlnxdlev	0.5	deze maximale relatieve Levenshteinafstand geldt altijd

Tabel B.3: Instellingen van Certilink

parameter	standaard	omschrijving
pplifconminmarage	15	de minimale huwelijksleeftijd
pplifconmaxmarage	115	de maximale huwelijksleeftijd
pplifconminbiragemot	15	de minimale leeftijd van de moeder bij geboorte van kind
pplifconmaxbiragemot	53	de maximale leeftijd van de moeder bij geboorte van kind
pplifconminbiragefat	15	de minimale leeftijd van de vader bij geboorte van kind
pplifconmaxbiragefat	100	de maximale leeftijd van de vader bij geboorte van kind
pplifconmaxmaragechi	100	de maximale leeftijd van een kind waarbij de ouders kunnen trouwen (effectief uitgeschakeld)
wpnfn	1	het gewicht van het verschil in voornamen
wpnpf	0	het gewicht van het verschil in prefices
wpnln	1	het gewicht van het verschil in achternamen
wptb	1	het gewicht van de tijdshulpmaat
wp	1	het gewicht van het verschil tussen persoonsknopen
wf	0	het gewicht van het verschil tussen gezinsknopen
testrun	false	*true* of *false*. Deze optie kan aangezet worden voor het testen van verschillende onderdelen van het programma. Dit is van nut bij debuggen.
analyzelargetables	false	*true* of *false*. Zet deze optie aan om Certilink de sleutels en indices van de grote tabellen te laten optimaliseren voordat queries naar de database worden gestuurd. Dit kost veel tijd, maar kan de performance effectief verbeteren.
bdagemot	"15,18,0.5", "19,22,0.2", "23,41,0", "42,46,0.2", "47,53,0.5"	Dit zijn de instellingen van tijdshulpmaat. Per string wordt de minimumleeftijd, de maximumleeftijd en de correctiewaarde gegeven voor de leeftijd waarop de moeder een kind kan krijgen.
bdagefat	"15,20,0.2", "20,75,0.0", "75,100,0.2"	Dit zijn de instellingen van tijdshulpmaat. Per string wordt de minimumleeftijd, de maximumleeftijd en de correctiewaarde gegeven voor de leeftijd waarop de vader een kind kan krijgen.
apppriority	4	de prioriteit van de applicatie voor MS-Windows
autoconnect	true	*true* of *false*. Als deze optie aanstaat verbindt Certilink automatisch met de database bij het opstarten van het programma.
autoreadtables	true	*true* of *false*. Als deze optie aanstaat leest Certilink automatisch de gegevens voor de tabellen in voor het overzicht op de *tables*-tab.

Tabel B.4: Instelling van Certilink, vervolg



Figuur B.7: Backup

to prefix: Vul hier een string in die wordt verwerkt in de backupnaam waaronder de tabellen worden weggeschreven. De nieuwe naam van de tabellen wordt: "clb" + prefix + oude naam. Met behulp van deze prefix kunnen de backups ook teruggehaald worden.

leave copy: Als deze optie aanstaat, worden de tabellen gekopieerd. Zo niet, dan worden bij het maken van backups de tabellen hernoemd. Dit is veel sneller, maar de oorspronkelijke tabellen zijn dan niet meer beschikbaar.

backup: Deze knop start het maken van de backups.

from prefix: Vul hier de prefix in van de tabellen die teruggezet moeten worden vanuit een backup. Dit is dezelfde string die is ingevuld bij het maken van de bewuste backuptabellen.

leave copy: Als deze optie uitstaat, worden de backtabellen teruggezet door de naam te veranderen. Dit is veel sneller, maar verwijdert effectief de backups na het terugzetten. Als deze optie aanstaat, worden de backups gekopieerd en blijven ze in stand.

restore: Met deze knop wordt het terugzetten gestart. De huidige tabellen worden overschreven.

merge: Ook met deze knop kan het terugzetten worden uitgevoerd. De gegevens van de backuptabellen overschrijven de huidige tabellen echter niet, maar worden er aan toegevoegd. De huidige tabellen worden effectief uitgebreid.

backups: Dit overzicht toont alle backups die aanwezig zijn op de database.

requery: Met deze knop kan het overzicht van de backups ververst worden.

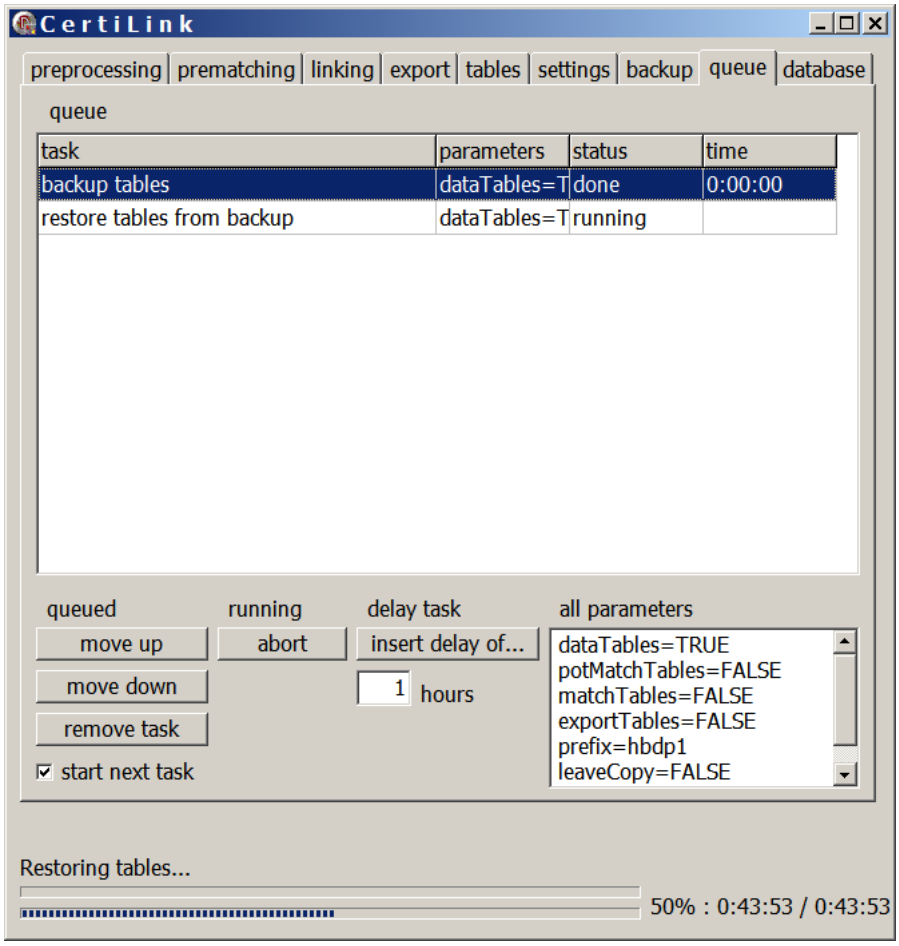
Queue

Omdat vele taken van Certilink lang duren, is het handig om meerdere taken na elkaar te plannen. Dit kan op de tab *queue* (zie figuur B.8 op de rechter pagina). Het systeem maakt onderscheid tussen grote taken en kleine taken. Kleine taken, zoals de *requery*-knoppen, kunnen niet gepland worden. Grote taken wel. Deze zijn herkenbaar aan knoppen waarvan het opschrift eindigt met drie puntjes ("..."). Wanneer op een dergelijke knop wordt gedrukt komt de taak automatisch in deze rij en wordt uitgevoerd. Als tijdens de uitvoering van de taak weer een grote taak wordt gestart, komt deze in de wachtrij. Deze wachtrij kan hier gemanipuleerd worden:

queue: In dit overzicht staan de taken die uitgevoerd zijn, de taak die nu uitgevoerd wordt en de taken die nog uitgevoerd moeten worden.

- **task:** de naam of omschrijving van de taak
- **parameters:** eventuele parameters van de taak (Deze zijn een resultaat van keuzes die de gebruiker heeft gemaakt bij het starten van de taak.)
- **status:** de status van het uitvoeren van de taak
- **time:** de tijd die verlopen is tijdens het uitvoeren van de taak

all parameters: In het *parameters*-veld in het overzicht passen vaak niet alle parameters. In het *all parameters*-vak zijn de parameters van de geselecteerde taak zichtbaar.



Figuur B.8: Queue

move up: Hiermee kan een taak eerder in de rij worden gezet. Een taak kan alleen omhooggeschoven worden als hij nog de status "queued" heeft. Hij kan niet eerder gepland worden dan een lopende taak.

move down: Hiermee kan een taak met de status "queued" later in de rij worden gezet.

remove task: Met deze knop kan een geselecteerde taak met de status "queued" worden verwijderd.

abort: Wanneer nodig kan een lopende taak worden afgebroken. Hierdoor kan echter werk half uitgevoerd op de database blijven staan. Ook kan het erg lang duren voordat een query op de database afgebroken kan worden.

insert delay task: Met deze knop wordt een lege taak in de rij gezet die een instelbaar aantal uren duurt. Tijdens het uitvoeren van deze taak doet Certilink niets. Het kan in bepaalde gevallen nuttig zijn om te beginnen met een *delay task* zodat Certilink pas op een bepaalde tijd begint met zijn taken.

start next task: Wanneer een taak is uitgevoerd begint Certilink automatisch aan de volgende taak, behalve wanneer deze optie uitstaat. Ook met de eerste taak wordt niet gestart zolang deze optie niet is aangevinkt. Om rustig een aantal taken te plannen kan de gebruiker de optie uitvinken en de taken in de rij zetten. Wanneer de taakwachtrij naar wens is ingesteld kan *start next task* aangezet worden en begint Certilink onmiddelijk met de eerste wachtende taak in de rij.

Database

De laatste tab van het programma regelt de verbinding met de database. Voordat deze gemaakt kan worden moet er op de computer waarop Certilink draait, of in het Windows-account waaronder de gebruiker is ingelogd, een ODBC-koppeling naar de database beschikbaar zijn. Deze kan het best met behulp van de beheerder van de databaseserver worden aangemaakt[1]. De vereiste gegevens voor het verbinden met de database via ODBC kunnen hier ingevuld worden:

ODBC data source name: Hier kan de DSN, de naam van de ODBC-koppeling, worden ingevuld.

database name: Vul hier de naam van de database op de server in.

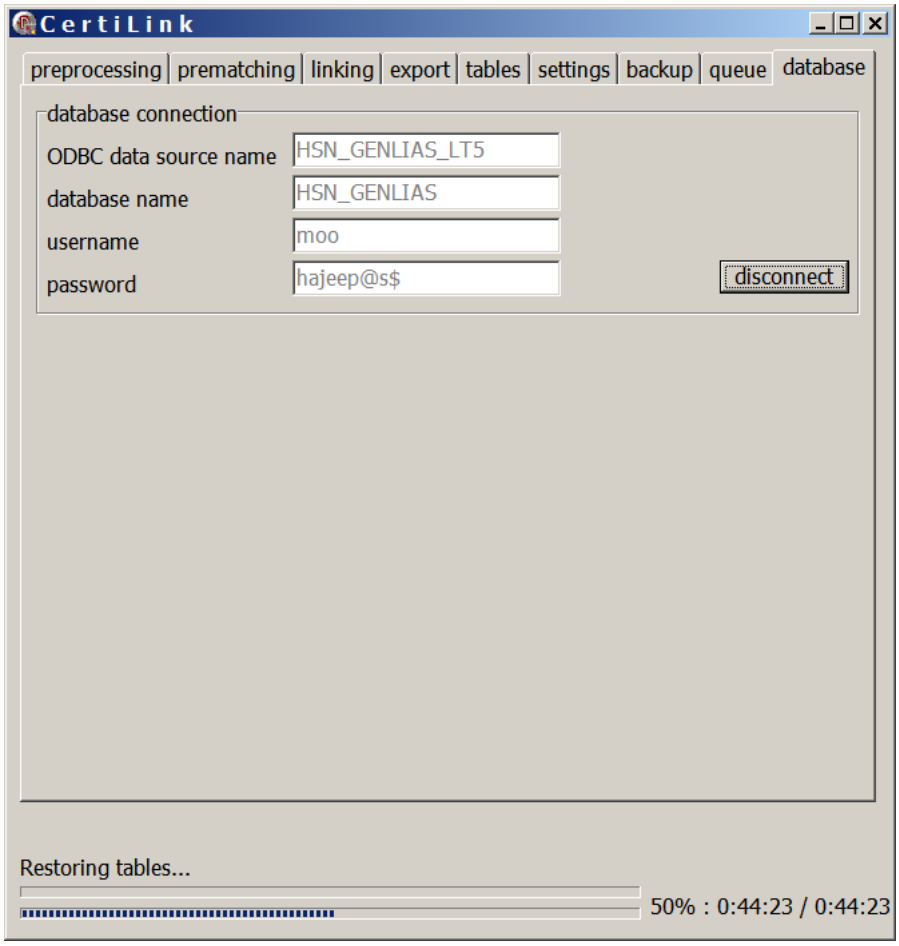
username: De gebruikersnaam voor de database kan hier worden ingevoerd.

password: Het wachtwoord voor de database kan hier.

connect / disconnect: Deze knop maakt of verbreekt de verbinding met de database.

Het invullen van de velden op deze tab komt overeen met het wijzigen van de instellingen *dbaliasname*, *dbdatabasename*, *dbusername* en *dbpassword*. Wanneer bij *settings* op *save* wordt gedrukt, worden deze inlog-gegevens opgeslagen in `settings.dat`.

[1] Deze koppeling kan gemaakt worden in Windows XP onder Start → Control Panel → Administrative Tools → Data Sources (ODBC). Hiervoor moet de ODBC-driver van MySQL geïnstalleerd zijn.



Figuur B.9: Database

Voortgang

Tot slot is onderaan de window van de applicatie de voortgang van de huidige taak zichtbaar. De onderste voortgangsbalk geeft de progressie aan van de gehele taak en de bovenste van een eventuele subtaak. Daar weer boven staat een korte omschrijving van de huidige activiteit. Naast de voortgangsbalken staat van links naar rechts het geschatte percentage van de taak dat volbracht is, de tijd die verstreken is tijdens de uitvoering van de taak en de geschatte resterende tijd. Wanneer de muisaanwijzer op het percentage geplaatst wordt, verschijnt het percentage met meer cijfers achter de komma. Dit kan van nut zijn om bij de grotere taken te controleren of er nog wel voortgang plaatsvindt. Wanneer we dit percentage langzaam zien toenemen weten we dat Certilink zich onvermoeibaar door het werk heenslaat.

Bibliografie

[L66] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 1966

[CL01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest en Clifford Stein, Introduction to Algorithms, 2nd edition, *MIT Press and McGraw-Hill*, 2001

[KO07a] K. Mandemakers en M. Oosten, Linking with the Dutch Genlias index of marriage certificates, 7th European Social Science History Conference, 26 februari – 1 maart 2007, Lissabon

[KO07b] K. Mandemakers en M. Oosten, Intergenerational linking of 5.000.000 marriage records from the Netherlands, 1812 – 1922, Workshop The Next Generation of Record Linkage from Systematic Sources, 5 – 6 maart 2007, University of Guelph, Canada

[RG02] Ragu Ramakrishnan en Johannes Gehrke, Database Management Systems, 3rd edition, *McGraw-Hill*, 2002