Universiteit Hasselt

Faculteit Wetenschappen

GOOGLE PAGERANK

Wiskundig Modelleren

Luca Pignatelli
Sam Beunckens
Jolien Drijkoningen
Tuur Willio

Contents

1 Inleiding **2**

2 Het probleem **2**
2.1 Hoe werkt Google 2
2.2 Het ordenen van webpagina's 3
2.3 Een website optimaliseren 3
2.4 De Google PageRank 4

3 Het wiskundig model **5**
3.1 Simplificaties 5
3.2 Het internet als model 5
3.3 De opbouw van het model 6
3.4 Het wiskundig model 8
3.4.1 Existentie en eenduidigheid 10
3.5 Gevoelig voor manipulatie 11
3.6 Andere toepassingen 11

4 Oplossen van het model **13**
4.1 Waarom de iteratieve methode? 13

5 Matlab functies **14**

6 Variabelen **16**
6.1 Dimensie Connecties 16
6.2 Voorspellen van gemiddeld aantal iteraties 20
6.3 $\mathbf{p}^{(0)}$ kiezen 22

7 Besluit **24**

8 Bijlage **27**
8.1 Matlab Functies 27

1 Inleiding

In dit project zullen we bestuderen hoe Google zijn websites ordend aan de hand van de zoekterm. We houden ons vooral bezig met de eerste PageRank methode, ontwikkeld door de oprichters van Google (Larry Page en Sergey Brin).

We stellen een wiskundig model op en bestuderen de invloed van verschillende variabelen op het aantal iteraties. We onderzoeken dit om te achterhalen hoe Google zo snel een PageRank kan genereren.

2 Het probleem

Er bestaan in Google zo een 60 miljard websites. Toch slaagt Google erin als een gebruiker een zoekopdracht invoert om zeer snel veel gepaste zoekresultaten terug te geven. Dit verslag geeft via een wiskundig model een antwoord op de vraag hoe Google dit allemaal kan doen op zo een snelle manier.

2.1 Hoe werkt Google

De doelen van Google en andere zoekmachines zijn om een gedetailleerde analytische index te maken van alle webpagina's, deze bijgewerkt te houden en snel te kunnen zoeken naar informatie relevant aan de gegeven zoekopdracht door de gebruiker[12]. Dit alles kan Google doen door gebruik te maken van Googlebots [9]. Deze gaan nieuwe en bijgewerkte pagina's scannen en aan zijn index toevoegen. Deze nieuwe informatie wordt gebruikt om de index te updaten en dus te verbeteren voor de gebruiker. De Googlebot gaat vaak deze pagina's opnieuw scannen, dit doet hij door de hele pagina's te laden zoals de gebruiker hem zien. Dit kan belastend zijn voor de server en daarom kan de serverhost ervoor kiezen om dit niet of minder vaak te doen. Dit is echter niet interessant voor commerciële webpagina's omdat zij juist wel opgemerkt willen worden door Google om zo veel mogelijk aanbevolen te worden bij de gebruikers. Het kan wel interessant zijn voor webpagina's die alleen gebruikt worden binnen beperkte kring.

2.2 Het ordenen van webpagina's

Bij het ordenen van alle webpagina's houdt Google rekening met ongeveer 200 factoren waaronder bijvoorbeeld: de locatie, het apparaat waarvan de zoekopdracht wordt uitgevoerd, de taal, etc.

2.3 Een website optimaliseren

Gebruikers kunnen hun website op verschillende manieren optimaliseren. Dit kan onder andere op volgende manieren [13]:

- Door de **kernwoorden**(belangrijke woorden waarover de website gaat) van hun website op een goede plaats te plaatsen, bijvoorbeeld: in titels en tussentitels, Google titel, in de metabeschrijving (zie figuur 1)

Figure 1: Google titel en metabeschrijving

- Voor de **metabeschrijving** is het belangrijk om deze aantrekkelijk te maken voor de gebruikers zodat zij meer geneigd zijn om de website te bezoeken.

- Door de **content geüpdatet** te houden bezoekt de Googlebot de website vaker, past deze vaker de score van de webpagina aan en is er dus een grotere kans dat de website meer aanbevolen zal worden.

- Door **links** te voegen naar andere webpagina's en ervoor te zorgen dat andere webpagina's naar deze webpagina links toe voegen zal de website ook meer aanbevolen worden. Verder in dit verslag wordt duidelijk waarom dit helpt.

- Door de **alt tags** (zie figuur 2) goed te kiezen en ze relevante info mee te geven wordt de website door Google beter aanbevolen als de zoekopdracht te maken heeft met de informatie in de alt tags.

SLECHTE ALT TAG:

**

Hier beschrijft de alt tag niks.

BETERE ALT TAG:

**

Hier beschrijft de alt tag een klein beetje waar de afbeelding over gaat.

Figure 2: Alt tags

2.4 De Google PageRank

Google gaat met behulp van wat deze weet een waarde geven aan de pagina die zijn relevantie voor de zoekterm uitdruk. Dit wordt dan de kans waarmee een gebruiker op een website terecht komt. Omdat Google veel websites bevat en de index vaak geüpdatet moet worden is hier veel werkgeheugen voor nodig en dus ook veel sterke computers.

3 Het wiskundig model

3.1 Simplificaties

Zoals eerder besproken houdt Google rekening met 200 factoren om zijn pagina's te orderenen. Voor het wiskundig model worden enkele simplificaties gemaakt. We beschouwen het internet als een bibliotheek van websites die bestaan uit simpele tekst, zonder andere mediabestanden zoals foto's of video's. Droge tekst met referenties naar andere websites (zie afbeelding) die we zullen bespreken als connecties.

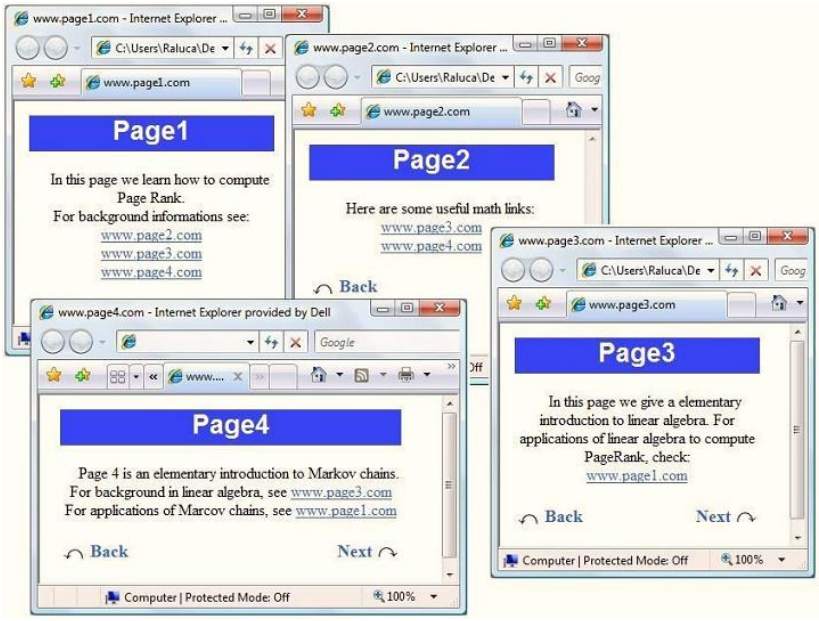


Figure 3: Voorstelling van het 'simpelere' internet (*Cornell University*)

3.2 Het internet als model

Elke website die we gaan beschouwen zal dus connecties hebben met andere websites. Zoals men in het vorige hoofdstuk zag dat zoeken met keywords niet optimaal is en makkelijk te manipuleren is, gaan we dus sorteren aan de hand van connecties. Veel connecties die leiden naar een bepaalde website als referentie, zou impliceren dat deze website een goede bron is en dus een hoge PageRank zou moeten hebben, een hogere plaats bij de zoekresultaten.

We kunnen hiermee alle sites bekijken naar hoeveel connecties ze hebben en naar welke andere sites.

3.3 De opbouw van het model

Om het model op te stellen beschouwen we een klein en simpel 'Internet' bestaande uit 4 websites zoals in onderstaande afbeelding. We kunnen dit

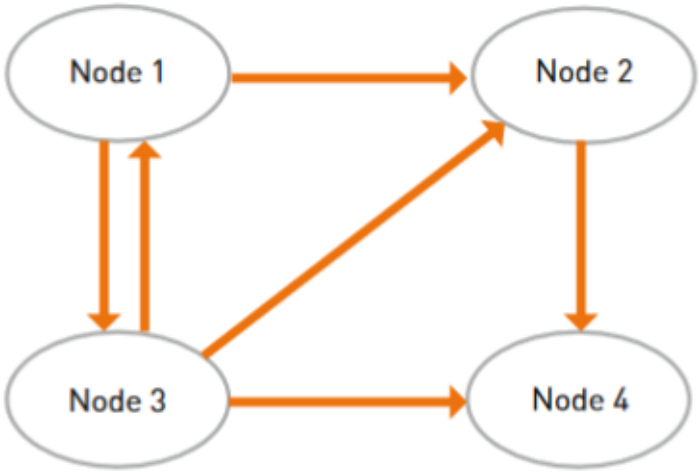


Figure 4: Klein netwerk van 4 websites (*A Primer on Mathematical Modelling*)

makkelijker en mathematischer voorstellen aan de hand van een matrix

$$A = \begin{array}{c} \begin{matrix} \text{V} & \text{A} & \text{N} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \\ \text{N} \\ \text{A} \\ \text{A} \\ \text{R} \end{matrix}$$

Waarbij een 1 een connectie voorstelt van de kolom website naar de rij website. Een 0 stelt dan respectievelijk geen connectie voor.
Zoals we in het vorige hoofdstuk al zagen, willen we een systeem dat een orde maakt op basis van de kans dat een willekeurige gebruiker op een site belandt.

We kunnen dus met vorige A matrix een nieuwe matrix opstellen

$$B = \begin{bmatrix} 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/3 & 0 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1 & 1/3 & 0 \end{bmatrix}$$

Deze matrix stelt dus de kansen voor dat een gebruiker van de kolom website naar een rij website gaat. Bijvoorbeeld de kans dat een gebruiker van website 3 naar 2 gaat is 1/3.
We zien ook duidelijk dat 4 geen uitgaande links heeft, en dus geen connecties heeft en dus overal 0 kansen heeft op transportatie. Dit wordt ook wel een *black hole* genoemd, omdat eens je bij website 4 bent kan je nergens meer naar toe via de referenties. En dat is het effect dat we proberen na te recreëren met matrices: willekeurige verplaatsing op het web aan de hand van uitgaande links van de ene website naar de andere. Maar website 4 zou heel dit proces een beetje plat leggen daarom moeten onze matrix B een beetje aanpassen zodat men wel van website 4 naar een andere website kan. Dit doen we op de eerlijkste methode: een kans van $1/n$ of in dit geval 1/4 naar elke website.

$$S = \begin{bmatrix} 0 & 0 & 1/3 & 1/4 \\ 1/2 & 0 & 1/3 & 1/4 \\ 1/2 & 0 & 0 & 1/4 \\ 0 & 1 & 1/3 & 1/4 \end{bmatrix}$$

Nu lijkt of we alle problemen hebben opgelost en we willekeurig gaan kunnen ronddwalen met onze matrix S, maar er schuilt nog een probleem dat hier niet zichtbaar is. Als we grotere netwerken gaan beschouwen kunnen we te maken krijgen met *closed subsets* waarbij, zoals de naam al weergeeft, eens je er in deze subset komt je er niet meer uit geraakt. Dit wordt weergeven in de onderstaande afbeelding, de gesloten subset is omcirkeld met rood. Dit zou dus niet perfect zijn voor willekeurige transportatie. Dus we introduceren een laatste aanpassing dat ons een globale transportatie zal geven. We voegen een willekeurigheidsfactor α toe in volgende formule voor onze nieuwe matrix G, met $N = 4$ in dit geval

$$g_{ij} = \alpha s_{ij} + \frac{1-\alpha}{N}$$

Het is ook duidelijk in de formule dat als $\alpha = 0$ we niet meer kijken naar onze vorige matrix S maar we een compleet willekeurige transportatie matrix opstellen waarbij alle kansen $1/n$ zijn. Als α 1 is voegen we geen willekeurigheid

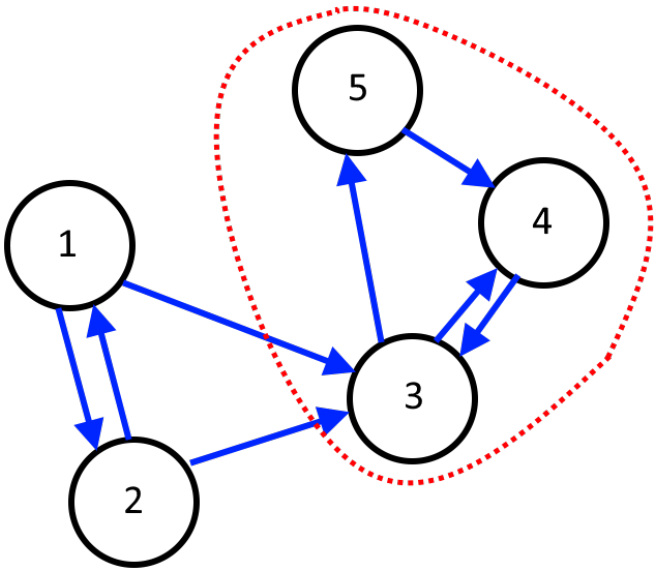


Figure 5: Voorbeeld closed subset

toe en behouden we onze matrix S zodat $G = S$. Uit bronnen [12] weten we dat Google een α van 0.85 gebruikt. Als we deze toepassen op onze matrix S, krijgen we onze finale matrix G

$$G = \begin{bmatrix} 0.0375 & 0.0375 & 0.3208 & 0.2500 \\ 0.4625 & 0.0375 & 0.3208 & 0.2500 \\ 0.4625 & 0.0375 & 0.0375 & 0.2500 \\ 0.0375 & 0.8875 & 0.3208 & 0.2500 \end{bmatrix}$$

3.4 Het wiskundig model

We hebben nu een kansen matrix G die ons de kansen geeft op transportatie van de ene website naar de andere, g_{ij} is de kans om van website j naar i te gaan. Zo is ook de kans om van eender welke site naar ene site i te gaan ook de som van deze termen $\sum_{k=1}^{N} g_{ik}$. Dit is allemaal wel maar voor 1 stap te doen, 1 keer klikken naar een andere website. Als we dit willen blijven doen moeten we eerst een startpositie definiëren. Ook opnieuw op de eerlijkste

manier die we kennen, eerlijke gelijke kans voor alle websites een vector P_0

$$P_0 = \begin{bmatrix} 1/n \\ 1/n \\ \vdots \\ 1/n \end{bmatrix}$$

Dit is zijn dus de kansen $p_i = P_0(i)$ dat we op een bepaalde website i zitten met $0 \leq i \leq N$. Uiteindelijk willen we hiervan onze PageRank bepalen in het oneindige, omdat we kan dan echt zien waar we nu naar toe drijven op het Internet moesten we blijven zoeken.
Nu de kans we opnieuw naar een website i gaan is dus nu niet zomaar de sommatie van g_{ij} maar nu voegen we een extra factor toe van onze vector P_0.

$$P_1(i) = \sum_{k=1}^{N} g_{ik} P_0(k)$$

De kans om op website i te geraken na 1 stap is dus p_i de som van kansen dat je op een website k, $P_0(k)$ zat en naar website i gaat, g_{ik}. Zo kunnen we een hele nieuwe vector opstellen, P_1 dus.

$$P_1 = \begin{cases} p_1 = g_{11}p_1 + g_{12}p_2 + \ldots + g_{1N}p_N \\ p_2 = g_{21}p_1 + g_{22}p_2 + \ldots + g_{2N}p_N \\ \ldots \\ p_N = g_{N1}p_1 + g_{N2}p_2 + \ldots + g_{NN}p_N \end{cases}$$

Dit ziet er natuurlijk uit als matrix multiplicatie. We krijgen dan dus:

$$P_1 = G * P_0$$

We kunnen dus voor elke stap onze kansen vector P vermenigvuldigen met onze transportatie matrix G. En om dus een oplossing te zoeken zouden we dus dit $n \to \infty$ moeten doen.

$$P_n = G * P_{n-1} = G * (G * P_{n-2}) = G^n * P_0$$

tot we krijgen:

$$P = G * P$$

We zullen later zien dat we dit best iteratief oplossen.

3.4.1 Existentie en eenduidigheid

We hebben nu een methode, maar dit heeft dit wel een oplossing, en is deze oplossing uniek? Het korte antwoord is ja, maar hoe werkt dit dan.
Ten eerste moeten we inzien dat een oplossing van ons model $P = GP$ eigenlijk de zoektocht is naar een eigenvector van matrix G met eigenwaarde 1. Ten tweede kunnen we zeggen dat onze matrix G een kolom-stochastische matrix is omdat de sommen van de kolommen gelijk zijn aan 1 en omdat de elementen tussen 0 en 1 liggen. Dit komt door de constructie van G. Rij-stochastische matrices hebben altijd een eigenwaarde 1. Dit is makkelijk in te zien, want stel we hebben een rij stochastische matrix W zodat $\sum_{k=1}^{n} w_{ik} = 1$ dan geldt:

$$W * \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

We hebben dus zeker een eigenwaarde 1 omdat we hiervoor een eigenvector hebben gevonden namelijk $[1, 1, 1, \cdots, 1]^T$ We kunnen ook aantonen dat eigenwaarde hetzelfde blijven als we een matrix transponeren, of met andere woorden een rij stochastische matrix wordt een kolom stochastische matrix.

$$\begin{aligned} p_{A^T}(\lambda) &= det(A^T - \lambda I_d) \\ &= det(A^T - \lambda I_d^T) \\ &= det((A - \lambda I_d)^T) \\ &= det(A - \lambda I_d) \\ &= p_A(\lambda) \end{aligned}$$

Dit geldt omdat $I_d = I_d^T$ en determinanten van getransponeerde blijven hetzelfde.
We weten nu dus dat onze kolom-stochastische matrix G een eigenwaarde 1 bevat en dus een oplossing heeft voor $P = GP$.

We kunnen ook zeggen dat onze matrix G een irreduciebele matrix is omdat elke toestand bereikbaar is na een eindig aantal stappen. Sterker zelfs met $\alpha = 0.85$ weten we dat elke toestand na 1 stap bereikbaar is omdat door onze willekeurigheidsfactor α we geen 0-elementen hebben in G. We weten ook

dat de spectraalstraal van onze matrix G, weeral omdat deze stochastisch is, 1 is. Dit kan bewezen worden met de *Gershgorin circle theorem*[2]. Volgens de Perron-Frobenius stelling[1] geldt dan dat we een unieke stationaire verdeling kunnen vinden zodat $P = GP$.

3.5 Gevoelig voor manipulatie

De PageRank is gevoelig voor manipulatie, omdat deze afhangt van links van webpagina's naar andere webpagina's. Hier worden twee mogelijkheden besproken om de PageRank te verhogen.
Ten eerste is er link building is het laten genereren van links van externe webpagina's naar een specifieke website met doel om de PageRank te verbeteren [6]. Interne en externe linkbuilding. Interne linkbuilding is van de ene naar de andere pagina op de website en externe linkbuilding zijn andere websites die naar die ene webpagina verwijzen. Er bestaan ook websites waar abonnementen op gekocht kunnen worden die dan artikels plaatsen met links die verwijzen naar jouw website.
Dan is er link farm is een groep websites die allemaal links naar elkaar sturen. Belangrijk bij linkfarming is dat een website links moet ontvangen en versturen om voor de zoekmachines relevant te blijven [7]. Maar ook hier zijn er manieren om dit te manipuleren. Websites die inkomende links ontvangen, maar dan manieren vonden om hun uitgaande links te verbergen of zelfs geen links te plaatsen. Er waren ook websites met alleen links naar andere pagina's. Maar Google heeft tegenwoordig verschillende manieren om dit soort manipulatie te herkennen door onder andere de bron na te kijken van waar de link komt.

3.6 Andere toepassingen

Het algoritme van de Google Pagerank of gelijkaardige algoritmes worden ook in andere toepassingen gebruikt [4]. Zo wordt de PageRank gebruikt onder andere in de wetenschap, websites en in de sport.
In de wetenschap wordt de PageRank gebruikt om de wetenschappelijke impact van onderzoekers te kwantificeren. De citaties en samenwerkingen tussen de wetenschappers worden gebruikt in combinatie met het PageRank algoritme om een classificatiesysteem voor individuele publicaties te bekomen voor individuele auteurs.
Voor de impactfactor van het Institute for Scientific Information en is er een

vervanging voorgesteld. Hierbij worden niet enkel het aantal citaties geteld maar ook het belang van de citaties.
Twitter gebruikt een gepersonaliseerde PageRank om gebruikers andere accounts voor te stellen die ze eventueel willen volgen.
In de sport wordt het PageRank algoritme gebruikt voor het rangschikken van: Teams in de National Football League in de VS, individuele voetballers en atleten te ranken.

4 Oplossen van het model

4.1 Waarom de iteratieve methode?

In dit hoofdstuk zullen we bespreken waarom we verkiezen de iteratieve methode te gebruiken. Beschouw het volgende voorbeeld:
We beschouwen de matrix S waaruit we de google matrix G construeren:

$$S = \begin{bmatrix} 0 & 0 & 1/3 & 1/4 \\ 1/2 & 0 & 1/3 & 1/4 \\ 1/2 & 0 & 0 & 1/4 \\ 0 & 1 & 1/3 & 1/4 \end{bmatrix}; G = \begin{bmatrix} 0.0375 & 0.0375 & 0.3208 & 0.2500 \\ 0.4625 & 0.0375 & 0.3208 & 0.2500 \\ 0.4625 & 0.0375 & 0.0375 & 0.2500 \\ 0.0375 & 0.8875 & 0.3208 & 0.2500 \end{bmatrix}$$

Als we de pagerank beginwaarde $p^{(0)}$ kiezen als :$\begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix}^T$ zal $Gp^{(0)}$ gelijk zijn aan $\begin{bmatrix} 0.161 & 0.267 & 0.196 & 0.374 \end{bmatrix}^T = p^{(1)}$
$p^{(1)}$ kunnen we interpreteren als de kans om op een site te belanden na 1 tussenstap. De algemene formule voor de kans om op een site te belanden na k stappen is dan:

$$p^{(k)} = Gp^{(k-1)} \tag{1}$$

Nu willen we het gedrag bestuderen als k naar oneindig gaat, we bekijken de berekening tot de nde pagerank, en versimpelen de uitdrukkingen:
$\mathbf{p}^{(1)} = G\mathbf{p}^{(0)}$
$\mathbf{p}^{(2)} = G\mathbf{p}^{(1)} = GG\mathbf{p}^{(0)} = G^2\mathbf{p}^{(0)}$

$\vdots$
$\mathbf{p}^{(n)} = G^n\mathbf{p}^{(0)}$
We merken op, G is diagonaliseerbaar en dus zal G^n ontbinden in VD^nV^{-1}. D^n is de matrix met eigenwaarde tot de nde macht op de hoofddiagonaal. Door dan deze matrix naar plus oneindig te laten gaan is het mogelijk de analytische oplossing van het probleem te vinden. Echter is dit niet altijd praktisch. Het gaat hier meestal over gigantische matrices en dan zal diagonalisering vaak niet praktisch of computationeel haalbaar zijn. Daarom verkiezen we de iteratieve methode, deze wordt uitgelegd in het volgende hoofdstuk omschreven als:'de pagerank functie'. Dit algoritme is niet alleen praktischer te implementeren maar zal ook computationeel veel eenvoudiger zijn om uit te voeren, en geeft ons vaak een nagenoeg perfect resultaat. Dit komt omdat de pagerank kansen niet ertoe doen, maar de volgorde als we ze sorteren van klein naar groot.

5 Matlab functies

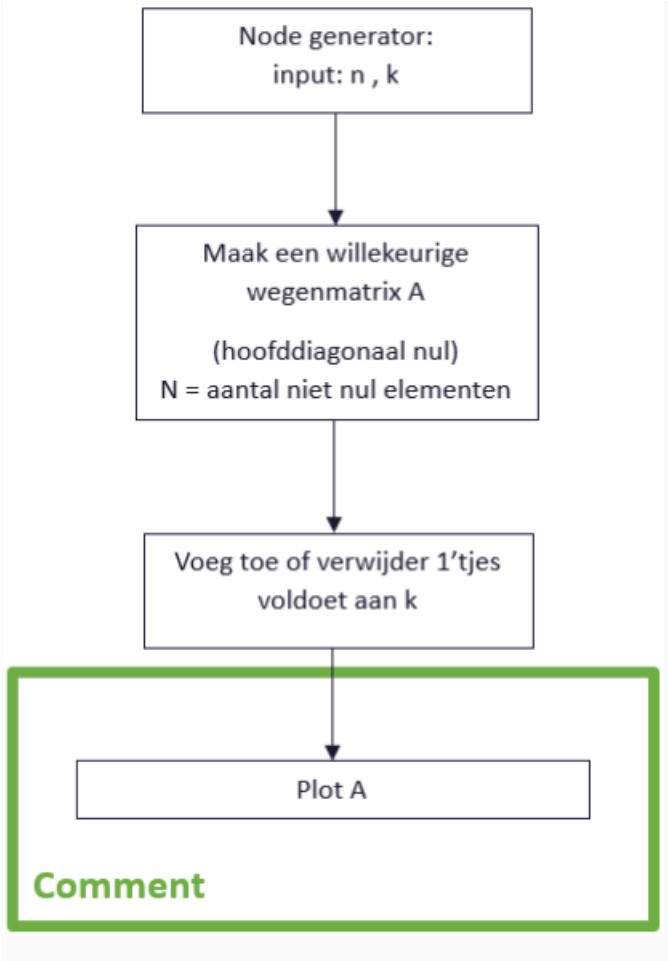


Figure 6: Dit is de flowchart voorstelling van de Node generator functie. De functie gebruikt als input n: de dimensie van de matrix (het aantal website in onze matrix aanwezig) en k: het aantal connecties (links) van webpagina's naar andere webpagina's die in het totaal aanwezig zijn. Dan worden er in de matrix A willekeurig eentjes of nullen toegevoegd of verwijderd tot dat het aantal eentjes in de matrix gelijk is aan k. Dan geeft de Node generator functie de matrix A terug

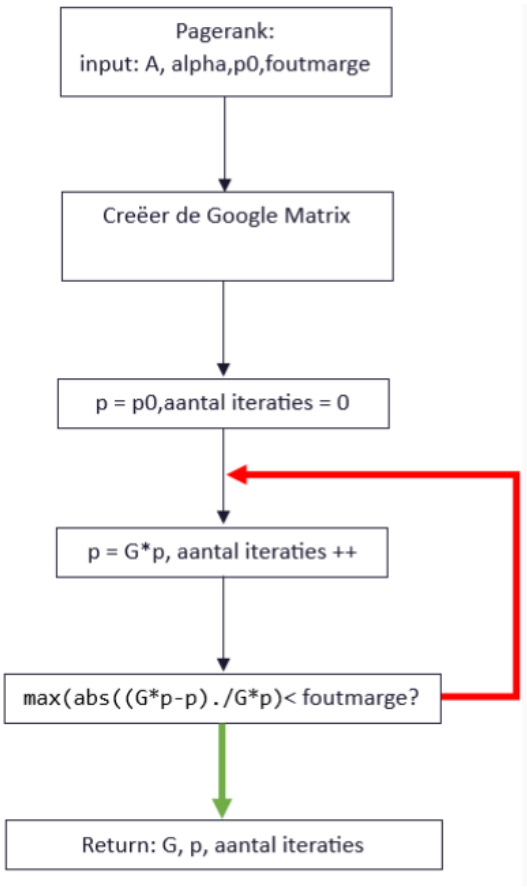


Figure 7: Dit is een flowchart voorstelling van de Pagerank functie. De functie gebruikt als input de matrix A, hierboven geconstrueerd, de factor alpha die aanduidt hoe willekeurig een surfer over het internet beweegt (door Google gekozen als 0.85), de p0 vector met beginwaardes voor de toekomstige PageRank vector en als laatste de foutmarge voor de stoptest. De functie maakt eerst een Google matrix met behulp van de matrix A Dit gebeurt met de elementen van de matrix G $g_{ij} = \alpha S_{ij} + \frac{1-\alpha}{N}$, waarbij S ook eerst geconstrueerd wordt zoals eerder in dit verslag vermeld. Dan wordt de PageRank vector p gelijkgesteld aan de beginvector p0. Dan wordt G vermenigvuldigt met p om zo de nieuwe p-vector te bekomen, elke keer dit proces gebeurt wordt er één iteratie bijgeteld. Dit proces herhaalt zich tot de fout via de formule in de flowchart kleiner is dan de opgegeven foutmarge. De functie geeft uiteindelijk de matrix G, de vector p en het aantal iteraties terug.

6 Variabelen

6.1 Dimensie Connecties

Voor de komende delen gaan we vaak gebruik maken van volgende data:

```
> summary(dataa2)
    Dimensie    Connecties     Iteraties
 Min.   : 2   Min.   :   1   Min.   : 0.000
 1st Qu.:57   1st Qu.: 863   1st Qu.: 3.000
 Median :72   Median :2026   Median : 4.000
 Mean   :68   Mean   :2430   Mean   : 4.837
 3rd Qu.:82   3rd Qu.:3676   3rd Qu.: 5.000
 Max.   :90   Max.   :8010   Max.   :50.000
> head(dataa2)
  Dimensie Connecties Iteraties
1        2          1        13
2        2          2         0
3        3          1         8
4        3          2         8
5        3          3        50
6        3          4         9
```

Een data set bestaande uit 242.970 rijen waarbij we voor elke dimensie, van 2 tot 90, alle connecties, van 1 tot $n^2 - n$, 10 maal testen en daar het gemiddeld aantal iteraties van nemen. Deze data is verkregen door een reeks *nested* `for`-loops met de functies in de bijlage.
We kunnen een groot deel aflezen van een heatmap gemaakt van al deze data.

Het witte gedeelte in de figuur is er omdat daar geen data is, een vierkante matrix van dimensie 4 kan moeilijk 4000 connecties bevatten. We kennen het voorschrift van deze grens omdat we onze matrices zo hebben opgesteld dat de connecties tot $dim^2 - dim$ gaan.
We kunnen nu proberen de andere kleurgrenzen te benaderen met 2de graads functies. Uit de data kunnen we de exacte grens tussen kleuren en dus iteraties halen en kunnen deze benaderen met 2de graads veelterm regressie die gebruikt maakt van data tot dimensie 50.

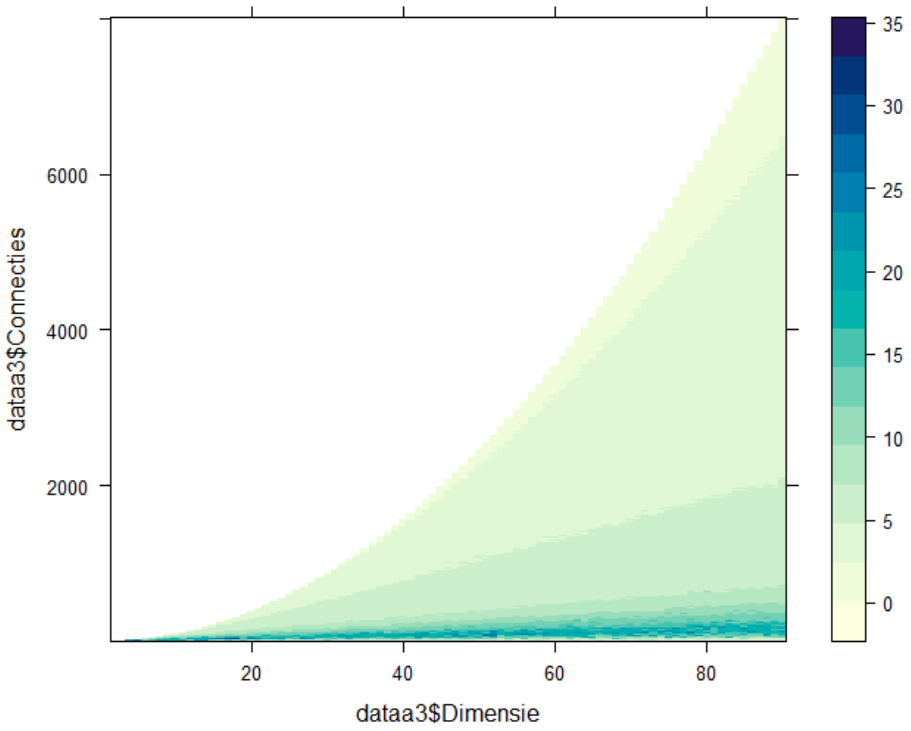
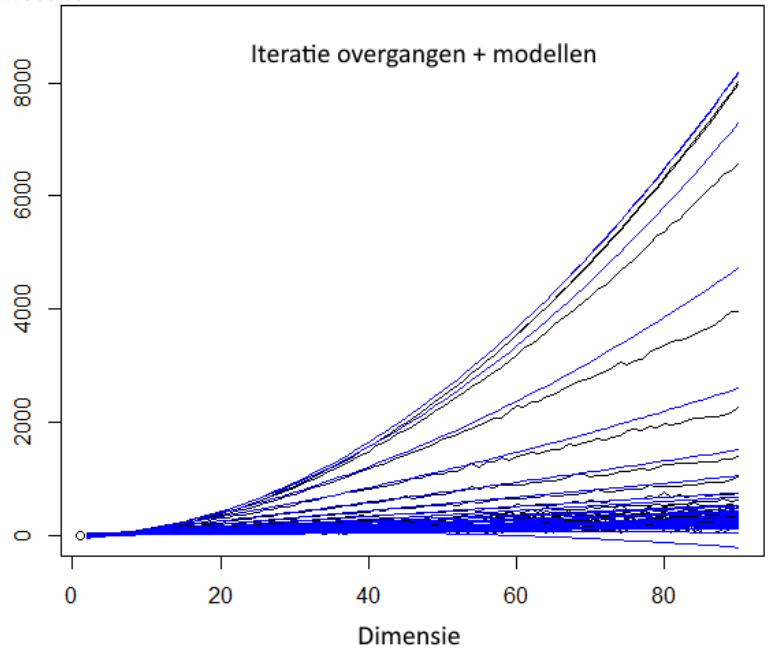


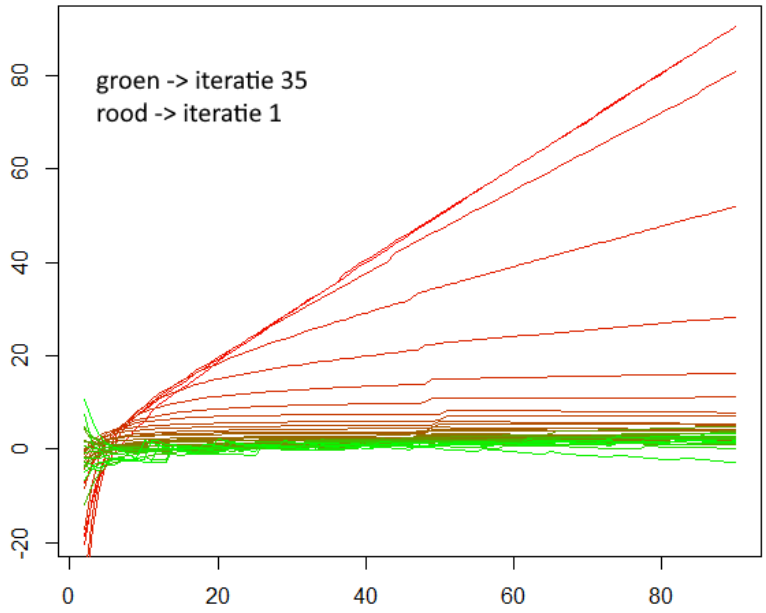
Figure 8: Heatmap: Deze toont het aantal dimensies op de x-as en het aantal connecties op de y-as. De kleur stelt aantal iteraties voor die nodig zijn om de PageRank te krijgen

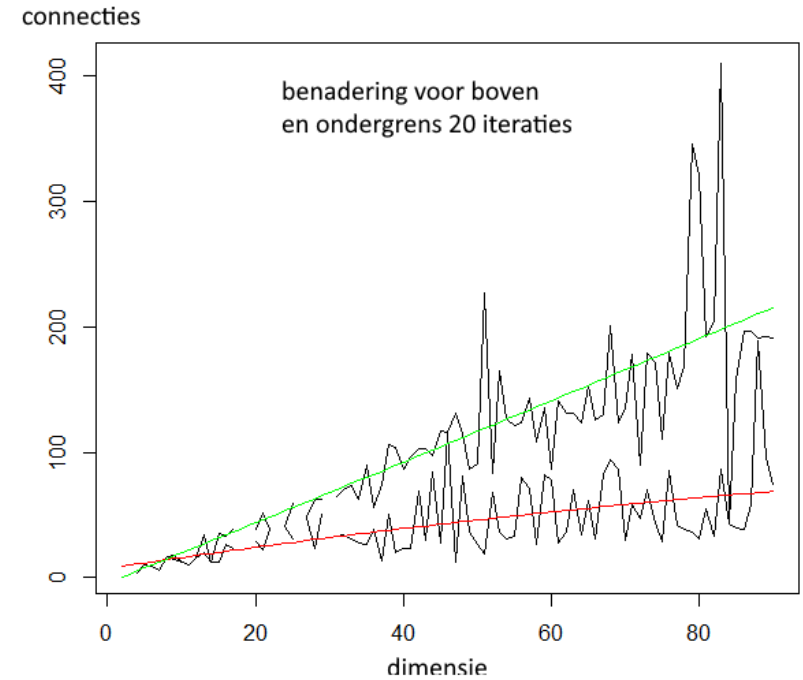
We zien duidelijk de contour van de heatmap terug. We zien wel dat bij lagere iteratie grenzen (rood) de fout groter is. We kunnen wel misschien een goed beeld geven van wanneer er een groot aantal iteraties gaat gebeuren bij een bepaalde dimensie

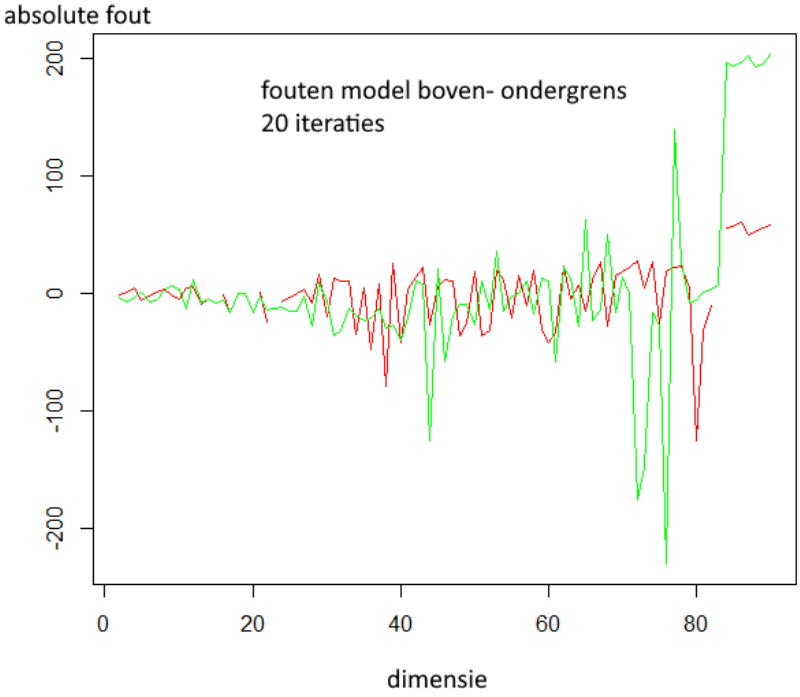
Connecties



fout/dimensie







We zien dat de fouten van de grens op 20 iteraties oploopt tot een 200. Dit

lijkt veel maar is nog redelijk als je weet dat rond dimensie 80 er $80^2 - 80 = 6320$ connecties beschikbaar zijn. We kunnen de bovengrens afschatten door een $200/80 = 2.5$ spelling te geven per dimensie. Dus voor onze 2de graads model voor de bovengrens doen we simpelweg $ax^2 + bx + c + 2.5x$. Nu kunnen we voor elke dimensie een regio geven waarbij deze voor dat aantal connecties maximaal, meer dan 20 iteraties, gaat itereren. Te vermijden dus! Ondergrens in functie van dimensie:

$$7.34495882 + 0.88463824x - 0.00224152x^2$$

Bovengrens in functie van dimensie:

$$-4.1952391 + 0.4.879916 - 0.0006927x^2$$

6.2 Voorspellen van gemiddeld aantal iteraties

Nadat we de verschillende zones hebben proberen te bepalen bekijken we of het mogelijk is om een schatting te maken over het aantal nodige iteraties tot convergentie. We zullen hiervoor poisson regressie gebruiken, aangezien we te maken hebben met een telling. Ook zullen we testen op een interactie term, we hebben namelijk al laten zien dat op een verschillend aantal dimensies, de verandering van het aantal connecties een andere invloed heeft. Het model ziet er als volgt uit.

$$\begin{aligned}
&\#iteraties\ tot\ convergentie\ = Y \\
&\#connecties = X_1 \\
&dimensies = X_2 \\
&log(\mathbb{E}(Y)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2
\end{aligned}$$

$$\begin{aligned}
&\beta_0\ = \text{intercept} \\
&\beta_1 = \text{stijging gemiddelde iteraties door connecties} \\
&\beta_2 = \text{stijging gemiddelde iteraties door dimensies} \\
&\beta_3 = \text{interactie term die tussen connecties en dimensies}
\end{aligned}$$

Deze test voeren we uit in Rstudio met de functie glm. We verkrijgen de volgende output:

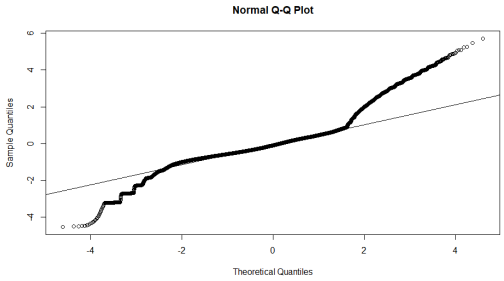
```
Call:
glm(formula = Iteraties ~ Connecties + Dimensie + Dimensie:Connecties,
    family = poisson(link = log), data = data_google)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.0547  -0.4243  -0.0653   0.3259   3.6838

Coefficients:
                     Estimate Std. Error  z value Pr(>|z|)
(Intercept)         2.553e+00  9.112e-03  280.179  < 2e-16 ***
Connecties         -3.875e-03  2.879e-05 -134.592  < 2e-16 ***
Dimensie            1.065e-03  2.455e-04    4.339 1.43e-05 ***
Connecties:Dimensie  6.174e-05  6.352e-07   97.210  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 95793  on 41649  degrees of freedom
Residual deviance: 19726  on 41646  degrees of freedom
AIC: 165693

Number of Fisher Scoring iterations: 4
```

Figure 9: R output glm functie voor poisson regressie

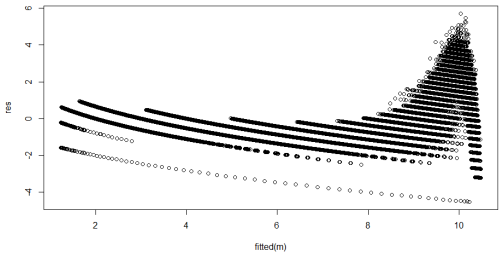
Als we naar de p-waarde kijken va nde interactieterm zien we dat deze significant is op $\alpha = 0.05$. Het model ziet er dan als volgt uit:

$$log(\mathbb{E}(Y)) = 2,533 - 3,875 * 10^{-3} X_1 + 1,065 * 10^{-3} * X_2 + 6,147 * 10^{-5} * X_1 X_2$$

We merken op, dit model is opgesteld met data tot 50 dimensies, We bekijken vervolgens de residu plots en de qqplot met behulp van R-studio om te bekijken hoe goed deze modellen effectief zijn.



(a) qqplot model



(b) Residuplot model

Zowel in de qqplot als in de residuplot zien we een regelmatige afwijking in de data. Dit duidt erop dat we niet de juiste methode van regressie hebben gekozen voor dit model. Achteraf is dit te verklaren door het hoofdstuk hiervoor waar we bepaalde zones kunnen modeleren in ons model met behulp van veeltermen. Een exponentiële groei zal dan uiteraard een grotere systematische afwijking hebben moesten we deze gebruiken.

6.3 $\mathbf{p}^{(0)}$ kiezen

Google gebruikt waarschijnlijk niet de vector $\mathbf{p}^{(0)}$ met alle waardes gekozen gelijk aan 1/N, maar zal waarschijnlijk een meer efficiënte vector kiezen [12]. Het volgende voorbeeld gebruikt volgende matrix A:

$$\mathrm{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Als deze matrix gebruikt wordt in de PageRank functie met $\alpha = 0.85$, de foutmarge 10^{-8} en de p-waarde zoals hieronder beschreven, worden volgend aantal iteraties verkregen.

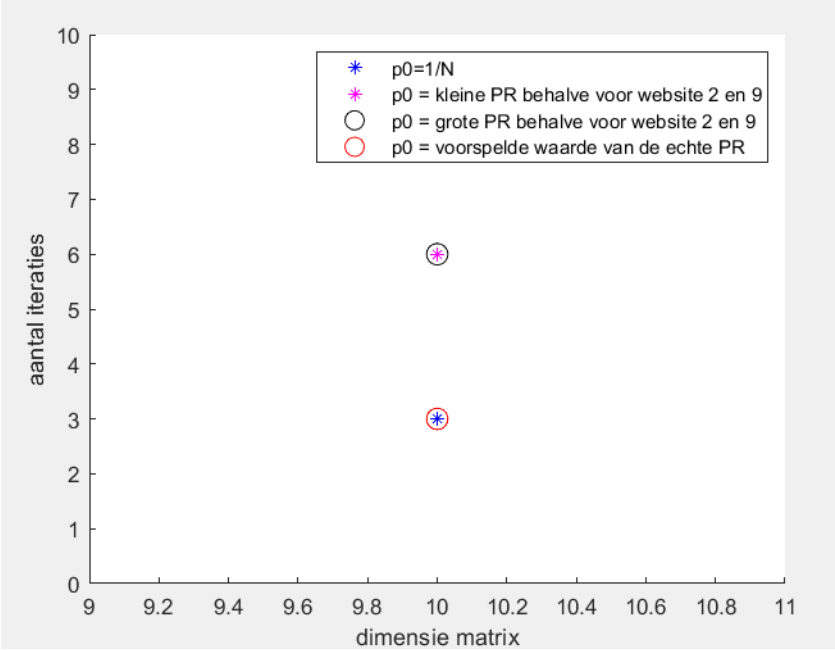


Figure 11: Deze figuur bevat het aantal iteraties voor de matrix A in het PageRank algoritme met de $\mathbf{p}^{(0)}$ waardes gekozen zoals in de legende beschreven.

De p-waardes in de afbeeldingen zijn zo gekozen, omdat deze interessant leken. We zien dat het minste iteraties optreden als $\mathbf{p}^{(0)}$ gekozen is met gelijke waardes zoals initieel voorgesteld in ons model en bij een voorspelde waarde van de echte toekomstige PageRank. Deze voorspelde waarde is gekozen door het algoritme een paar iteraties te laten doen en dan dit te kiezen als startwaarde, dus is het logisch dat deze minder iteraties nodig heeft.

Ook uit de resultaten in 12 blijkt dat de $\mathbf{p}^{(0)}$-vectoren met gelijk gekozen waardes of een voorspelde waarde over het algemeen minder iteraties bevat en dus beter is om mee te werken.

Besluit: uit deze resultaten concluderen we dat de keuze van de $\mathbf{p}^{(0)}$-vector invloed heeft op het aantal iteraties en dus de vlotheid van het model.

Opmerking: De keuze van de $\mathbf{p}^{(0)}$-vector beïnvloed ook de uiteindelijke PageRank en is niet voor alle $\mathbf{p}^{(0)}$-waardes hetzelfde. Welke het beste resultaat geeft wordt hier niet besproken.

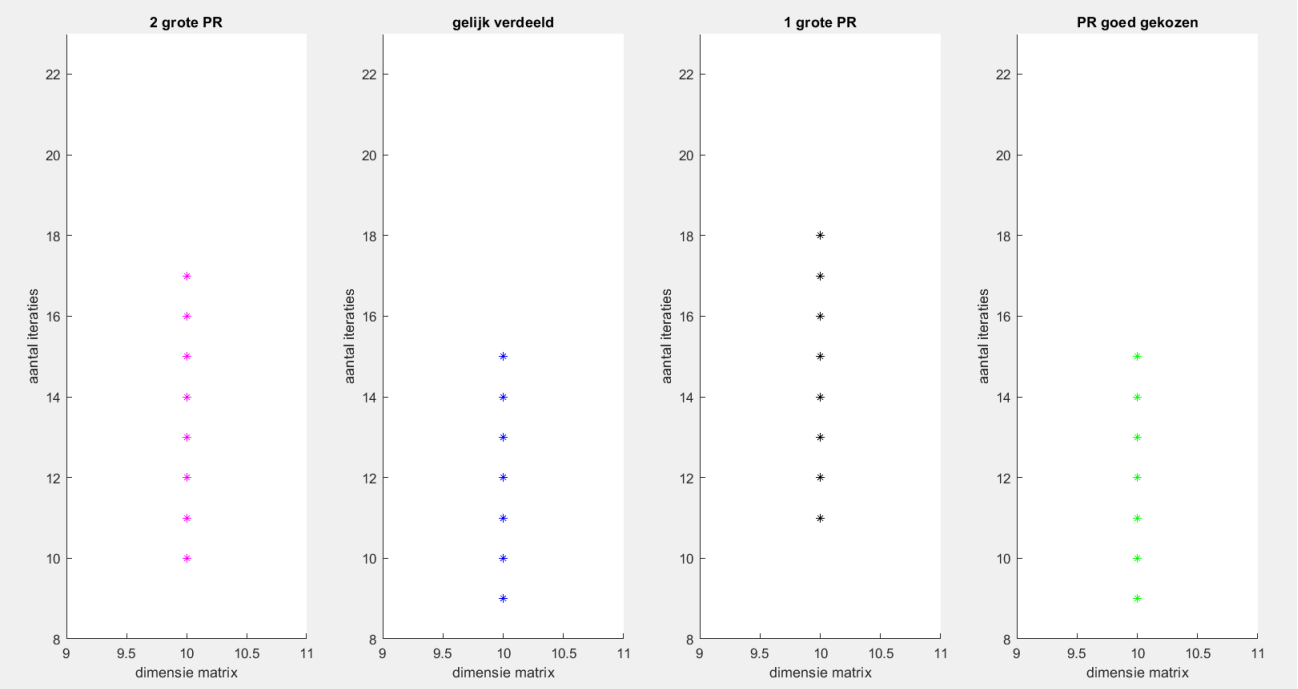


Figure 12: Deze figuur bevat het aantal iteraties voor een willekeurige matrix A van dimensie 10. De gekozen $\mathbf{p}^{(0)}$-waardes zijn hier: 2 random gekozen grote waardes en de rest kleinere waardes, een gelijke verdeelde vector $\mathbf{p}^{(0)}$, een $\mathbf{p}^{(0)}$-vector met 1 random gekozen grote waarde en de rest kleiner en als laatste een voorspelde waarde (gekozen zoals hierboven beschreven).

7 Besluit

We hebben verschillende methodes toegepast om een efficiëntere methode te vinden om de PageRank te vinden.

We zien duidelijk op de heatmap (zie figuur 8) dat het aantal iteraties sterkt zakt naarmate dat er meer connecties (linken naar andere sites) zijn. Dit heeft Google natuurlijk niet zelf in de hand, aangezien Google de sites niet schrijft.

Onze kansvector van de 0de stap $\boldsymbol{p}^{(0)}$ anders opstellen bezorgde ons ook niet veel succes.

Google zal een ander algoritme gebruiken dan ons algoritme, welke veel efficiënter zal werken. Daarbij houden ze rekening met veel meer variabelen. Hoewel ons model veel simplificaties bevatte zal het algoritme van Google veel efficiënter werken.

De PageRank blijft een methode die vaak gebruikt wordt om de relevantie van bepaalde 'pagina's' te bepalen. Het wordt ook niet enkel door Google gebruikt. Zo zijn er nog andere toepassingen voor de PageRank:

- Wetenschappelijke impact van onderzoekers
- Sport
- Twitter
- ...

Hieruit besluiten we dat de PageRank inderdaad een efficiënte methode is om de relevantie van iets te bepalen.

References

[1] Perron–Frobenius Theorem. https://en.wikipedia.org/wiki/Perron-Frobenius_theorem.

[2] Stochastic Matrix. https://en.wikipedia.org/wiki/Stochastic_matrix.

[3] Wat zijn alt tags bij afbeeldingen? (uitleg definitie). https://www.searchsignals.nl/woordenboek/alt-tags/, Jun 2021.

[4] https://en.wikipedia.org/wiki/PageRank, May 2023.

[5] In-depth guide to how google search works, 2023.

[6] Link buidling. https://nl.wikipedia.org/wiki/Link_building, 2023.

[7] Link farm. https://en.wikipedia.org/wiki/Link_farm, 2023.

[8] PageRank. https://en.wikipedia.org/wiki/PageRank, 2023.

[9] Wat is google bot, 2023.

[10] BJÖRN JOHANSSON and EMIL ÖSTERBERG. *Algorithms for Large Matrix Multiplications*. SCHOOL OF ENGINEERING SCIENCES, STOCKHOLM, 2018.

[11] Peter W. Jones and Peter Smith. *Stochastic Processes, An Introduction*. CRC Press, third edition, 2017.

[12] Alfio Quarteroni and Paola Gervasio. *A Primer On Mathematical Modelling*, volume 121. Springer, 2020.

[13] Gery Stevens. Zoekmachineoptimalisatie: Alles over seo en vindbaar worden met je site! `https://geryaal.nl/online-marketing/seo/wat-is-zoekmachineoptimalisatie/`, Mar 2023.

8 Bijlage

8.1 Matlab Functies

```
\begin{knitrout}
\definecolor{shadecolor}{rgb}{0.969, 0.969, 0.969}\color{
    fgcolor}\begin{kframe}
\begin{alltt}
function [A] = \hlkwd{Nodes_generator}(n,k)
\hlkwd{if}(k>n^2-n)
    \hlkwd{disp}(\hlstr{"TOO MUCH CONNECTIONS, LESS THAN n^2-
    n, KEEP 0 ON DIAG"})
    return
end
A=\hlkwd{randi}([0 1],n,n);
A = A-\hlkwd{diag}(\hlkwd{diag}(A));
N = \hlkwd{nnz}(A);

T = N>k;
W = \hlkwd{randperm}(\hlkwd{length}(\hlkwd{find}((A+\hlkwd{
    eye}(n,n)*(~T))==1*T)),\hlkwd{abs}(k-N));
V=\hlkwd{find}(A+\hlkwd{eye}(n,n)*(~T)==1*T);
\hlkwd{A}(\hlkwd{V}(W))=1*(~T);
A = A-\hlkwd{diag}(\hlkwd{diag}(A));
%
% clf
% p=\hlkwd{plot}(\hlkwd{digraph}(A),\hlstr{"bo"},\hlstr{"
    Layout"},\hlstr{"auto"});
% p.NodeColor =\hlstr{"r"};
% p.MarkerSize=7;
% p.LineWidth = 1;
end
\end{alltt}
\end{kframe}
\end{knitrout}

\begin{knitrout}
\definecolor{shadecolor}{rgb}{0.969, 0.969, 0.969}\color{
    fgcolor}\begin{kframe}
\begin{alltt}
function [G,k,p] = \hlkwd{PageRank}(A,alpha,p0,error)
n = \hlkwd{length}(A);
S=\hlkwd{zeros}(n,n);
for i=1:n
    summ = \hlkwd{sum}(\hlkwd{A}(:,i));
```

```
    \hlkwd{if}(summ==0)%black hole
        \hlkwd{S}(:,i) = 1/n*\hlkwd{ones}(n,1);
    end
    for k=1:n
        \hlkwd{if}(\hlkwd{A}(k,i)~=0)
            \hlkwd{S}(k,i)=1/summ;
        end
    end
end

G = alpha .* S + (1-alpha)/n*\hlkwd{ones}(n,n);
k=0;
p=p0;
\hlkwd{while}(~\hlkwd{isequal}(G*p,p) && \hlkwd{max}(\hlkwd{
    abs}((G*p-p)./G*p))>error)
    p=G*p;
    k=k+1;
end
end
\end{alltt}
\end{kframe}
\end{knitrout}
```

Listing 1: Example MATLAB code