Bachelorscriptie
Informatica / Informatiekunde

Radboud Universiteit

Link prediction op het RU domein

Auteur:
Sietse Mooren
4238508

Inhoudelijk begeleider:
Dr. Suzan Verberne
s.verberne@cs.ru.nl

Tweede lezer:
Prof. dr. ir. Arjen de Vries
A.deVries@cs.ru.nl

24 januari 2017

Samenvatting

Verschillende online hulpmiddelen als zoekmachines en browsers willen hun gebruikerservaring optimaliseren. Een manier om dit te doen is door te voorspellen welke webpagina een gebruiker wil gaan bezoeken en hier op te anticiperen. Dit onderzoek richt zich op link prediction binnen één domein. In deze scriptie wordt een Markov chain netwerk getraind op een domein dat bestaat uit meer dan 35.000 unieke URL's en de resultaten worden vergeleken met de resultaten van eerder onderzoek op dezelfde dataset. Verder bekijken we of Neurale netwerken betere voorspellingen kunnen doen door gebruik te maken van meer informatie over de bezochte URL's.

Inhoud

Hoofdstuk 1

Inleiding

De Radboud universiteit heeft vier jaar lang alle web requests binnen het Radboud domein opgeslagen in apache log files. Door slim gebruik te maken van de data zou een algoritme op de server kunnen voorspellen welke webpagina's gebruikers mogelijk willen bezoeken. Het voorspellen van URL's kan er voor zorgen dat bezoekers makkelijker en sneller bij pagina's komen die voor hun relevant zijn. We kunnen bijvoorbeeld suggesties geven aan een gebruiker. Verder kun je pagina's sneller bezoeken door deze alvast in de cache te zetten [5].

Bram Arends heeft in zijn research report [1] al aangetoond dat het voorspellen van URL's op basis van de bezochte pagina's van een gebruiker heel goed mogelijk is. Het voorspellen van pagina's wordt in dit onderzoek *link prediction* genoemd en wij zullen deze term in de rest van de scriptie gebruiken.

Om link prediction te realiseren is in het verleden (net als in het onderzoek van Bram Arends) gebruik gemaakt van Markov chains omdat het zeer intuïtief is. Het is tot nu toe vooral bij kleinere domeinen toegepast. In deze scriptie lever ik twee belangrijke bijdragen op het eerdere werk met dezelfde data:

- Door gebruik te maken van Neurale Netwerken kunnen we achter diepere verbanden komen en voorspellingen doen op URL's waar Markov chains dit niet kunnen.
- Bovendien waren de resultaten van Bram niet allemaal zoals verwacht (zie sectie 2.1). In deze scriptie zullen we daar in meer detail kijken.

De onderzoeksvragen die ik in deze scriptie ga beantwoorden zijn:

1. Zijn de resultaten gevonden door Bram Arends te reproduceren?
2. Kunnen er betere voorspellingen gemaakt worden door gebruik te maken van een Neurale Netwerken?

3. Welke features zijn het belangrijkste in het voorspellen van de volgende URL?

Hoofdstuk 2

Onderzoek

2.1 Reproduceerbaarheid

2.1.1 Data voorbereiding

Voor we kunnen beginnen met het vergelijken van de markov chain models moeten we eerst zorgen dat we dezelfde data gebruiken. In het paper van Bram Arends [1] is de data van 31/01/2011 tot 08/02/2011 gebruikt waarbij 08/02/2011 de test-set is. Ook ik zal deze periode gaan gebruiken voor mijn experimenten.

- 99.999.99.9 - - [05/Feb/2011:04:02:40 +0100] "GET http://www.ru.nl/nieuws/persberichten-0/vm/overzicht/@793936/ dokter-knie/ HTTP/1.1" 200 18829 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)" TCP REFRESH MISS:FIRST UP PARENT
- 999.99.9.999 - - [05/Feb/2011:04:03:15 +0100] "HEAD http://www.ru.nl/nsm HTTP/1.1" 301 151 "-" "Consilio/3.01.03 (Web-Hare Application Portal; LinkChecker)" TCP MISS:NONE

Bovenstaande items geven een sample weer uit de apache log[1] file van 31/01/2011. De log files worden gebruikt om feedback te krijgen over de activiteiten en de prestaties van een domein. Deze log files moeten nog gefilterd worden. Afbeeldingen, fonts en andere file types moeten worden verwijderd uit de log files.[2] Wat we over houden zijn de URL's die de gebruikers hebben ingetypt of hebben aangeklikt.

Buiten gebruikers zijn er ook web crawlers die het domein bezoeken. Deze web crawlers bezoeken meer URL's dan normale bezoekers. Er is

[1] https://httpd.apache.org/docs/1.3/logs.html

[2] Gefilterde files eindigend op axd,aspx,bmp,doc,docx,jpg,jpeg,gif,css,js,png,sav,swf,ttf, txt,xls,xlsx,xml,zip,ppt,pptx,ico.

daarom door Bram Arends gekozen om bezoekers die meer dan vijftig URL's in een sessie hebben of meer dan vijftig sessies in de data hebben te verwijderen uit de data. Een sessie is de periode vanaf dat een gebruiker de eerste request naar het Radboud domein doet tot deze gebruiker stopt met het bezoeken van URL's binnen dit domein. Uit de logfiles is moeilijk af te leiden wanneer een gebruiker precies gestopt is. Om dit probleem op te lossen heb ik ervoor gekozen (net als Bram Arends) om een sessie te beëindigen als een gebruiker langer dan dertig minuten geen nieuwe URL heeft bezocht [1].

Nu we alle data in het zelfde format hebben als in [1] kunnen een Markov chain model maken en de resultaten vergelijken.

2.1.2 Markov chain voor link prediction

Een Markov chain model berekent hoe groot de kans is om van een gegeven URL s_i naar de volgende te bezoeken URL s_j te gaan. Het Markov model doet dit door gebruik te maken van states. In ons geval zijn states de door gebruikers bezochte URL's. Gegeven een set van states $S = s_0, s_1, s_2, ..., s_n$ berekent het Markov chain model de kans van state s_j door alleen te kijken naar de huidige state en niet naar states die voor state s_i zijn gekomen. In [2] wordt de kans voor de states berekend en opgeslagen in een matrix door gebruik te maken van de volgende formule.

$$A(s, s^{'}) = \frac{C(s, s^{'})}{\sum_{s^{''}} C(s, s^{''})}[1]$$

$A(s, s^{'})$ is de kans dat je van state s naar state $s^{'}$ gaat. $C(s, s^{'})$ is het totaal aantal overgangen van state s naar $s^{'}$. Tot slot staat $\sum_{s^{''}} C(s, s^{''})$ voor de som van alle overgangen die vanuit state s gemaakt kunnen worden.

Voor het verkrijgen van een label van een state s wordt gebruikt gemaakt van formule 2.2. Deze formule geeft weer hoe je de label voorspeld aan de hand van eerder bezochte URL's. Er is in het paper van Bram Arends onderscheid gemaakt uit drie voorspellingen. De eerste is historie 1, deze gebruikt alleen de meest recente URL. Historie 2 gebruikt buiten de meest recente URL ook de een-na-laatste bezochte URL gebruikt. De laatste variant is historie 3 die gebruikt de drie laatst bezochte URL's.

$$\underbrace{\underbrace{url_1 \longrightarrow \underbrace{url_2 \longrightarrow \underbrace{url_3}_{\text{historie 1}}}_{\text{historie 2}} \longrightarrow \underbrace{url_4}_{\text{label}}}_{\text{historie 3}}} \tag{2.1}$$

$$s(t) = max(a_1 i_{t-1} A, a_2 i_{t-2} A^2, a_3 i_{t-3} A^3, ...)[1] \tag{2.2}$$

Hier staat a_i voor een gewicht dat door de gebruiker is gespecificeerd. In [1] zijn de gewichten geven in tabel 2.1 gebruikt. Ik gebruik dezelfde

gewichten. A is de matrix met alle transities, i_t geeft de state weer op tijdstip t en $s(t)$ geeft de kans om naar state s te gaan op tijdstip t.

historie	a_{t-1}	a_{t-2}	a_{t-3}
1	1	-	-
2	0.75	0.25	-
3	0.75	0.125	0.125

Table 2.1: gebruikte gewichten

2.1.3 Resultaten

Nu we een gelijke data en algoritmiek hebben als in het onderzoek van Bram Arends kunnen we de resultaten gaan vergelijken. Het eerste waar we naar gaan kijken zijn voorspellingen waarbij uitsluitend gekeken wordt naar de URL met de hoogste opvolgkans. We gaan dit testen op de sessies van de achtste dag. Doordat we hoogstens 3 URL's nodig hebben wordt net als bij het onderzoek van Bram Arends één sessie opgesplitst. Onderstaand is een voorbeeld te zien van hoe een sessie wordt opgedeeld voor oplopend historie 1, 2 en 3.[3]

$$
\begin{aligned}
&\underbrace{url_3}_{\text{historie voor } url_4} \rightarrow \underbrace{url_4}_{\text{eerste label}} \rightarrow \underbrace{url_5}_{\text{historie voor } url_5} \rightarrow \underbrace{url_6}_{\text{tweede label}} \rightarrow ... \\
&\underbrace{url_2 \rightarrow url_3}_{\text{historie voor } url_4} \rightarrow \underbrace{url_4}_{\text{eerste label}} \rightarrow \underbrace{url_5 \rightarrow url_6}_{\text{historie voor } url_7} \rightarrow \underbrace{url_7}_{\text{tweede label}} \rightarrow ... \\
&\underbrace{url_1 \rightarrow url_2 \rightarrow url_3}_{\text{historie voor } url_4} \rightarrow \underbrace{url_4}_{\text{eerste label}} \rightarrow \underbrace{url_5 \rightarrow url_6 \rightarrow url_7}_{\text{historie voor } url_8} \rightarrow \underbrace{url_8}_{\text{tweede label}} \rightarrow ...
\end{aligned}
\tag{2.3}
$$

In tabel 2.2 en 2.3 zijn de gemiddelde resultaten uit het paper van Bram te zien vergeleken met de resultaten van de volledige test set op dag acht. We hebben voor het gemiddelde resultaat van Bram zijn resultaten gekozen omdat in zijn onderzoek vijf keer 1000 willekeurige sessies uit de testset zijn gehaald. Van deze vijf willekeurige data sets is steeds getest welk percentage goed voorspeld was. De resultaten van deze data sets hadden een standaard deviatie van minder dan één procent. Dit geeft aan dat de test set uniform is en een vergelijking met de volledige test set gelijke resultaten zou moeten geven. We hebben geen cross validatie gebruikt omdat we dezelfde train-set (31/01/2011 t/m 07/02/2011) en testset (08/02/2011) als Bram Arends wilde gebruiken.

[3]De oorspronkelijke implementatie van Bram Arends was in Matlab. De documentatie `https://nl.mathworks.com/help/matlab/ref/mat2cell.html` bevestigt dat de testset van Bram Arends op bovenstaande manier (2.3) gecreëerd is.

Method	Hist 1	Hist 2	Hist 3
Markov Bram Arends	0.6431	0.4673	0.8027
Markov Sietse Mooren	0.7113	0.5754	0.7310

Table 2.2: Markov chain vergelijken met het beste resultaat

Method	Hist 1	Hist 2	Hist 3
Markov Bram Arends	0.8290	0.6106	0.9029
Markov Sietse Mooren	0.7605	0.6425	0.7850

Table 2.3: Markov chain vergelijken met de top drie resultaten

2.1.4 Discussie

Uit de resultaten is af te lezen dat ze niet hetzelfde zijn. Het verschil is steeds rond de tien procent. De redenen hiervoor zijn ons niet helemaal duidelijk. Het verschil kan mogelijk worden verklaard door de volgende drie punten.

- Verschil in het filteren van request met betrekking tot de file types.
- In het onderzoek van Bram wordt naast het Markov chain model ook clustering van URL's gedaan. Helaas worden in dat paper niet de resultaten gegeven voor de Markov chain zonder clustering.
- Uit het onderzoek van Bram is moeilijk te achterhalen wat de manier is geweest voor het opsplitsen van de sessies (van dag 08/02/2011) naar een testset die gebruikt kan worden door het Markov model. Mochten we beide op een andere manier de testset hebben gemaakt dan levert dit al snel verschillende resultaten op. Een van de verschillen kan de eerste gebruikte label zijn. Zo heb ik er voor gekozen dat zowel hist 1,2 en 3 de vierde URL als eerste label hebben (zie figuur 2.1). Dit omdat dan alle histories exact de zelfde labels hebben. Uit de paper van Bram was dit niet te achter halen en kan het dus zijn dat hist 1 en 2 andere labels hebben(zie figuur 2.3).

Verder hebben bovenstaande resultaten (tabel 2.2 en 2.3) bij historie 2 een lagere nauwkeurigheid dan bij historie 1 en 3. Bram kon dit niet verklaren. Mijn nadere analyse wijst uit dat dit waarschijnlijk komt doordat bij het opsplitsen van de testsessies voor de verschillende histories weinig overeenkomende labels zijn. Deze splitsing kan er voor zorgen dat bepaalde combinaties die in historie 1 en 3 voorkomen niet in 2 voorkomen. Door uit te zoeken welke combinaties afwijkend gedrag vertonen en deze te verwijderen zouden de resultaten meer overeen moeten komen met onze verwachtingen.

Door naar de goed voorspelde URL's te kijken zien we dat de twee volgende combinaties veel voorkomen: ru.nl → ru.nl/ en ru.nl/ → ru.nl/?. De / geeft aan dat de URL een directory is. Het ? geeft aan dat de URL een string query is. Beide varianten worden niet door de gebruiker bezocht maar kan een bijproduct zijn van bijvoorbeeld java-script. Door de log files opnieuw te filteren met als extra toevoeging de URL's die eindigen op / of ? krijgen we de resultaten te zien in tabel 2.4 en 2.5.

Method	Hist 1	Hist 2	Hist 3
Markov zonder ? en / filtering	0.7113	0.5754	0.7310
Markov met extra filtering	0.4582	0.4624	0.4606

Table 2.4: Markov chain vergelijking origineel met extra filtering gebruik hoogste resultaat

Method	Hist 1	Hist 2	Hist 3
Markov zonder ? en / filtering	0.7605	0.6426	0.7850
Markov met extra filtering	0.5334	0.5496	0.5477

Table 2.5: Markov chain vergelijking origineel met extra filtering gebruik top drie resultaten

De resultaten zijn nu meer in lijn met onze verwachtingen: als we de geschiedenis langer maken, krijgen we in vergelijking met historie één betere voorspellingen. Met andere woorden: de twee extra gebruikte URL's bevatten nuttige informatie voor link prediction, hoewel de verschillen klein zijn.

Tot slot willen we testen of de manier van het opsplitsen van de sessies in de testset invloed heeft op de nauwkeurigheid. Dit doen we door de data op een andere manier te prepareren, namelijk gebruik makend van een sliding window, waarbij het label steeds één URL naar rechts opschuift. Voor historie 3 ziet dat er als volgt uit.

$$\underbrace{url_1 \rightarrow url_2 \rightarrow url_3}_{\text{historie voor } url_4} \rightarrow \underbrace{url_4}_{\text{eerste label}} \rightarrow \underbrace{url_2 \rightarrow url_3 \rightarrow url_4}_{\text{historie voor } url_5} \rightarrow \underbrace{url_5}_{\text{tweede label}} \rightarrow ... \tag{2.4}$$

De resultaten voor het opnieuw opsplitsen van de testsessies zijn te zien in onderstaande tabellen.

Method	Hist 1	Hist 2	Hist 3
Markov zonder ?, / filtering en zonder sliding window	0.7113	0.5754	0.7310
sliding window zonder ? en / filtering	0.5230	0.5195	0.5219
Markov met extra filtering zonder sliding window	0.4582	0.4624	0.4606
sliding window met extra filtering	0.4633	0.4584	0.4617

Table 2.6: vergelijking opsplitsen test session met hoogste resultaat

Method	Hist 1	Hist 2	Hist 3
Markov zonder ?, / filtering en zonder sliding window	0.7605	0.6426	0.7850
sliding window zonder ? en / filtering	0.5901	0.5926	0.5956
Markov met extra filtering zonder sliding window	0.5334	0.5496	0.5477
sliding window met extra filtering	0.5371	0.5419	0.5439

Table 2.7: vergelijking opsplitsen test session met drie hoogste resultaten

Door gebruik te maken van een sliding window zijn voor zowel de data set met de / en ? als de data set zonder / en ? de resultaten naar verwachting. De verwachting was dat extra historie wel invloed zal hebben maar niet voor verschillen van tien procent kan zorgen. Nu we uniforme resultaten hebben voor de drie gebruikte histories kunnen we gaan kijken of door gebruik te maken van een ander classificatie algoritme we betere resultaten kunnen behalen.

2.2 Link predictie met Neurale Netwerken (NN)

Binnen de computer science afdeling is veel succes behaald door te werken met Neurale Netwerken. Doordat onze dataset een hoog aantal verschillende labels heeft werken veel classificatie algoritmes niet. Zo hebben we met weinig succes (scores onder de vijf procent) geëxperimenteerd met Naive Bayes.[4] Het leek ons daarom goed te kijken wat NN voor deze dataset kan betekenen en omgaan met de vele aantal labels. Voor we verder gaan met de input voor het NN wordt in het kort uitgelegd hoe een NN werkt. Onderstaande afbeelding geeft op een hoog niveau weer wat hoe een NN tot een label komt.

$$x_1 \rightarrow \boxed{W^1} \rightarrow x_2 \rightarrow \boxed{W^2} \rightarrow, x_3, ..., \rightarrow \boxed{W^i} \rightarrow x_i$$

Hier is x de input van W waarbij W een laag is met een input en een output. In bovenstaand voorbeeld is x_1 de input voor W^1 met als resultaat

[4] http://scikit-learn.org/stable/modules/naive_bayes.html

x_2 die weer in de volgende laag gebruikt wordt als input. Als de data door deze ketting van lagen is gegaan wordt voor iedere laag berekend welke parameters belangrijk zijn. Dit proces heet back error propagation. Hoe goed het NN werkt wordt berekend aan de hand van een loss function. De functie berekent hoe goed je netwerk werkt tegenover de echte waardes. Dit proces kan herhaald worden tot de optimale staat van het Netwerk bereikt is [7]. Om voorspellingen te doen wordt de test data als input van laag W^1 gegeven die het dan doorgeeft tot de laatste laag een voorspelling geeft. Bij het voorspellen wordt geen back error propagation meer gedaan.

2.2.1 Features

Om het resultaat van het Markov Chain model te verbeteren kunnen we in een neuraal netwerk meer features gaan gebruiken die ons mogelijk meer informatie opbrengen. De nieuw toegevoegde features zouden er voor moeten zorgen dat het neurale netwerk beter onderscheid kan maken over de te voorspellen pagina's. In deze benadering wordt iedere nieuw bezochte URL binnen een sessie gezien als een losse feature. De verdeling tussen de train-(31/01/2011 t/m 07/02/2011) en testset (08/02/2011) blijft onveranderd .

$$\underbrace{url_1}_{\text{feature 1}} \rightarrow \underbrace{url_2}_{\text{feature 2}} \rightarrow \underbrace{url_3}_{\text{feature 3}} \rightarrow ...$$

Buiten de URL kunnen we van de log files andere gegevens gebruiken. Zo is er een datum en de status van de gerequeste URL. Een voorbeeld van een status is 200 wat voor een get request betekent dat de request succesvol was en de opgevraagde data is teruggestuurd [4]. Door gebruik te maken van deze drie gegevens hebben we de volgende features gemaakt/gebruikt.

- De URL, de gerequeste URL door een bezoeker.
- De status, is de waarde die binnen de log file bij een URL te vinden is.
- De tijd tot de volgende URL, de tijd in minuten tot de volgende URL wordt berekend door de tijd van een URL van de tijd van de opvolgende URL af te trekken.
- De dag, is de dag van een maand voor de eerste URL binnen een sessie (1 tot maximaal 31). We hebben deze feature toegevoegd om maandelijkse patronen te kunnen detecteren.
- De count, het aantal keer dat de URL voorkomt binnen die feature.
- Het cluster, de kmean over de eerst bezochte URL van een sessie. In het onderzoek van Bram zijn 30 clusters gevonden [1]. Ik ga van het zelfde aantal clusters uit. Door alleen gebruik te maken van de eerst

bezochte URL wordt iedere gebruiker bij het bezoeken van het RU domein in een cluster gezet.

- Is sub URL, boolean die aangeeft of een text van de URL voorkomt in de opvolgende URL.

De dag en het cluster features komen maar één keer voor. Voor de andere features wordt het voor alle URL's in de sessie berekend. Onder staand een voorbeeld sessie gevolgd door de daaruit volgende features.

$$\underbrace{ru.nl, 01/Oct/2014:00:01:03, 200}_{eersterequest} \rightarrow$$

$$\underbrace{ru.nl/onderzoek, 01/Oct/2014:00:01:33, 200}_{tweederequest} \rightarrow$$

$$\underbrace{ru.nl, 01/Oct/2014:00:03:33, 200}_{derderequest} \rightarrow$$

$$\underbrace{http://www.ru.nl/overons, 01/Oct/2014:00:05:03, 200}_{vierderequest} \rightarrow$$

$$\underbrace{http://www.ru.nl/overons/contact, 01/Oct/2014:00:05:13, 200}_{vijfderequest}$$

De URL's krijgen per feature een unieke label toegewezen in de vorm van een getal.

	Day	Cluster	URL_1	Status_1	Count
1	1	1	1	200	2
2	1	1	2	200	1
3	1	1	1	200	2
4	1	1	3	200	1

Table 2.8: feature van 1 sessie met geschiedenis 1

Geschiedenis 1 heeft geen "is sub URL" en tijd tot de volgende URL omdat de opvolgende URL voorspeld moet worden en de gegevens dus nog niet bekend zijn. Voor geschiedenis 2 en 3 kunnen we wel de is sub URL en het verschil in tijd bepalen. In onderstaande tabel 2.9 is te zien welke features je krijgt voor de boven gegeven sessie.

	Day	Cluster	URL_1	Status_1	Count	is_sub_URL_1	tijd_tot_volgende_URL_1	URL_2	Status_2	Count_2
1	1	1	1	200	2	True	0.5	2	200	1
2	1	1	2	200	1	False	2	1	200	1
3	1	1	1	200	2	True	1.5	3	200	1

Table 2.9: feature van 1 sessie met geschiedenis 2

Url, status, is sub url, dag en cluster zijn categorical features. NN kan hier heel goed mee omgaan. Tijd en count is continious data het Neurale Netwerk kan deze data wel gebruiken maar werkt minder goed dan categorical data. Nu we weten welke features we gaan gebruiken en testen moeten we weten hoeveel lagen en nodes per laag we gaan gebruiken. Voor dit onderzoek hebben we gekozen voor twee lagen met in de eerste laag 1000 nodes en de tweede laag 500 nodes. Beide lagen hebben een dropout van twintig procent, wat wil zeggen dat twintig procent van de input niet wordt gebruikt. Dit zorgt er voor dat de kans op overfitting kleiner wordt[6]. Met deze opzet hebben we meerdere testen gedraaid om te kijken of het NN beter kan classificeren dan het Markov chain algoritme. Verder kunnen we zien of de extra features meer informatie geven waardoor we hogere scores krijgen bij het classificeren. Voor het opstellen van het NN hebben we binnen python de Keras library gebruikt.[5]

features	Hist 1	Hist 2	Hist 3
URL	0.2625	0.3945	0.3890
URL,Status	0.2628	0.3929	0.3852
URL,Status,kmean,day	0.2205	0.3853	0.3784
URL,Status,kmean,day,count	0.1691	0.3828	0.3751
URL,Status,kmean,day,count,sub url, time to next url		0.1546	0.0794

Table 2.10: Resultaten van NN op de testset met verschillende features en gebruikt van hoogst voorspelde label

features	Hist 1	Hist 2	Hist 3
URL	0.3704	0.4810	0.4719
URL,Status	0.3698	0.4769	0.4674
URL,Status,kmean,day	0.3201	0.4697	0.4619
URL,Status,kmean,day,count	0.2193	0.4660	0.4577
URL,Status,kmean,day,count,sub url, time to next url		0.2384	0.1408

Table 2.11: Resultaten van NN op de testset met verschillende features en gebruikt van de drie hoogst voorspelde label

Wat duidelijk zichtbaar is uit de bovenstaande tabellen (2.10,2.11) is dat een Neuraal Netwerk met iets meer informatie dan alleen historie 1 al een veel betere voorspelling kan doen. In vergelijking met de resultaten van het Markov chain model (tabellen 2.1.4, 2.1.4)is te zien dat we bij hist 1 bijna

[5]https://keras.io/

twintig procent lager scoren voor hist 2,3 is dit slechts 10 procent. De extra features geven voor deze dataset geen extra informatie mee en zorgen er voor een lagere score. Dit kan komen doordat de input voor het NN nog relatief klein is. Verder kan het verschil komen doordat het Neurale netwerk meer parameters heeft dan observaties. Dit zorgt in complexe modellen al snel voor overfitting[3] en dus een lagere score.

Hoofdstuk 3

Conclusies

Door het reproduceren van het onderzoek van Bram Arends weten we dat het moeilijk kan zijn om exact de zelfde resultaten te krijgen. Dit kan komen door andere manieren van het opstellen van test sets, maar ook door kleine verschillen in het preprocessen. Verder zijn we erachter gekomen dat binnen het RU domein opvolgende webpaginas aangemaakt worden die voor een gebruiker niet van belang zijn maar veel invloed kunnen hebben op de resultaten. Denk hierbij aan het ? die vaak na een / kwam. Deze combinatie zorgt voor een vertekend beeld van de resultaten. Door deze URL's eruit te filteren is het gelukt om de resultaten te krijgen die meer in lijn waren met de verwachtingen (geen verslechtering bij een geschiedenis van 2 in plaats van 1 URL). Ook het opsplitsen van de test sessie heeft veel invloed op de resultaten. Het gebruik van een sliding window (zie sectie 2.1.4 figuur 2.4) geeft resultaten naar verwachting en is dan ook de juiste manier om de test sessies op te splitsen.

Het Neurale netwerk had slechtere resultaten dan het Markov chain netwerk. Dit komt waarschijnlijk doordat het NN is geoverfit. De meest aannemelijke reden is dat er te weinig input is zowel in het aantal sessies. Het lijkt er op dat de features: status, tijd tot volgende URL, dag, count, cluster en sub URL geen nuttige informatie geven bij de gebruikte train en test set waardoor net als in het Markov netwerk alleen de URL's bruikbare informatie bevat.

Tot slot wil ik het nog kort hebben over de behaalde percentages van het Neurale Netwerk en het Markov chain algoritme en mogelijk verder onderzoek. Met een nauwkeurigheid tussen de 0.45 en 0.54 is Markov chain nog steeds een goede manier om een voorspelling te binnen het RU domein. Verder is het NN met een score tussen de 0.39 en 0.48 maar rond de zes procent slechter. Het is ook nog niet duidelijk of het Neurale netwerk andere labels voorspelt als het Markov chain model. Het zou kunnen dat een combinatie van de twee netwerken nog mogelijk voor verbetering zorgt. Dit met het gebruik van een grotere train en test set zodat uitsluitsel gegeven kan

worden over de nuttigheid van de gebruikte features in het Neurale netwerk. Verder kan het vergroten van de data set zorgen dat overfitten minder kan worden omdat er dan meer observaties komen in vergelijking met het aantal parameters.

Bibliografie

[1] Bram Arends. Link prediction and web user clustering.

[2] Dartmouth College Charles Grinstead, Swarthmore College J. Laurie Snell. *Introduction to Probability.* American Mathematical Society, 1997.

[3] B S Everitt and A Skrondal. *The Cambridge Dictionary of Statistics; 4th ed.* Cambridge University Press, Leiden, 2010.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1, 1999.

[5] Smith, M. D. Schechter, S., Krishnan, M. Using path profiles to predict http requests. *Computer Networks and ISDN Systems*, 30(1):457–467, 1998.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[7] Jianxin Wu. *Introduction to Convolutional Neural Networks.* National Key Lab for Novel Software Technology Nanjing University, China, 2016.