Voorwoord

Voor U ligt het resultaat van enkele maanden hard werken én van vijf jaar studie aan de Erasmus universiteit. Voor het feit dat ik deze studie heb kunnen volgen en daarnaast zelfs tijd over had om mij in vele andere zaken te verdiepen heb ik mijn ouders te danken. Daarom draag ik dit werk aan hen op. Bedankt.

De keuze voor het onderwerp van mijn scriptie werd meteen duidelijk toen deze door mijn scriptiebegeleider aan mij werd voorgesteld. In deze scriptie zijn meerdere disciplines vertegenwoordigd, waaronder filosofie, kunstmatige intelligentie en het gebied van informatie-zoeksystemen. Omdat dit onderwerp zo breed is kon ik mij uitleven in mijn interessen voor elk van deze vakgebieden, waarbij ik misschien hier en daar wat te enthousiast ben geweest. Ik hoop dat de lezer mij kan vergeven als ik soms ergens te diep in ga op een interessant onderwerp en hoop tevens dat U net zo veel plezier heeft met het lezen van deze scriptie als ik had met het schrijven ervan.

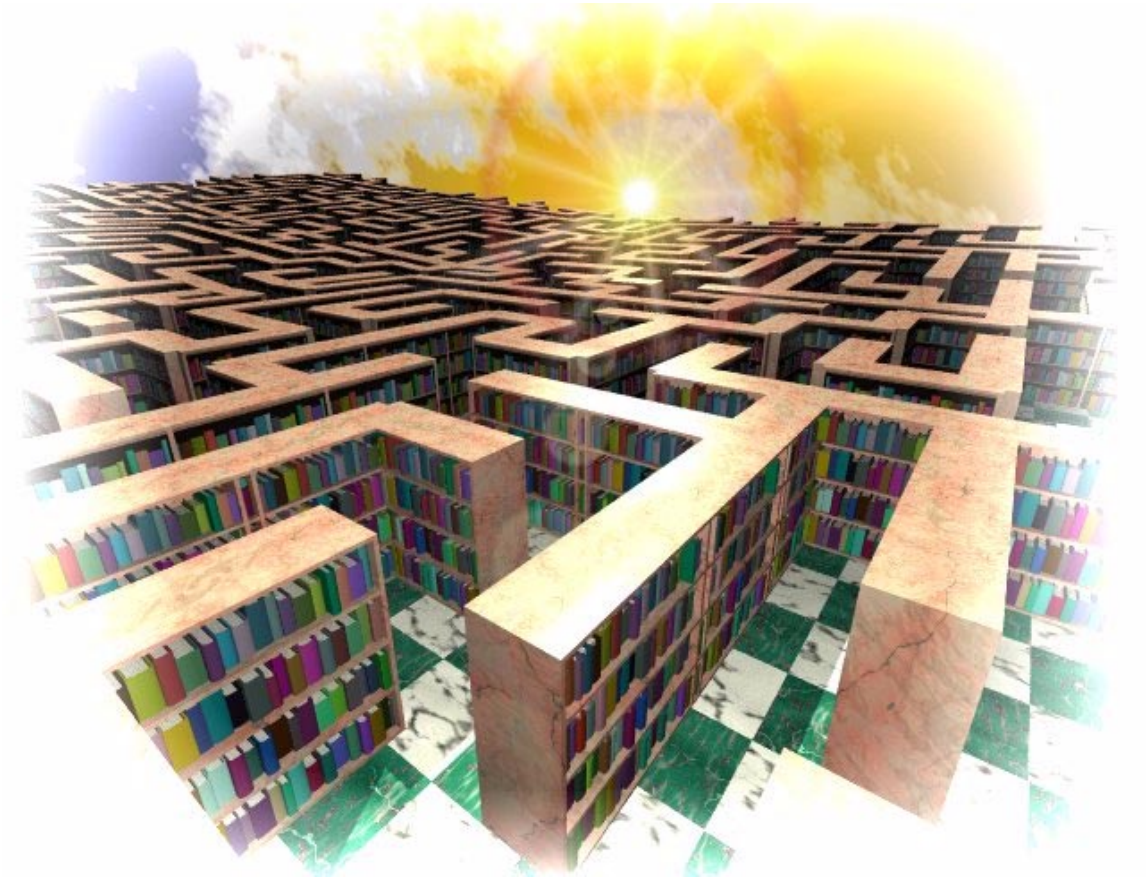
Het zal de lezer niet ontgaan dat deze scriptie niet alleen gericht is op het vinden van een praktische oplossing voor een geconstateerd probleem. Minstens net zo belangrijk is de vraag die in dit werk naar voren komt, namelijk: “wat is kennis?” We zien dat de realiteit van het antwoord op deze vraag zeer complex is. Momenteel vinden we dan ook verschillende modellen van deze werkelijkheid welke allemaal tekort schieten in het benaderen van deze werkelijkheid. Dat wil niet zeggen dat deze modellen fout zijn of dat de ene beter is dan de andere. Momenteel kan alleen door al deze theoretische constructies aan te houden inzicht ontstaan in een dieper gelegen werkelijkheid.

Ik had deze scriptie niet kunnen schrijven zonder de hulp van vele anderen. Ik zou dan ook Jan van der Berg willen bedanken voor zijn goede begeleiding bij het schrijven van deze scriptie. Hiernaast ook dank voor de hulp van Niek Weustink, met name voor het beschikbaar stellen van de testset en Anne Veling voor zijn uitleg over de Aqua-browser. Mijn zus, Karen Schuemie, ben ik zeer erkentelijk voor het feit dat zij de (vele) taalfouten uit deze scriptie heeft verwijderd. Verder wil ik Theo Buitendijk bedanken voor het lenen van zijn kleurenprinter en ben ik Alain Wielaard dank verschuldigd voor het meedenken over de illustraties in deze scriptie.

De illustratie op bladzijde tien stelt overigens een ontluikende lotusbloem voor. In de oosterse symboliek staat deze voor groei van kennis en wijsheid en heeft dan meteen ook betrekking op het uiteindelijke doel van deze scriptie.

Hendrik Ido Ambacht, Juni 1998
Martijn Schuemie

Inhoudsopgave



Hoofdstuk 1: Inleiding

"For those who wish to get clear of difficulties it is advantageous to state the difficulties well; for the subsequent free play of thought implies the solution of the previous difficulties, and it is not possible to untie a knot which one does know"
(Aristoteles, p.1572, Metaphysics)

In deze inleiding wordt eerst aandacht besteed aan de achtergrond van mijn afstudeeronderzoek (1.1). Daarna wordt de probleemstelling geïntroduceerd en besproken (1.2). Vervolgens wordt ingegaan op de gebruikte onderzoeksmethodiek (1.3) en de opbouw van deze scriptie (1.4).

1.1 Achtergrond

We leven in het "informatietijdperk". Dat wil zeggen: informatie heeft in onze samenleving een prominente positie ingenomen. Computers geven ons de mogelijkheid om informatie snel te benaderen, te bewerken en op te slaan, en dat tegen steeds lagere kosten. Als gevolg hiervan treffen we bijvoorbeeld steeds vaker grote 'pakhuizen' van informatie aan (Amerongen 1997), op het internet is het groeiende informatieaanbod niet te stuiten en ook de boekdruk-industrie produceert meer boeken dan ooit te voren. Het gevolg is dat wij meer informatie tot onze directe beschikking hebben dan we ooit kunnen verwerken. De term

‘informatie‘ behoeft echter een nuancering. We kunnen een onderscheid maken tussen ‘gegevens’ en ‘informatie’:

“**Gegevens** (data) zijn de objectief waarneembare neerslag van feiten of kennis op een bepaald medium, zodanig dat deze gegevens uitgewisseld kunnen worden. **Informatie** is de betekenis die een persoon volgens bepaalde conventies aan de gegevens toekent of eraan ontleent.” (Bemelmans, p. 43)

Dat iets een gegeven is kan doorgaans objectief worden vastgesteld, maar of iets informatie is, is afhankelijk van de persoon die het nodig heeft. Gegevens kunnen in een duidelijke structuur opgeslagen zijn, zoals een relationele of object-georiënteerde database, maar ze kunnen ook als tekst, dus in de vorm van een natuurlijke taal, zijn opgeslagen.
We kunnen dus stellen dat er sprake is van een overvloed aan gegevens waaruit wij kunnen kiezen, en afhankelijk van de relevantie van deze gegevens kunnen wij ze ‘informatie’ noemen. Om te helpen bij deze keuze kunnen we gebruik maken van een Informatie-Zoeksysteem (ook wel “Information Retrieval” genaamd). Voor gegevens die tekstueel zijn vastgelegd kunnen we bijvoorbeeld gebruik maken van een systeem dat werkt met trefwoorden of indexen om de documenten te vinden waarin we zijn geïnteresseerd. Deze methoden schieten echter af en toe tekort: de gegevens die aan de gebruiker gepresenteerd worden zijn soms irrelevant. Deze documenten bevatten dan wel de trefwoorden waar de gebruiker naar op zoek was, maar gaan toch over een ander dan het gewenste onderwerp. Daarentegen kan het ook voorkomen dat een document wel over het gewenste onderwerp gaat maar niet de specifieke trefwoorden bevat, waardoor het door het systeem onopgemerkt blijft.
Wat zo’n Informatie-Zoeksysteem mist is inzicht in de betekenis van de documenten. Hierdoor is het systeem niet in staat de relevantie voor de gebruiker te bepalen. Het formuleren van een passende zoekopdracht vergt in de huidige situatie nogal wat moeite (Kaski e.a., 1996). We kunnen dus zeggen dat de genoemde zoeksystemen niet voldoende effectief zijn. Zij missen een gewenste **selectieprecisie** en **vindkracht** (zie hoofdstuk 4).
Veel beter zou een systeem zijn dat tot op zekere hoogte de relevantie van de documenten voor de gebruiker kan bepalen met behulp van kennis over de documenten en de onderwerpen die zij behandelen. In sommige bibliotheken worden boeken ingedeeld in categorieën, en deze categorieën zijn een voorbeeld van dergelijke kennis over het domein van de collectie documenten. Zo’n indeling vergt echter veel werk aangezien deze handmatig moet worden aangelegd en is daarnaast ook erg persoonsgebonden.
Systemen die zelf in staat zijn dergelijke domeinkennis automatisch op te bouwen zijn momenteel al in ontwikkeling. Een voorbeeld hiervan is het WEBSOM-algoritme (Kohonen) dat met behulp van self-organizing maps een tweedimensionale kaart creëert waarop documenten, die qua semantische inhoud gelijk zijn, dichtbij elkaar worden weergegeven. Met behulp van een eenvoudige gebruikersinterface kan dan de kaart worden verkend. Met behulp van enkele muis klikken kan op delen van de kaart worden ingezoomd en kunnen snel relevante documenten worden gevonden. Een belangrijk voordeel is ook dat, indien een document is gevonden dat van belang is voor de gebruiker, de omliggende documenten meestal ook interessant zijn. De gebruiker kan het gevonden document ook gebruiken als ‘strik’ om toekomstige interessante documenten te vangen (Honkela e.a., 1996, p.8).
Een groot nadeel aan het WEBSOM algoritme is echter dat het de complete tekst nodig heeft voor de analyse, wat bij grote documenten voor een probleem kan zorgen. In praktijk blijkt dit algoritme slechts toepasbaar voor kleine documenten, tot ongeveer tien pagina’s. Voor grote documenten, zoals boeken, is dit algoritme dus niet toepasbaar. Naast het feit dat grote documenten niet goed door het WEBSOM-algoritme kunnen worden verwerkt is

het feit dat niet alle boeken in elektronische vorm beschikbaar zijn een groot probleem. Indien dit het geval is zou het complete boek moeten worden gescand en omgezet moet worden van digitale afbeelding naar tekst, wat grote problemen en kosten met zich meebrengt aangezien dit proces niet foutloos verloopt (Alexander 1997).
Als oplossing voor dit probleem is door N. Weustink het idee geopperd om niet het volledige boek te gebruiken, maar slechts een deel ervan, en dan met name de index (Weustink). De door hem ontwikkelde methode kon echter niet een bevredigende ordening van de documenten leveren; een oplossing voor dit probleem blijft dus nog steeds noodzakelijk.

1.2 Probleemstelling

In de vorige paragraaf is gebleken dat er sprake is van een overvloed aan (tekstuele) gegevens. Het gevolg hiervan is dat het voor de gebruiker moeilijk is om de voor haar/hem relevante informatie te vinden. De huidige zoeksystemen gebaseerd op trefwoorden en indexen schieten hierbij tekort omdat zij niet op de hoogte zijn van de semantische betekenis van de documenten en de systemen die gebruik maken van een handmatig aangelegde indeling (ook wel 'thesaurus' genaamd) zijn weer moeilijk te onderhouden. Dit alles leidt ertoe dat de gebruiker veel tijd en moeite moet investeren om de voor hem interessante documenten te vinden.
Om dit op te lossen is een systeem vereist dat bepaalde kennis over de onderwerpen van de documentencollectie zelf opbouwt, zoals kennis over de semantiek van de woorden die in de documenten voorkomen. Het WEBSOM-algoritme heeft die eigenschappen maar is daarentegen niet in staat om grote documenten zoals boeken te verwerken.
Het onderzoek dat ten grondslag ligt aan deze scriptie is gericht op het ontwerpen en implementeren van een algoritme dat ongeveer dezelfde functionaliteit zou moeten bieden als het WEBSOM-algoritme wat betreft toegangssnelheid, gebruikersvriendelijkheid, selectieprecisie en vindkracht maar die dit realiseert op basis van een index in plaats van de volledige tekst. Het beoogde systeem hoeft niet op self-organizing maps te zijn gebaseerd. De complete formulering van de doelstelling van deze scriptie is dan als volgt:

> Het ontwerpen en implementeren van een algoritme ten behoeve van een informatie-zoeksysteem dat, met behulp van domeinkennis die opgebouwd is op basis van indexen, een hoge mate van selectieprecisie en vindkracht bereikt.

Als randvoorwaarden geldt dat de toegangssnelheid en de gebruikersvriendelijkheid niet minder mag zijn dan een traditioneel informatie-zoeksysteem.

1.3 Methodiek

De doelstelling van deze scriptie vraagt om een informatie-zoeksysteem dat beschikt over domeinkennis. (Domein-)kennis is echter een breed begrip dat vele betekenissen heeft en waar bijna geen eenduidige mening over bestaat:
"Unfortunately, there is still so little general agreement, specially in semantics, about the aim and the precise nature of the subject, and about the models of description to be used, that much of the discussion is more philosophical than scientific."(Palmer 1981)
Deze scriptie zal dan ook beginnen met een bespreking van verschillende visies op kennis die inderdaad meer filosofisch is dan wetenschappelijk. Deze ideeën worden echter concreet gemaakt door te kijken naar de wijze waarop deze ideeën hun neerslag hebben in kennisrepresentatie in computers, zodat een duidelijk beeld ontstaat op welke wijze kennis kan worden gerepresenteerd ten behoeve van een informatie-zoeksysteem.
Om een goed informatie-zoeksysteem te ontwerpen doet men er goed aan bestaande ervaringen en bevindingen mee te nemen. Deze liggen vastgelegd in de ontwerpen van de huidige informatie-zoeksystemen en deze systemen zullen dan ook worden onderzocht, waarbij tevens een maatstaf wordt bekeken waarmee de prestaties van deze systemen kunnen worden geëvalueerd.
Onder deze informatie-zoeksystemen bevinden zich al enkele systemen welke gebruik maken van zelfopgebouwde domeinkennis zoals het WEBSOM-algoritme. Deze zullen apart worden bekeken, waarbij zal worden nagegaan of in deze systemen een algemene architectuur is te vinden waarvan is af te leiden hoe zo'n systeem werkt en uit welke onderdelen het moet bestaat. Tevens zal worden bekeken wat de sterke en zwakke punten zijn van deze systemen.
Aan de hand van deze bevindingen is het dan mogelijk een algoritme te ontwerpen dat voldoet aan de doelstelling van deze scriptie. Dit algoritme maakt gebruik van een grotendeels nieuwe vorm van kennisrepresentatie die Associatieve Conceptuele Ruimte (ACR) genoemd zal worden. Het algoritme dat in staat is kennis te onttrekken uit indexen van documenten ten behoeve van een informatie-zoeksysteem zal het ACR-WEBSOM-algoritme worden genoemd.
Van het ACR-WEBSOM is een prototype geschreven welke wordt getest met behulp van een testset uit de praktijk. Bij het testen worden de belangrijkste aannamen die ten grondslag liggen aan het ACR-WEBSOM-algoritme geverifieerd. Ter afsluiting wordt het gepresenteerde algoritme geëvalueerd en zullen aanbevelingen worden gegeven voor verder onderzoek.

1.4 Opbouw

In hoofdstuk 2 zal eerst het begrip 'kennis' worden onderzocht, waarbij enkele ideeën op dit gebied worden gepresenteerd. Deze ideeën zijn op te delen in twee hoofdklassen, welke 'klassiek' en 'modern' zullen worden genoemd. In hoofdstuk 3 blijkt dat deze twee hoofdklassen ook hun tegenhangers hebben in de wijze waarop kennis wordt gerepresenteerd in computers.
Hoofdstuk 4 begint met de bespreking van een methode waarop informatie-zoeksystemen kunnen worden geëvalueerd. Vervolgens zullen in dit hoofdstuk een breed scala van informatie-zoeksystemen worden behandeld.
In hoofdstuk 5 zal in groter detail een tweetal informatie-zoeksystemen worden onderzocht welke reeds gebruik maken van zelfopgebouwde domeinkennis. Aan de hand van dit onderzoek zal worden bepaald wat de algemene eigenschappen van een dergelijk systeem

dienen te zijn. Als laatste zal in dit hoofdstuk een eerdere poging van N.Weustink tot het ontwerpen van een systeem dat gebruikt maakt van kennis welke onttrokken wordt uit de indexen van boeken worden behandeld en wat de tekortkomingen waren van dit systeem.
Vervolgens zal in hoofdstuk 6 een nieuwe vorm van kennisrepresentatie, Associatieve Conceptuele Ruimte genaamd, worden gepresenteerd en zal het ACR-WEBSOM-algoritme uiteen worden gezet, een algoritme voor een informatie-zoeksysteem dat in staat is kennis te onttrekken uit indexen van boeken.
In hoofdstuk 7 zijn de praktijktests en bijbehorende resultaten beschreven, waarna in hoofdstuk 8 enkele conclusies ten opzichte van het ACR-WEBSOM-algoritme worden genomen en tevens enkele aanbevelingen worden gedaan voor verder onderzoek in dit gebied.

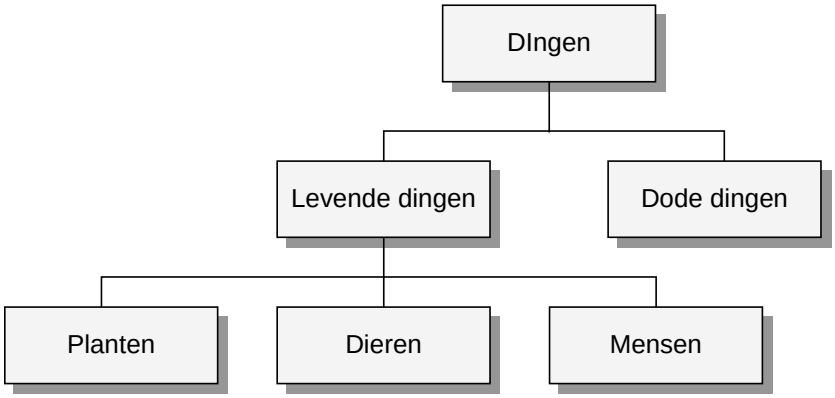
Hoofdstuk 2: Kennis

*“Rationality is not a property of men, nor a fact about men. It is **a task for men to achieve**. A difficult and severely limited task. It is difficult to achieve rationality even partially.”*
(Popper, p.134)

Om een informatie-zoeksysteem uit te rusten met kennis moet duidelijk zijn wat dit inhoudt.
In dit hoofdstuk wordt onderzocht wat bedoeld wordt met de term ‘kennis’ en wat het inhoudt om te zeggen dat je ‘kennis hebt over iets’. Dit onderwerp is al enkele millennia oud, maar zoals uit dit hoofdstuk zal blijken, nog steeds actueel. Inzicht in het begrip kennis is essentieel voor het ontwerpen van een nieuw algoritme ten behoeve van een informatie-zoeksysteem. In het volgende hoofdstuk worden enkele implementaties van de kennisideeën uit dit hoofdstuk beschreven. Een informatie-zoeksysteem vraagt echter om een specifiek soort kennis en het zal blijken dat geen van de in hoofdstuk 3 beschreven bestaande implementaties hier meteen volledig aan voldoen.
Als eerste zal het klassieke kennisidee worden toegelicht (2.1), waarna ik zal ingaan op enkele moderne kennisideeën (2.2). Uiteindelijk worden deze twee ideeën tegenover elkaar gezet (2.3).

2.1 Het klassieke kennisidee

Dit kennisidee krijgt het predikaat ‘klassiek’ omdat het stamt uit de klassieke Griekse filosofie. Met name Aristoteles zou men kunnen aanwijzen als de grondlegger van deze leer. Daarvoor was Plato echter al begonnen met enkele van de basisconcepten. Hij maakte onderscheid tussen ‘kennis’ en ‘mening’, waarbij kennis betrekking heeft op iets dat onveranderlijk is. Een mening, zoals mooi of lelijk, groot of klein en licht of zwaar is daarentegen wel veranderlijk. We kunnen iets wat groot is in een ander opzicht bijvoorbeeld ook klein noemen. Meningen “hebben betrekking op veranderlijke dingen waarvan geen kennis mogelijk is” (Plato, p.142). Kennis is dus volgens deze definitie objectief.
Aristoteles onderkende in kennis een zekere structuur. Kennis gaat over dingen, kennis heeft altijd een object. Maar in de werkelijkheid is een oneindig aantal verschillende individuele dingen. Hierin schuilt dus een probleem.
“If, on the one hand, there is nothing apart from individual things, and the individuals are infinite in number, how is it possible to get knowledge of the infinite individuals ?” (Aristoteles, P.1578, Metaphysics)
De individuele dingen hebben echter vaak dingen gemeen met elkaar. We kunnen zeggen dat ze tot een bepaalde categorie behoren. Plato dacht dan ook dat ze afgeleid waren van eeuwige Vormen, dat ieder object slechts als een schaduw is van een diepere werkelijkheid. Dingen zijn echter niet tot één bepaalde categorie beperkt: “And there will be several patterns of the same thing, and therefore several Forms, e.g. animal and two-footed and also man himself will be the Forms of man.” (Aristoteles, p.1567, Metaphysics).
Kennis is een structuur die door de mens is aangelegd. In kennis kon Aristoteles een hiërarchie aan te brengen, zoals blijkt uit het voorbeeld in *figuur 2.1*. (Gaarder, p.125)



Figuur 2.1: Voorbeeld van een hiërarchie van concepten

De verschillende categorieën noemen we ook wel concepten. In dit geval is de relatie tussen de concepten van het type ‘is een’. Oftewel, we kunnen zeggen: ‘mensen’ zijn ‘levende dingen’. De eigenschappen van een bepaald concept gelden dan ook voor alle onderliggende concepten. Met deze structuur kan men vervolgens nieuwe feiten over de realiteit te weten komen. Hiervoor zijn twee inferentie-instrumenten beschikbaar: deductie en inductie.

- Via deductie kunnen we binnen de structuur gebruik maken van de bestaande kennis om nieuwe kennis te verwerven, bijvoorbeeld: Socrates is een mens, alle mensen zijn sterfelijk, dus (deductie) Socrates is sterfelijk.

- Inductie daarentegen maakt gebruik van een groot aantal van de individuele dingen in de realiteit om geheel nieuwe kennis toe te voegen, bijvoorbeeld: alle mensen waarvan ik heb gehoord gaan uiteindelijk dood, dus (inductie) alle mensen zijn sterfelijk.

(er bestaat in ieder geval nog één andere inferentiemethode naast deductie en inductie: abductie (Charniak e.a, p.21ff). Dit is echter een zwakkere vorm van inferentie en zal snel leiden tot foute conclusies. Een voorbeeld van abductie is: alle mensen zijn sterfelijk, Socrates is sterfelijk ,dus (abductie) Socrates is een mens.)

Zowel deductie als inductie maken gebruik van de logica. Logica stamt ook uit de Griekse oudheid en voor een lange tijd dacht men dat Aristoteles hiermee de wetten van de menselijke gedachte had gevangen. Zijn logische systeem bevatte drie proposities:
1. de wet van de identiteit (wat betekend dat “één is één”, “twee is twee” etc.)
2. de wet van non-contradictie (een stelling en zijn negatie kunnen niet tegelijkertijd waar zijn)
3. de wet van het uitgesloten midden (een stelling is óf waar óf niet waar, hij kan niet tegelijkertijd beide zijn).

(Coveney e.a., p.22)

De in deze paragraaf beschreven structuur is sinds de oudheid kenmerkend voor Westerse kennis. Mechanische assemblages, software, wetenschappelijke en technische kennis zijn op deze manier gestructureerd (Pirsig, p.87), uiteraard in vormen die vele malen complexer zijn dan hierboven beschreven. Deze systematiek staat bekend onder de noemer ‘rationaliteit’ en zonder deze rationaliteit zou de moderne wetenschap en haar produkten zoals televisies en maanraketten niet bestaan.

2.2 Een modern kennisidee

Het moderne kennisidee dat in deze paragrafen wordt beschreven stamt uit deze eeuw. Het is gebaseerd op de bevindingen in de neurologie en het kunstmatige intelligentie onderzoek. Kennis wordt gezien als een produkt van onze hersenen en om het beter te begrijpen moeten we inzicht hebben in de werking van de hersenen.
In deze paragraaf zal eerst worden geschetst hoe kennis door evolutie is ontstaan (2.2.1), waarna de biologische werking van het brein aan de orde komt (2.2.2). Vervolgens zal worden bekeken hoe kennis in het brein zou kunnen zijn gerepresenteerd (2.2.3).

2.2.1 Kennis en evolutie

Om te begrijpen hoe kennis in elkaar zit moeten we weten hoe het is ontstaan en welk nut het heeft. De ideeën in deze paragraaf zijn grotendeels ontleend aan de filosoof Karl Popper. Popper stelt om te beginnen, net als Aristoteles en Plato, dat er twee werelden zijn, namelijk:

- 'Wereld 1': de wereld van fysieke dingen en hun fysieke en fysiologische staten

- 'Wereld 2': de wereld van mentale staten, van subjectieve gedachten

In dit geval valt kennis duidelijk onder wereld 2. We kunnen kennis echter opdelen in twee typen kennis, namelijk subjectieve en objectieve kennis.
Subjectieve kennis is wat een persoon weet. Bijvoorbeeld:

'*Ik weet* dat water is opgebouwd uit waterstof en zuurstof.'
'*Hij zag* een gele schijf.'

Objectieve kennis is 'algemene kennis', zoals wetenschap. Bijvoorbeeld:

'Het is *algemeen bekend* dat water uit waterstof en zuurstof is opgebouwd'
'Het is *algemeen bekend* dat de aarde om de zon draait.'

Deze kennis is niet afhankelijk van een persoon en we kunnen dan ook zeggen dat deze een eigen bestaansrecht, een eigen wereld heeft:

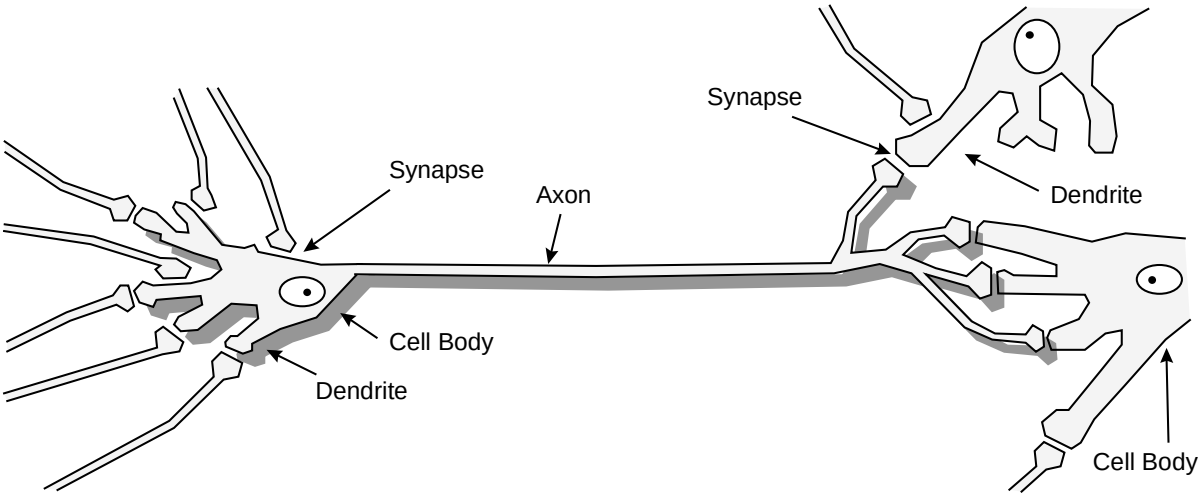
- 'Wereld 3': produkten van het brein, zoals wetenschap, literatuur en taal.

Zo kunnen we bijvoorbeeld zeggen dat 'Hamlet' van Shakespeare behoort tot wereld 3. Als we Hamlet zouden lezen dan zou dit in de eerste plaats op papier staan, dus zijn neerslag hebben in wereld 1. Hamlet is echter onafhankelijk van het boek waar het in geschreven staat. Ook is een voorstelling van Hamlet in het theater niet 'Hamlet' zelf, maar slechts een afgeleide of instantie ervan.
Hiermee wordt niet bedoeld dat wereld 3 compleet reëel is, in tegenstelling tot bijvoorbeeld het idee van de eeuwige en goddelijke Vormen van Plato. "I Would say that really the name 'world 3' is just a way of putting things, and the thing is not to be taken too seriously." (Popper, p.17) Het concept 'wereld 3' is echter wel een belangrijk hulpmiddel bij het begrijpen van de ontwikkeling van kennis.
Een organisme heeft zijn gedrag te danken aan de invulling van zijn wereld 2. Bij eenvoudige organismen is deze wereld 2 grotendeels aangeboren en kan het individuele organisme hier zelf niets aan veranderen. Een vlieg leeft bijvoorbeeld in de (impliciete) veronderstelling dat zijn overlevingskansen groter zijn als hij zich beweegt naar plekken waar meer licht is. Als er zich situaties voordoen waar dit niet opgaat, zoals bij een hete lamp of een gesloten raam, zal de vlieg zijn eigen gedrag niet aan kunnen passen en zal hij waarschijnlijk sterven. Mocht er, door willekeurige mutatie, een vlieg ontstaan die het ingeboren gedrag heeft dat hij lampen en ramen kan ontwijken, dan geeft dit hem een voordeel in de overlevingsstrijd en zullen er volgens de evolutieleer na verloop van tijd bijna uitsluitend vliegen zijn die dit gedrag vertonen. We zien hier dus een verandering in wereld 2 van een vlieg door natuurlijke selectie van de individuen. Het soort 'vlieg' heeft iets geleerd, heeft kennis opgedaan.

Deze vorm van leren zien we echter nauwelijks meer bij mensen. In plaats van te sterven met onze foute veronderstellingen, kunnen we deze toetsen met behulp van eerdere kennis en deze veronderstellingen oftewel theorieën in onze plaats laten sterven. Deze theorieën zijn niet meer een vast onderdeel van onze fysiologie., hoewel natuurlijk de capaciteit om theorieën te vormen en te evalueren wel aangeboren is. We zien gelijk wat het voordeel dit geeft in het gevecht om het bestaan. Mensen zijn beter in staat om zich aan te passen aan extremen dan welk andere organisme. Mensen kunnen zelfs (tijdelijk) op het maanoppervlak leven met behulp van hun technologie, en technologie behoort tot wereld 3.
Het is echter belangrijk om te beseffen dat veruit de grootste hoeveelheid kennis niet door één mens zelf wordt ontwikkeld. Het is verkeerd om te veronderstellen dat alles wat een mens weet is afgeleid uit zijn directe waarnemingen van de werkelijkheid. Dit is het belang van wereld 3 voor de mensheid. “As for subjective knowledge, much of it is simply taken over from objective knowledge. We learn a great deal from books, and in universities” (Popper, p.12). Dit betekend dat wanneer we een Informatie-Zoeksysteem, of elk ander systeem, uit willen rusten met kennis dit vooral moet gebeuren met behulp van wereld 3 oftewel vanuit boeken en andere publikaties.

2.2.2 Werking van de hersenen

Om te begrijpen hoe mensen kennis kunnen vormen en na kunnen denken moeten we weten hoe hersenen werken. En om te begrijpen hoe hersenen werken moeten we eerst iets weten over de werking van hun onderdelen: de neuronen. De menselijke hersenen zijn opgebouwd uit zo'n 100 miljard neuronen die met elkaar verbonden zijn met in totaal ongeveer 100 triljoen *synapses*. (Churchland, p.4) In *figuur 2.2* is een neuron weergegeven:



Figuur 2.2: Een schematische weergave van een neuron. De elektrische signalen stromen via de *dendrites* binnen en via de *axon* verder naar andere neuronen. In dit diagram stroomt de informatie dus van links naar rechts. (aangepast van Crick, p.92)

Er zijn veel verschillende typen neuronen maar de meeste werken echter volgens hetzelfde principe. Een typische neuron kan op drie manieren reageren op de signalen die hij ontvangt via de synaptische ingangen op zijn cellichaam en vertakkingen ofwel 'dendrites.' Sommige signalen stimuleren de neuron, andere remmen hem en sommige veranderen zijn gedrag. Als de neuron voldoende gestimuleerd wordt zal deze zelf ook een signaal afgeven, wat via de *axon* wordt doorgegeven en middels de aan de *axon* verbonden *synapses* wordt overgedragen aan andere neuronen.
Dit is dus de hoofdfunctie van een neuron. Hij ontvangt informatie, meestal in de vorm van elektrische signalen, van andere neuronen. Hierna wordt in feite een complexe dynamische som van deze input gemaakt en het resultaat word doorgegeven aan andere neuronen.
Het is voor te stellen dat een signaal via een gesloten circuit van neuronen zichzelf een tijd lang in stand kan houden. Op die manier kan informatie worden opgeslagen, maar dit is altijd tijdelijk van aard. Informatie moet dus ook op een meer structurele manier worden opgeslagen.
De neuronen zijn verbonden door hun *synapses*. Een *synaps* is in feite een smalle scheiding tussen de ene neuron en de andere waar informatie wordt overgegeven middels het uitwisselen van bepaalde stoffen. *Synapses* zijn op te delen in twee soorten: stimulerende en remmende (Crick, p.98). Dit betekend dat de activering van één neuron het activeren van de volgende neuron respectievelijk kan stimuleren of kan afremmen.
De sterkte of het 'gewicht' van de synaptische verbinding bepaald de mate waarin het signaal wordt overgebracht. Een neuron dat een sterke verbinding heeft met een volgende neuron kan deze dus sneller tot activering over laten gaan. Het zijn deze synaptische

gewichten die de informatie representeren die in onze hersenen zijn opgeslagen en ons gedrag en denken bepalen. Een deel van deze gewichten zijn bij onze geboorte, dus genetisch, bepaald. Bij het voorbeeld van de vlieg uit de vorige paragraaf is dit een aanzienlijk groter deel dan bij mensen.
De overige gewichten worden vanaf de geboorte langzaam veranderd volgens een vast leerprincipe, zodat een normaal mens onder normale omstandigheden eerst onder andere leert te zien, te lopen en te praten en vervolgens ook verdere cognitieve vaardigheden ontwikkelt. Als antwoord op de vraag wat dit leerprincipe nu precies inhoudt gaf de Canadese psycholoog Donald Hebb al in 1949 al een speculatief antwoord:

“When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased” (Hebb, p.62)

Dit houd in dat wanneer twee neuronen gelijktijdig actief zijn de synaptische verbinding ertussen wordt versterkt. Dit verschijnsel staat bekend onder de term ‘long-term potentiation’ of LTP en in recent onderzoek is gebleken dat dit leermechanisme zich inderdaad ook bij zoogdieren voordoet (Johnston). Deze veranderingen in de gewichten blijven over een lange termijn in stand en vormen de basis voor ons geheugen.
Zoals we hebben gezien zijn neuronen hele eenvoudige processors vergeleken bij een CPU van een moderen seriële computer. Daarnaast is de neuron ook nog bijzonder traag. Een bewerking in een neuron duurt ongeveer 10 milliseconden, wat in de praktijk neerkomt op een ‘kloksnelheid’ van 40 to 100 hertz, tegenover de kloksnelheid van een gemiddelde PC van 133 miljoen hertz. Ook de betrouwbaarheid van neuronen laat veel te wensen over.
Toch is het menselijk brein op bepaalde punten veruit superieur aan de meest geavanceerde supercomputer. Bijvoorbeeld zoiets als op twee benen rennen door een bos met behulp van de gegevens van twee ogen is (nog) niet door een seriële computer te doen. De kracht van de hersenen zit in het feit dat deze alle bewerkingen parallel uitvoert. 40 tot 100 keer per seconden voert het brein 100 miljard bewerkingen naast elkaar uit. Tevens zorgt deze parallelle bewerking voor een ongekende robuustheid. Een seriële computer is zo betrouwbaar als zijn minst betrouwbare onderdeel. De hersenen hebben daarentegen bijvoorbeeld geen last van de ca. 10.000 neuronen die we per dag kwijtraken.

“An army of fumbling tortoises, by an artful strategy, manages to outrun the hare.” (Churchland, p.14)

2.2.3 Kennis in de hersenen

In de vorige paragraaf is beschreven hoe de configuratie van de gewichten van de *synapses* ons langetermijn geheugen vormen. In deze paragraaf zullen we kijken wat voor vormen dit geheugen aan kan nemen. Om te beginnen kunnen we geheugen in drie categorieën opdelen: (Crick, p.67)

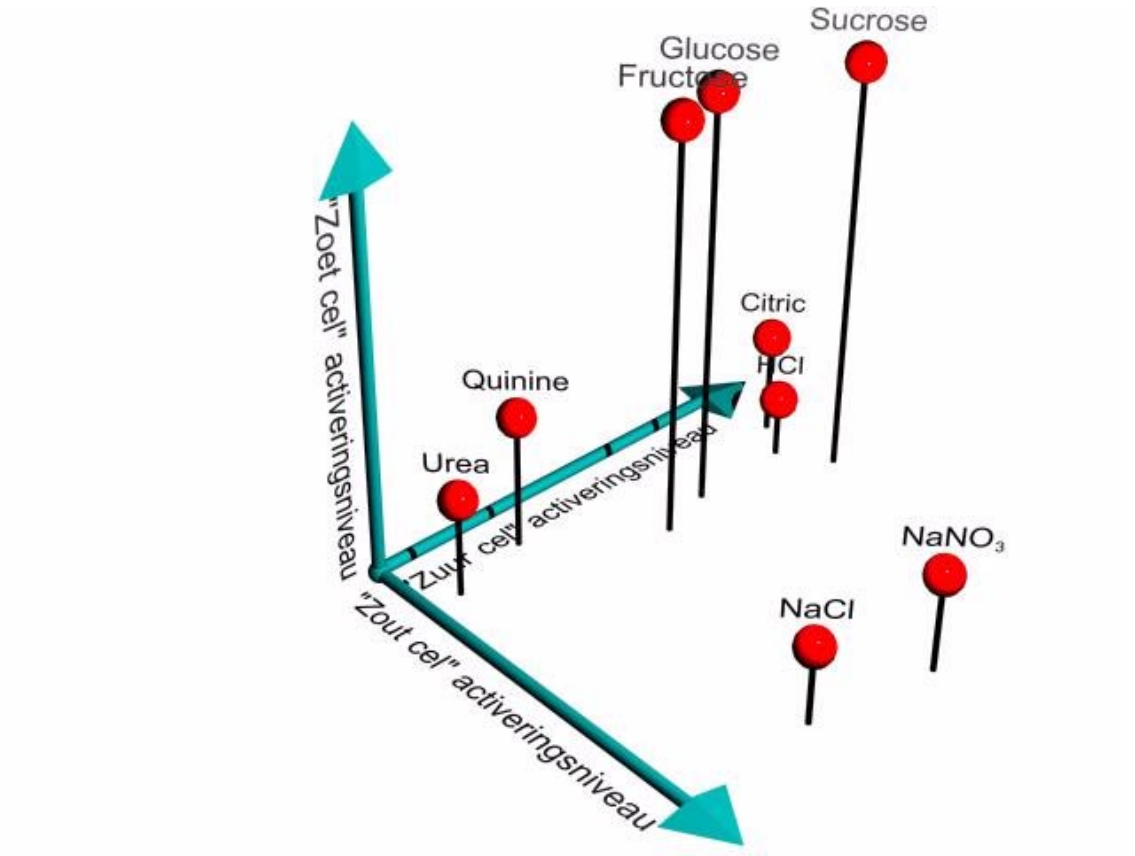
- **Episodisch geheugen** zorgt ervoor dat we ons bepaalde gebeurtenissen kunnen herinneren, zoals bijvoorbeeld de dag wanneer we voor het eerst rijles hebben gekregen.
- **Procedureel geheugen** betreft al onze aangeleerde en aangeboren vaardigheden, zoals het kunnen besturen van een auto in het drukke verkeer.
- **Categoriaal geheugen** bevat de meer permanente informatie die we over de buitenwereld hebben. Een voorbeeld zou kunnen zijn de kennis die iemand heeft over de werking van een auto, de motor en het schakelmechanisme, of kennis over verbranding van fossiele brandstoffen.

Episodisch geheugen is in die zin bijzonder dat het meestal in één keer in het geheugen wordt opgeslagen. Daarentegen moet procedureel geheugen vaak meerdere keren worden geconfronteerd met de te leren vaardigheid, net zoals categoriaal geheugen vaak meerdere keren de feiten moet doorlopen.
Over de werking van episodisch geheugen is nog weinig bekend, behalve dat het nauw verbonden is met de hippocampus, een gebied in de hersenen (Churchland, P.165,166).
De manier waarop de hersenen informatie representeren kunnen we simplistisch voorstellen als een vector in een (hoogdimensionale) ruimte. Om dit idee toe te lichten zullen we het voorbeeld van smaak-representatie gebruiken:
Mensen zijn berucht om het slecht omschrijven van smaken. Onderzoek heeft uitgewezen dat de mens slechts vier verschillende smaak-receptoren heeft, die we ‘zoet’, ‘zuur’, ‘zout’ en ‘bitter’ kunnen noemen, aangezien een stof die bijvoorbeeld bijna uitsluitend de ‘zoet cel’ activeert ook als zoet wordt omschreven. In de activering van deze cellen is een bepaalde gradatie mogelijk. Een smaak is dus te omschrijven met behulp van de vier activeringsniveau’s, zoals bijvoorbeeld in *figuur 2.3*.

Figuur 2.3: Smaak als vier activeringsviveau’s

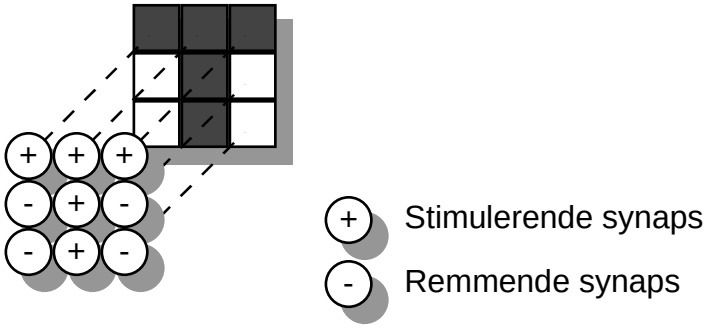
Hier zien we gelijk waarom een smaak zo moeilijk is om te omschrijven. Als deze receptoren maar tien verschillende gradaties zouden kunnen onderscheiden, dan is mogelijk om 10^4 verschillende smaken te herkennen. Het aantal woorden in onze taal schiet daarbij dus tekort. Meestal gebruiken we dan een vergelijking met een andere smaak. Als we een aantal

smaken in een figuur weergeven, zoals in *figuur 2.4*, dan kunnen we dit ons voorstellen als het vinden van een bekende smaak die de kleinste afstand heeft tot de expliciet te maken smaaksensatie. Als we zeggen dat fructose smaakt als glucose dan bedoelen we dat deze smaken in deze ruimte dicht bij elkaar liggen.



Figuur 2.4: Smaak-ruimte met de posities van een aantal bekende smaken. **De 'bitter' dimensie is niet weergegeven voor overzichtelijkheid.**
(aangepast van Churchland, p.24)

Hoe de hersenen de verschillende smaken kunnen herkennen met behulp van neuronen kunnen we ons als volgt voorstellen. De smaak-receptoren zijn verbonden met meerdere neuronen en de neuron met de synaptische configuratie die het meest aansluit bij de smaak-vector wordt geactiveerd, waarbij misschien andere neuronen hierdoor meteen worden geremd, zodat we uiteindelijk één duidelijke smaak kunnen herkennen.
Bij het herkennen van visuele aspecten is dit duidelijk voor te stellen: Het netvlies is opgebouwd uit een groot aantal receptoren, die afhankelijk van de hoeveelheid licht die erop valt een signaal afgeven aan een grote verzameling neuronen in de visuele cortex, een gebied aan de achterkant van de hersenen. Deze neuronen hebben zo'n synaptische configuratie dat zij alleen op bepaalde patronen reageren. Een voorbeeld hiervan is weergegeven in *figuur 2.5*. Hier zien we de schematische weergave van de synaptische configuratie van een neuron die geactiveerd wordt wanneer zich een 'T'- patroon voordoet op een stukje van het netvlies van 3 bij 3 receptoren.



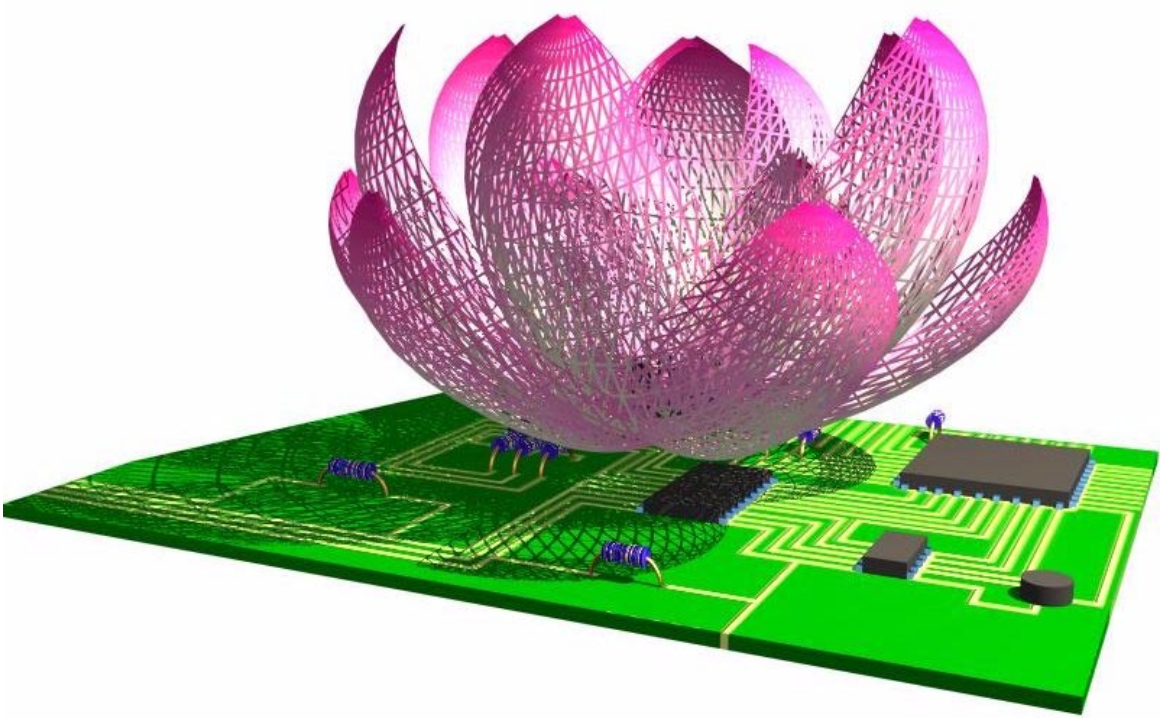
Figuur 2.5: een neuron die volgens deze configuratie is verbonden met het netvlies zal maximaal geactiveerd zijn bij dit 'T' -patroon. (Churchland, p. 37)

We kunnen zeggen dat deze cel een 'T-detector' is. Hiernaast zullen nog vele andere patroon-specifieke neuronen in dit gebied aanwezig zijn. In een tweede laag kunnen de outputs van deze bovenliggende neuronen dan gevoelig zijn voor patronen in de activering van de eerste laag. Als in de eerste laag bijvoorbeeld een aantal letters worden gedetecteerd, dan kan in de tweede laag hierin een woord worden herkend.
Het is aannemelijk dat het procedurele geheugen ongeveer op deze manier werkt: een eerste laag herkend patronen in een input, zoals hoeken en lijnen op het netvlies. Een volgende laag kan hier weer een patroon in herkennen, zoals de afbeelding van een verkeerslicht dat op rood staat. Volgende lagen herkennen de handelingen die bij dit patroon horen, zoals het trappen op de rem, waarna deze signalen worden doorgegeven aan de spieren. Op deze manier ontstaat een koppeling tussen stimulus en respons.
Het ontstaan van deze 'hiërarchie' van neuronen kunnen we ons als volgt voorstellen: Als mensen voor het eerst gaan zien en naar een driehoek kijken, dan doen ze dit stuk voor stuk alle hoekpunten apart te bekijken. Dan geldt volgens Hebb:
"When A, B and C are looked at successively, in any order, but in a short period of time, activity may continue by reverberation in two of the structures while the third is sensorily aroused.... According to the assumptions made earlier, simultaneous activity in a, b, and c would establish facilitation between them....The resulting superordinate system must be essentially a new one, by no means a sum or hooking together of a, b and c." (Hebb, p.96,97).
Door herhaling ontstaat dus langzaam een nieuw systeem dat geactiveerd wordt wanneer we een driehoek zien. Het concept 'driehoek' is ontstaan. Indien tegelijkertijd het woord 'driehoek; wordt gehoord, dan zal na herhaling ook een verband ontstaan tussen het woord en het figuur.
Deze koppeling tussen taal en zintuigelijke waarneming is mijns inziens zeer belangrijk. In paragraaf 2.2.1 hebben we gezien dat taal behoort tot wereld 3, terwijl onze zintuigen in de eerste plaats wereld 1 waarnemen. Zoals gezegd komt het grootste gedeelte van de kennis die opgeslagen ligt in onze hersenen, de subjectieve kennis in wereld 2, van de objectieve kennis in wereld 3. Door taal kunnen we verdere, abstractere concepten ontwikkelen, zoals de relativiteitstheorie en quantummechanica. We moeten echter niet vergeten dat deze concepten net als die van de driehoek zijn opgebouwd uit simpelere, onderliggende concepten die eerder zijn ontstaan. In de woorden van Hebb: "The classical dispute of physics about the nature of light was really asking, Is light like a shower of pebbles, or like ripples in a bathtub?" (Hebb, p.119)

We kunnen nu ook stellen dat we een speculatief beeld hebben van categoriaal geheugen. Het bestaat uit een aantal concepten die aan elkaar verbonden zijn met verbindingen die allerlei gradaties kennen. Dit netwerk komt het meest in de buurt bij het idee van kennis zoals dit in paragraaf 2.1 is beschreven, alhoewel duidelijk mag zijn dat de regels van logica wel in een dergelijke structuur opgeslagen kunnen worden, maar dat zij niet het gedrag van het opslagmedium bepalen.

2.3 Klassieke en moderne kennis

In paragraaf 2.1 en 2.2 zijn twee verschillende kennisideeën behandeld. Het klassieke kennisidee gaat uit van de rationaliteit waarbij wordt verondersteld dat ons brein volgens vaste regels te werk gaat. Het moderne kennisidee toont echter aan dat het menselijk brein misschien wel in staat is tot rationaliteit, maar dat de regels van rationaliteit niet de grondbeginselen van het menselijk denken vormen.
Rationaliteit vereist duidelijkheid, vraagt om zwart-wit stellingen, maar het brein werkt eigenlijk met name in ‘grijstinten’, in sterke en minder sterke koppelingen tussen concepten. Het werd vroeger ook niet als wenselijk beschouwd dat rationaliteit vaagheid kon bevatten. Gebleken is echter dat in bepaalde situaties vaagheid een voordeel kan geven. Dit zijn situaties waarvan ook wel wordt gezegd dat we ‘intuïtie’ nodig hebben om een oplossing te vinden. Dit heeft onder andere geleid tot het ontstaan van ‘fuzzy logic’, een vorm van logica waarbij de wet van het uitgesloten midden is komen te vervallen (Coveney, p.74).
Rationaliteit biedt de mogelijkheid om een stelling kritisch te onderzoeken en is de basis van alle wetenschappelijke kennis. In bepaalde gevallen is rationaliteit echter gedeeltelijk overbodig. Een informatie-zoeksysteem hoeft niet de inhoud van de documenten kritisch te analyseren, maar moet wel een intuïtieve schatting kunnen maken van de onderwerpen die de documenten behandelen en op basis hiervan de relevantie voor de gebruiker kunnen evalueren.
Verder is in paragraaf 2.2.1 gebleken dat het niet noodzakelijk is om kennis over de wereld te verkrijgen via een directe verbinding met die wereld: de meeste kennis die een mens heeft is afkomstig uit materialen zoals boeken en tijdschriften. Dit gegeven is essentieel voor de haalbaarheid van een systeem dat kennis moet opbouwen.
Het klassiek en moderne kennisidee hebben ook hun neerslag op de kunstmatige intelligentie en de implementatie hiervan in software. In hoofdstuk 3 zal hier dieper op in worden gegaan.



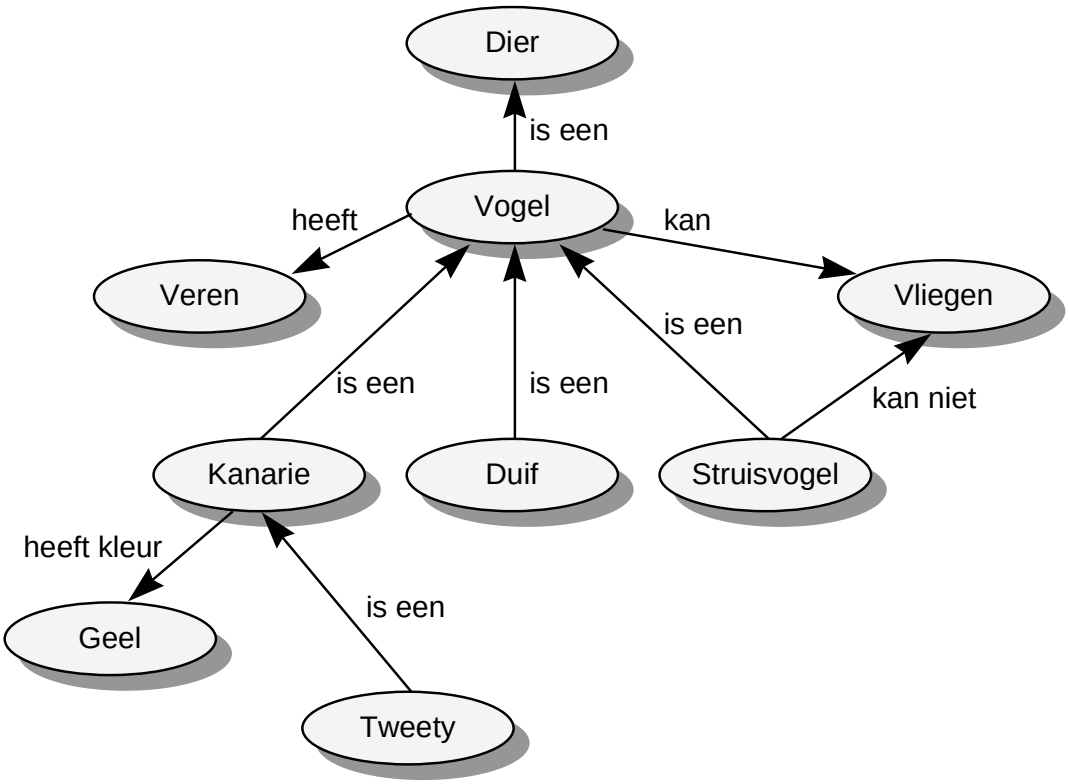
Hoofdstuk 3: Kennisrepresentatie in een computer

*"As we consider various cognitive structures and their limitations, we notice that **any** cognitive structure or vehicle or carrier of knowledge will have **it's own** physical characteristics peculiar to it **as a vehicle** rather than being characteristic of the object to be represented. " (Radnitzky e.a., p.34)*

Nadat in hoofdstuk 2 het onderwerp kennis in het algemeen aan de orde is geweest, zal in dit hoofdstuk worden bekeken hoe kennis gerepresenteerd kan worden in een computer. Hierbij zal blijken dat de twee kennisideeën, de klassieke en de moderne, ook hun parallellen hebben in de kunstmatige intelligentie: respectievelijk de symbolische en de connectionistische kunstmatige intelligentie. Eerst zal de symbolische methode behandeld worden (3.1), waarna een tussenvorm aan bod komt, namelijk connectionistische semantische netwerken met spreading activation (3.2). Vervolgens worden enkele volledige connectionistische methoden behandeld die gebruik maken van conceptuele ruimte (3.3). Tenslotte volgen nog enkele conclusies (3.4).

3.1 Symbolische kennissystemen

Symbolische kennissystemen zijn in het algemeen gebaseerd op het klassieke kennisidee zoals behandeld in paragraaf 2.1. De structuur van de kennis in deze systemen is vergelijkbaar met de hiërarchie van concepten. In deze hiërarchie kunnen ook de eigenschappen van de verschillende concepten worden weergegeven. Hierdoor ontstaat een structuur zoals in *figuur 3.1*.



Figuur 3.1: Voorbeeld van een semantische netwerk (aangepast van Crestani, p.458)

Deze structuur wordt ook wel een semantische netwerk genoemd. Ieder concept is weergegeven als een knoop en de hiërarchische relaties tussen de concepten zijn weergegeven met behulp van ‘is een’ verbindingen (Shastri, p.15). Knopen op het laagste niveau representeren individuen terwijl knopen op een hoger niveau klassen en categorieën voorstellen (Quillian, p.234f). Abstractere concepten staan dus hoger in de hiërarchie. We kunnen ook zeggen dat lager gelegen concepten elementen zijn van de bovenliggende verzameling. Zo is ‘duif’ een element uit de verzameling van vogels.
Eigenschappen van concepten worden óók als knopen aangegeven en deze eigenschappen gelden ook voor alle concepten die middels een ‘is een’ verbinding aan dit concept zijn gekoppeld en lager in de hiërarchie staan. Eigenschappen zijn middels een niet-hiërarchische verbinding zoals ‘kan’ of ‘heeft’ gekoppeld aan de concepten. In feite zijn eigenschappen zelf ook concepten en kunnen zelf ook onderdeel uitmaken van een hiërarchie. ‘Geel’ is bijvoorbeeld ondergeschikt aan het concept ‘kleur’
De volledige semantische betekenis van een woord kan nu als volgt worden gedefinieerd: “a word’s full concept is defined in the memory model to be all the nodes that can be reached by an exhaustive tracing process, originating at its initial, patriarchical type node, together

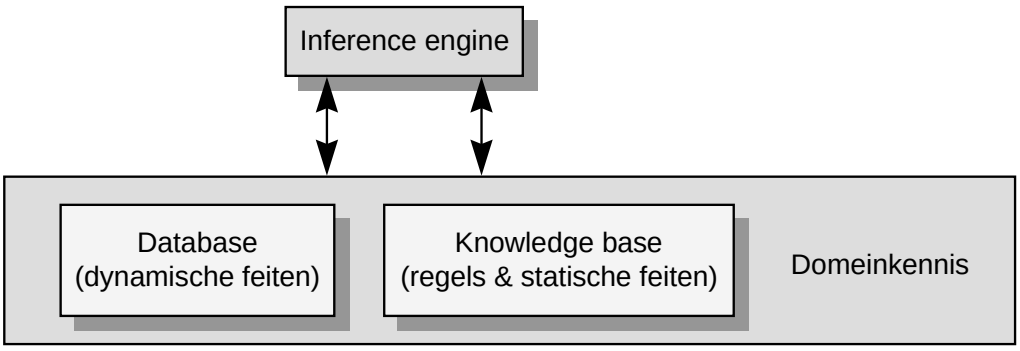
with the total sum of relationships among these nodes" (Quillian, p.238) De volledige betekenis van een woord is dus dat woord en alle woorden die vanaf dat woord kunnen worden bereikt, inclusief de relaties die die woorden tot elkaar hebben.
Semantische netwerken kunnen worden gezien als een grafische vorm van een ander soort kennis-representatie, namelijk 1^e-orde logica (Shastri, p.20), (Steels, p.146). Bij deze methode wordt de kennis gerepresenteerd in een logische taal zoals PROLOG. Als voorbeeld kunnen we de kennis uit het eerder behandelde semantisch netwerk met zo'n logische notatie weergeven:

$$\forall\ (x)\ \text{vogel}(x) \rightarrow \text{dier}(x)$$

Oftewel iedere vogel is een dier, en

$$\forall\ (x)\ \text{kanarie}(x) \rightarrow \text{heeft\_kleur}(x, \text{geel})$$

wat betekent dat iedere kanarie de kleur geel heeft. "Het belangrijkste verschil tussen deze (logische) talen en semantische netwerken is dat deze in feite in tekstuele vorm worden opgeslagen waardoor ze systematisch kunnen worden doorzocht, terwijl informatie in een semantisch netwerk alleen toegankelijk is door de verbinding te volgen." (Steels, p.148)
Om inderdaad gebruik te kunnen maken van de kennis zoals die is opgeslagen in een logische taal kan gebruik worden gemaakt van de inferentie-instrumenten zoals besproken in paragraaf 2.1. De kombinatie van kennis gerepresenteerd in een logische taal met een programma dat deze inferentie-instrumenten gebruikt wordt ook wel een kennissysteem genoemd. Bij deze systemen wordt verder nog een scheiding aangebracht tussen statische kennis die bijna altijd geld en dynamische kennis die alleen betrekking heeft op een specifiek geval waarover de gebruiker met behulp van de statische kennis meer te weten wil komen. De opbouw van een typisch kennissysteem is weergeven in *figuur 3.2*.



Figuur 3.2: Opbouw van een kennissyteem (aangepast van Bioch)

De in deze paragraaf behandelde kennis-representaties zijn gebaseerd op het klassieke kennisidee en hebben dan ook dezelfde kenmerken. De definitie van de kennis is strikt en precies. Opvallend is dat de kennis zelf als een passief object wordt gezien. Er is een heldere scheiding tussen de feiten en de regels die de kennis vormen in het kennis-bestand (knowledge base) en de bewerkingen die op die kennis middels inferentie worden uitgevoerd (inference engine).
Deze vorm van representatie wijkt af van de manier waarop onze hersenen kennis behandelen op het niveau van de neuronen zoals besproken in paragraaf 2.2, waar geen sprake is van de genoemde scheiding. Neuronen en hun verbindingen representeren niet

alleen kennis, zij bewerken deze ook. Dit betekent echter niet dat symbolische kennisrepresentatie geen betrekking heeft op menselijke kennis: “What the brain does may be thought of at some level as a kind of computation” (Charniak e.a., p 6). Dit is dan ook het argument waarmee de zogenaamde klassieke kunstmatige intelligentie, zoals in deze paragraaf voor een klein deel is beschreven, zichzelf rechtvaardigt in het toepassen van seriële bewerking van symbolen als simulatie van de menselijke intelligentie.

3.2 Connectionistische semantische netwerken

In paragraaf 2.2.2 is beschreven hoe de menselijke hersenen door middel van het parellel uitvoeren van bewerkingen uiteindelijk op vele gebieden effectiever kan werken dan een normale seriële computer. Om die reden heeft men geprobeerd deze parallelle structuur over te nemen in de computer. Helaas gebeurt dit voorlopig hoofdzakelijk door simulatie van een parallel systeem op een seriële computer waardoor bijvoorbeeld de snelheid momenteel nog te wensen over laat omdat zo’n systeem nog steeds één voor één alle bewerkingen uit moet voeren in plaats van allemaal tegelijkertijd. Parallelle bewerking heeft echter ook nog andere voordelen, zoals het om kunnen gaan met vage en onvolledige informatie, zodat zelfs gesimuleerde parallelle architecturen steeds vaker als oplossingen worden geïmplementeerd.
Het is mogelijk om een semantisch netwerk als connectionistisch netwerk, dus met parallelle bewerkingen, te realizeren. (Shastri, p.113ff) Een Connectionistisch Semantisch Netwerk (CSN) bestaat, net als een gewoon semantisch netwerk, uit een aantal knopen en verbindingen tussen de knopen. Aan iedere knoop ook is een woord en hiermee een concept verbonden. De verbindingen hebben bij de meeste CSN echter geen label zoals ‘is een’ of ‘heeft kleur’. Zij hebben daarentegen wel een gewicht welke vergelijkbaar is met de sterkte van de synaptische verbindingen zoals beschreven in paragraaf 2.2.2. De knopen combineren de outputs van andere knopen die verbonden zijn via deze verbindingen tot één inputactivatie. Middels een transferfunctie wordt dan een output voor deze knoop berekend. We definiëren:

o_i = output van knoop i
w_{ij} = gewicht van verbinding van knoop j naar knoop i
a_i = activatieniveau van knoop i

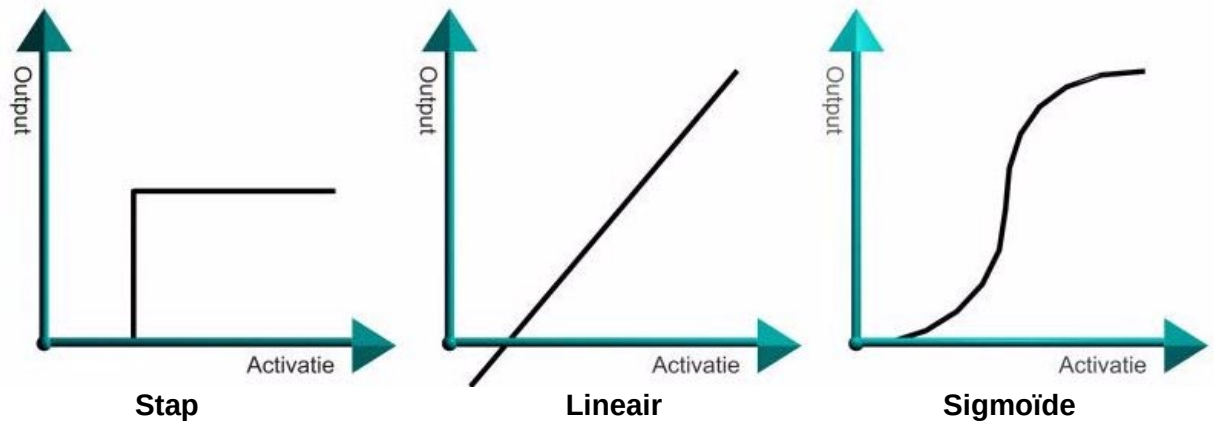
Meestal geldt dat:

$a_i = \sum_j w_{ij} o_j$ waarin o_j de output is van knoop j

Vervolgens wordt met behulp van deze activering de output van de knoop berekend middels de transferfunctie:

$o_i = f(a_i)$

Er zijn drie hoofdtypen transferfuncties die vaak worden gebruikt. Deze zijn weergegeven in *figuur 3.3*. De output van deze knoop dient dan eventueel weer als input voor andere knopen.



Figuur 3.3: Drie veel gebruikte transferfuncties (Bioch), (Crestani, p.463)

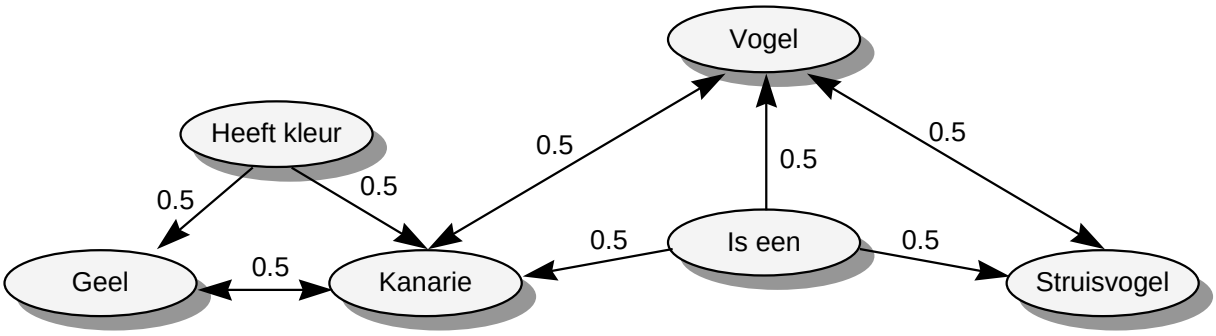
Indien nu enkele knopen bijvoorbeeld door de gebruiker worden geactiveerd kan dit een kettingreactie van activeringen tot gevolg hebben. Dit staat bekend onder de naam Spreading Activation (SA). Hierdoor kunnen aanverwante concepten worden gevonden.
Alhoewel bij een CSN meestal geen labels worden gegeven aan de verbindingen is het toch mogelijk de relaties tussen concepten weer te geven. In *figuur 3.4* is een voorbeeld hiervan gegeven. Hier zien we dat bepaalde knopen toegevoegd zijn zoals 'heeft kleur' en 'is een', die niet een specifiek concept voorstellen.

Stel voor dit voorbeeld verder dat:

$$a_i = \sum_j w_{ij} o_j$$

en de transferfuntie is een stapfunctie:

$$f(a) = \begin{cases} 0 & \text{voor } a < 1 \\ 1 & \text{voor } a \geq 1 \end{cases}$$



Figuur 3.4: Voorbeeld van een CSN met concepteigenschappen

In dit voorbeeld kunnen we zien dat indien 'kanarie' en 'is een' volledig worden geactiveerd ($o_{kanarie} = o_{is\_een} = 1$), dit zal leiden tot de activering van 'vogel':

$f(a_{vogel}) = f(\sum_j w_{ij}o_j) = f(\ 0{,}5.1 + 0{,}5.1) = f(1) = 1$

Zo zal ook de activering van ‘kanarie’ en ‘heeft kleur’ verder uitsluitend de activering van ‘geel’ tot gevolg hebben. Andersom werkt dit ook. Activering van ‘vogel’ en ‘is een’ leidt tot activering van alle knopen die een vogel representeren, dus ‘kanarie’ en ‘struisvogel’. Op deze manier kan kennis worden onttrokken aan het netwerk en gecompliceerde kennisstructuren kunnen op deze manier worden geïmplementeerd.
Een voordeel van een CSN is dat het nu mogelijk is om het systeem zelf te laten leren. Bij een gewoon semantische netwerk moesten alle concepten en verbindingen handmatig worden aangemaakt. Bij een CSN is het mogelijk de verbindingen te laten bepalen op basis van bijvoorbeeld een statistische analyse van de mate waarin woorden in een document bij elkaar voorkomen (Crestani, p.459). Een nadeel van deze methode is dat deze geen precieze kennis op kan bouwen. Het kan alleen de kennis ongeveer benaderen en is daarom niet zozeer geschikt voor ‘hogere’ cognitieve vaardigheden zoals redeneren en bekritiseren.

3.3 Conceptuele ruimte

We hebben in hoofdstuk 2 al gezien dat de manier waarop hersenen informatie representeren op is te vatten als het plaatsen van vectoren in een (hoogdimensionale) ruimte. Eerst zal dit idee verder worden onderzocht (3.3.1) waarna een tweetal volledig connectionistische methoden aan bod zullen komen die zo’n hoogdimensionale ruimte in een computer kunnen construeren, namelijk simpele recurrente netwerken (3.3.2) en self-organising maps (3.3.3).

3.3.1 Kennis als hoogdimensionale ruimte

We kunnen als volgt een onderscheid maken tussen taal en dan met name de individuele woorden, en concepten (Schäuble, p.255):

- Een **woord** (of term) is een reeks letters
- Een **concept** is de semantische betekenis of de correcte interpretatie van een woord.

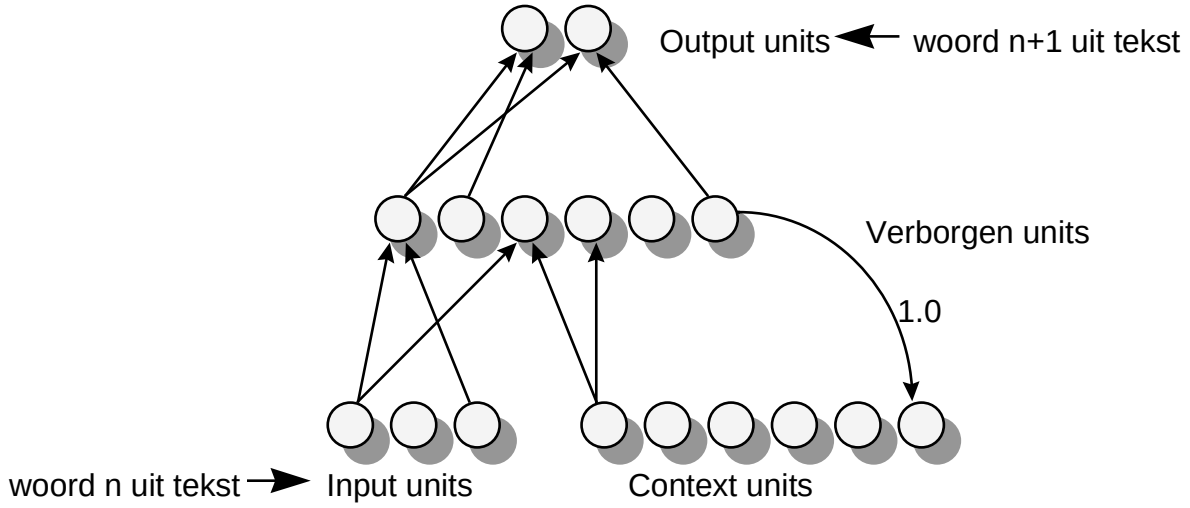
Met andere woorden: een term verwijst naar, of identificeert, een concept. In het vervolg zullen we ervan uitgaan dat er een één op één relatie bestaat tussen termen en concepten. Dit is uiteraard niet volledig correct aangezien dan o.a. het bestaan van homoniemen wordt verwaarloosd. Het betekent echter een aanzienlijke vereenvoudiging van het onderwerp . Interessant om te vermelden is dat kinderen in de eerste leerfasen ook een sterke voorkeur hebben voor deze één op één relatie (Imai e.a., p.170).
Het belangrijkste hier is echter dat er een duidelijk onderscheid is te maken tussen een term (op het syntactische of fysieke niveau) en het corresponderende concept (op conceptueel of semantisch niveau). Dit semantische niveau kunnen we beschouwen als een continuüm of ruimte. Indien we bijvoorbeeld twee termen definiëren met hun representaties op het semantisch niveau, zoals ‘rond’ en ‘plat’, dan is daarmee een tussenliggend spectrum van andere concepten gecreëerd waar geen termen aan verbonden zijn. We kunnen hierbij bijvoorbeeld denken aan een concept dat gekenmerkt zou kunnen worden door de term ‘ellipsvormig’.
In dit voorbeeld kunnen we ook zien dat concepten ‘rond’, ‘plat’ en ‘ellipsvormig’ weer tot een bepaalde deelverzameling behoren, namelijk die van vormen. We kunnen ook zeggen dat ze

een cluster van punten vormen in een **conceptuele ruimte**. Deze clusters zijn te onderscheiden van de overige concepten omdat hun onderlinge afstand kleiner is dan tussen deze concepten en ieder ander concept. “Thus, there must be considerable open space between the clusters” (Gelder, van, p.181). Dit geldt niet altijd, aangezien sommige clusters in elkaar overlopen omdat sommige concepten bepaalde eigenschappen gemeen hebben (Elman, p.140) en omdat de onderliggende natuurlijke taal vaag en dubbelzinnig kan zijn.
Het aantal dimensies dat deze ruimte moet hebben is afhankelijk van het kennisdomein dat men tracht te beschrijven en is niet van te voren vast te stellen. Alleen moet het duidelijk zijn dat een ruimte met hoge dimensionaliteit mogelijkheid geeft tot een betere rangschikking van de clusters waarbij meer clusters in de nabijheid van één ander cluster kunnen liggen zonder dat zij automatisch onderling een kleine afstand moeten hebben. Een hogere dimensionaliteit geeft dus de mogelijkheid tot het vormen van complexere structuren in deze conceptuele ruimte.
Het is ook mogelijk om clusters weer te combineren tot clusters en op deze manier een hiërarchie van concepten te maken. Tevens is er geen ondergrens noodzakelijk aan de verfijning die binnen een cluster weer aan te maken is.

3.3.2 Simpele Recurrente Netwerken

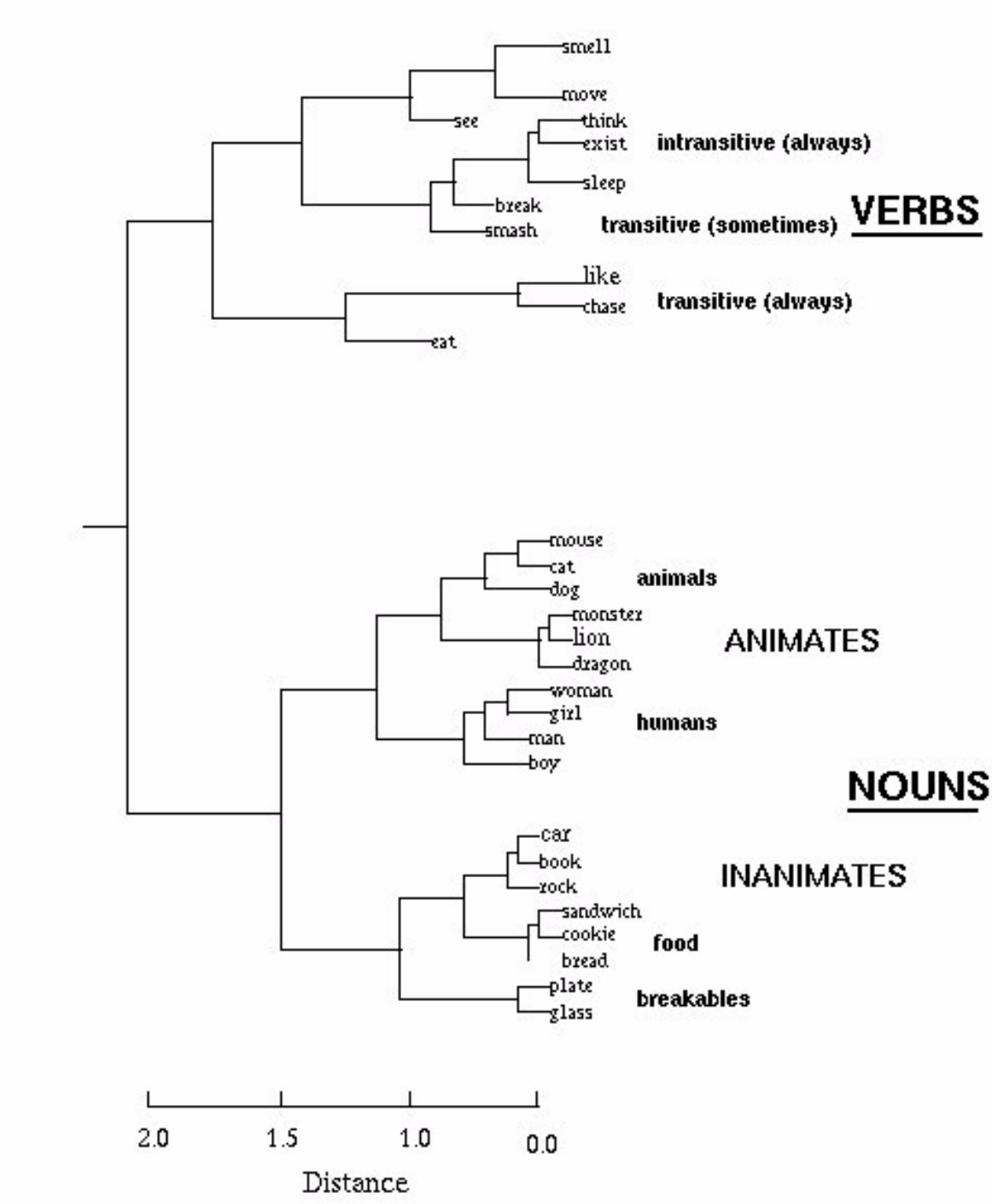
Kennis groeit in het algemeen niet voor niets. In paragraaf 2.2.1 hebben we gezien dat het ontwikkelen van kennis een overlevingsstrategie is die in het verleden zeer succesvol is gebleken. Kennis op zichzelf doet echter niets. Het moet wel noodzakelijk zijn bij een bepaalde handeling in de realiteit. In deze paragraaf zal een voorbeeld behandeld worden waarbij kennis als noodzaak opgebouwd wordt om een bepaalde handeling beter uit te voeren.
De handeling waar het om gaat is eenvoudig: een aantal achtereenvolgende woorden uit een zin worden aan een systeem gegeven, waarna deze het volgende woord moet voorspellen. Uiteraard is het onmogelijk altijd het goede antwoord te geven aangezien meestal meer dan één woord in aanmerking komt als volgend woord. Het is echter wel mogelijk om te zeggen of dit bijvoorbeeld een werkwoord of een zelfstandig naamwoord moet zijn en of dit in enkelvoud of meervoud moet staan. Het systeem dat hierna zal worden beschreven is hier inderdaad toe in staat.
Voor deze taak is gebruik gemaakt van een kunstmatig soort neurale netwerken dat bekend staat onder de naam Simpele Recurrente Netwerken (SRN). De werking van de neuronen of units uit dit netwerk is hetzelfde als die van de knopen uit paragraaf 3.2 met betrekking tot het combineren van de inputs met de bijbehorende gewichten tot een output volgens een transferfunctie. In dit geval is de transferfunctie echter een continue functie, bij voorkeur de sigmoïde. Hiernaast is ook de architectuur van het netwerk anders. In *figuur 3.5* is deze structuur weergegeven. Iedere input-unit stelt een woord voor en indien dit het volgende woord is dat aan het systeem wordt gevoerd dan is uitsluitend deze input-unit actief. Deze inputlaag is volledig verbonden met de verborgen laag, net als de context units. De activering van de verborgen laag wordt niet alleen gebruikt als input voor de outputlaag, maar zal bij de volgende trainings-stap ook unit voor unit worden overgenomen door de context units. Hierdoor heeft het netwerk een soort korte termijn geheugen, waardoor het zich een vorige interne staat kan herinneren. De output-units zijn stuk voor stuk weer geassocieerd met de woorden uit de vocabulaire. Voor ieder woord uit de tekst dat als input aan het netwerk wordt gegeven zal het netwerk als output het volgende woord uit de tekst moeten produceren.

De gewichten in dit netwerk worden getraind met behulp van het Error Back-Propagation algoritme, dat gegeven het juiste resultaat en het resultaat van het netwerk de fout berekend en deze vervolgens terugpropageert door het netwerk. (Voor gedetailleerde informatie over de werking van het Error Back-Propagation algoritme, zie Hertz e.a., p.115ff.)



Figuur 3.5: SRN voor woordvoorspelling. Niet alle verbindingen en units zijn getekend voor overzichtelijkheid. (Elman, p.143)

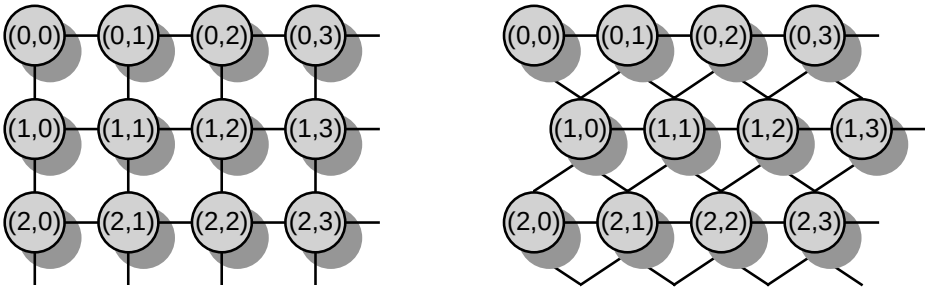
Het netwerk is getest met een beperkte vocabulaire van 29 woorden en een eenvoudige grammatica, waarmee 10.000 trainingszinnen werden gemaakt. Deze werden aan het systeem gevoerd. Om beter te kunnen presteren moest het netwerk kennis opbouwen over de betekenis van de verschillende woorden en welke eigenschappen bepaalde woorden hebben. Later is deze kennis zichtbaar gemaakt door een woord in de inputlaag te activeren en vervolgens de output van de verborgen laag als vector te zien. Het aantal elementen van deze vector (70) maakte het echter onmogelijk deze ruimte volledig zichtbaar te maken. Door het toepassen van hiërarchische cluster analyse kon echter het beeld van *figuur 3.6* worden opgesteld.
In dit figuur zien we dat het SRN een duidelijke clustering in z'n interne representatie van de concepten heeft gecreëerd. Zonder dat dit aan het systeem uitgelegd was heeft dit systeem als eerste een scheiding gemaakt tussen werkwoorden en zelfstandig naamwoorden. Binnen deze clusters vinden we echter verdere clusters zoals levende en niet-levende dingen met een verdere opsplitsing in dieren en mensen.
Deze kennis zat dus al in de zinnen zelf en kon worden afgeleid in de wijze waarop een woord is gebruikt; in welke context het voor kan komen.
Opvallend is dat in dit voorbeeld de clustering zowel plaats vindt op basis van syntactische kenmerken, zoals werkwoord of zelfstandig naamwoord, als op basis van semantische kenmerken, zoals tussen mensen en dieren of tussen breekbare dingen en eetbare dingen.
(Gelder, van, p.182)



Figuur 3.6: Hiërarchische clustering in de representatie van woorden in de verborgen laag van het SRN. (Elman, p.147)

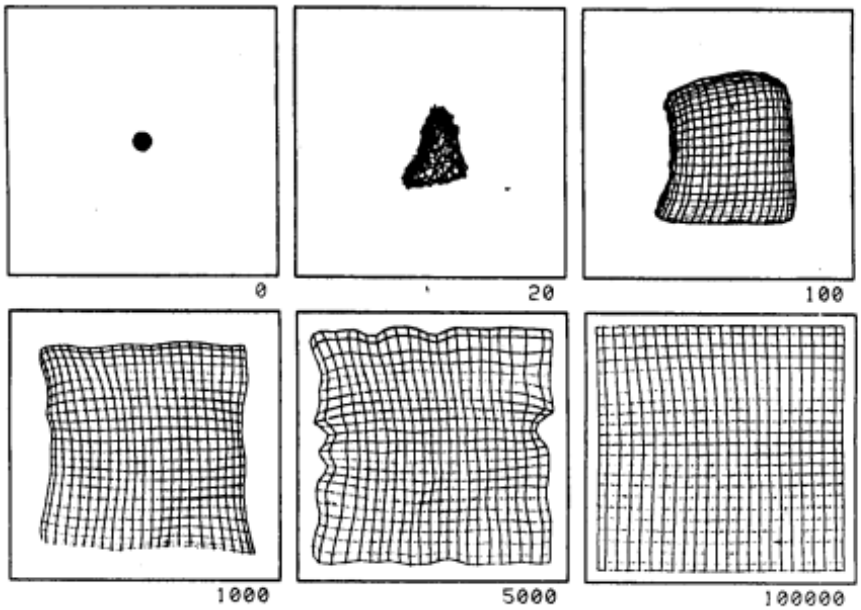
3.3.3 Self-Organizing maps

Werking van het Self-Organizing Map algoritme
Het 'Self-Organizing Map'-algoritme (SOM-algoritme) werkt met een map of **raster van eenheden**. Dit raster kan iedere dimensionaliteit aannemen, maar in het vervolg zullen we uitgaan van raster van n eenheden (één-dimensionaal) of van n bij m eenheden (tweedimensionaal). De eenheden in zo'n tweedimensionaal raster kunnen op verschillende manieren worden gerangschikt zoals weergegeven in *figuur 3.7*.



Figuur 3.7: Rangschikking van eenheden in een tweedimensionaal raster: Rechthoekig (links) of hexagonaal (rechts)

Aan iedere eenheid in zo'n raster is een referentievector verbonden. Deze referentievector is meestal van een hogere dimensionaliteit dan het raster, oftewel het aantal elementen van de vector is groter dan één of twee. Bij het volgende voorbeeld zal deze echter twee dimensies hebben, net als het raster. In dit voorbeeld dat is weergegeven in *figuur 3.8* is een raster met een rechthoekige structuur toegepast. In *figuur 3.8*. zijn de referentievectoren van dit raster weergegeven, waarbij een lijn is getrokken tussen de referentievectoren van de eenheden die in het raster naast elkaar liggen.



Figuur 3.8: Referentievectoren van een SOM na respectievelijk 0,20,100,1.000, 5.000 en100.000 trainingstappen. (Kohonen, p.82)

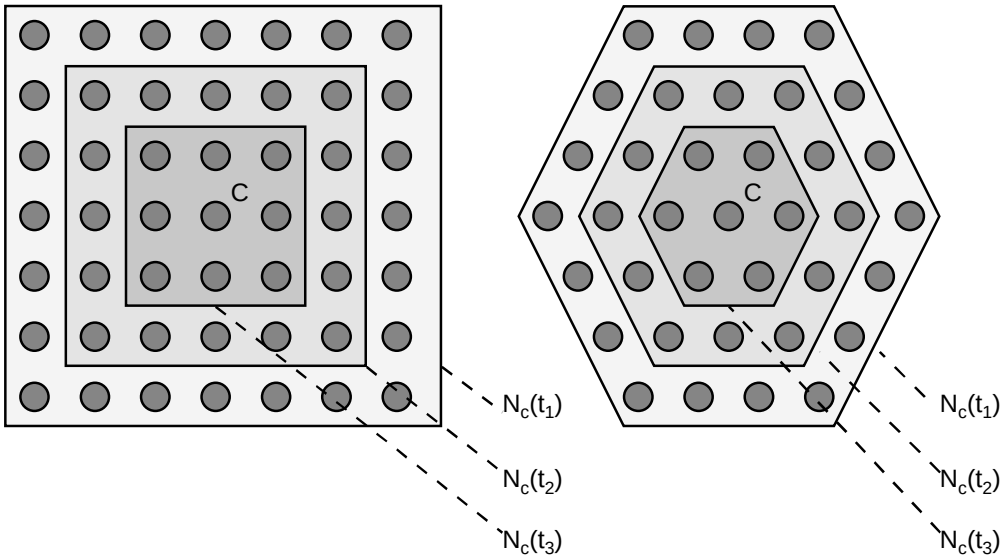
Tevens is in dit figuur telkens een vierkant weergegeven. Op dit vierkant is een uniforme verdeling gedefinieerd, waaruit telkens een trekking wordt gedaan wat resulteert in een tweedimensionale vector. Door deze vectoren aan het SOM-algoritme te voeren zien we dat de referentievectoren steeds beter verspreid worden over de totale verdeling. We kunnen de SOM zien als een elastisch net dat de verdeling probeert te spannen. Het SOM-algoritme probeert de afstand tussen de referentievectoren en de getrokken vectoren zo klein mogelijk te maken. Gebieden waar zich een hogere concentratie trainingsvectoren voordoet en waar dus een hogere kansdichtheid is zullen leiden tot een hogere concentratie van referentievectoren in deze gebieden. Dit wordt als volgt bereikt:
De referentievectoren krijgen in het begin hun startwaarden die bijvoorbeeld willekeurig worden bepaald. Om het netwerk alvast een stuk in de goede richting te brengen kan ook eerst de autocorrelatie matrix van de input-vectoren worden bepaald (Kohonen, p.107). De twee eigenvectoren die behoren tot de twee grootste eigenwaarden van deze matrix omspannen dan een tweedimensionale lineaire subruimte waarin het raster van referentievectoren kan worden gedefinieerd.
Na de initialisatie worden alle getrokken vectoren stuk voor stuk behandeld:
Als eerste wordt voor de getrokken vector x bepaald welke referentie vector m_i van de SOM deze het dichts benaderd. Hiervoor wordt meestal de Euclidische afstand gebruikt. De winnende eenheid wordt aangeduid met c en de bijbehorende referentievector met m_c. We kunnen dus stellen dat:

$$|| x - m_c || = \min_i \{ || x - m_i || \}$$ (|| . || geeft de Euclidiaansche afstand)

Vervolgens worden de referentievectoren van de winnende eenheid c en alle eenheden die in het raster nabij eenheid c liggen aangepast zodat de afstand tussen deze vectoren en x kleiner wordt:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)]$$

Hierin is $h_{ci}(t)$ de topologische buurtfunctie die afneemt naarmate eenheid c en eenheid i verder van elkaar verwijderd zijn in het raster. Meestal geldt dat $h_{ci}(t) = \alpha(t)$ als $i \in N_c$ en $h_{ci}(t) = 0$ als $i \notin N_c$ waarbij N_c de buurtset is van c. In *figuur 3.9* is weergegeven welke eenheden meestal als buren van c worden beschouwd en dus in N_c zitten.



Figuur 3.9: Twee voorbeelden van een buurtfunctie ($t_1 < t_2 < t_3$). (Kohonen, p.79)

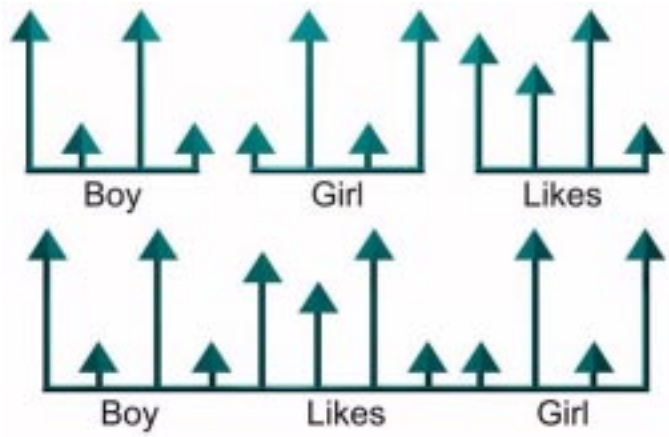
$\alpha(t)$ is op te vatten als de leersnelheid. In het begin zal deze over het algemeen hoog zijn, terwijl later voor de fijnere aanpassingen deze lager zal zijn. Ook de buurtset van de winnende eenheid zal eerst groot zijn, soms de halve diameter van het raster, en later afnemen totdat de buurt alleen nul is, oftewel uitsluitend de winnende eenheid c wordt aangepast. Een andere veel gebruikte mogelijkheid voor de buurtfunctie is de Gaussiaanse functie:

$$h_{ci} = \alpha(t).\exp\left(-\frac{|| m_c - m_i ||^2}{2\sigma^2(t)} \right)$$

waarin $\alpha(t)$ weer de leersnelheid en $\sigma(t)$ de breedte van de buurt vertegenwoordigen.
Het SOM-algoritme kan dus een raster van referentievectoren spannen in een inputruimte waarbij dit raster de verdeling van de punten in deze ruimte probeert te benaderen (Kohonen, p.80). Een belangrijke eigenschap van dit raster is dat het topologie-bewarend is: Inputvectoren die dicht bij elkaar in de inputruimte liggen corresponderen met referentievectoren die behoren tot eenheden die in het raster dicht bij elkaar liggen.

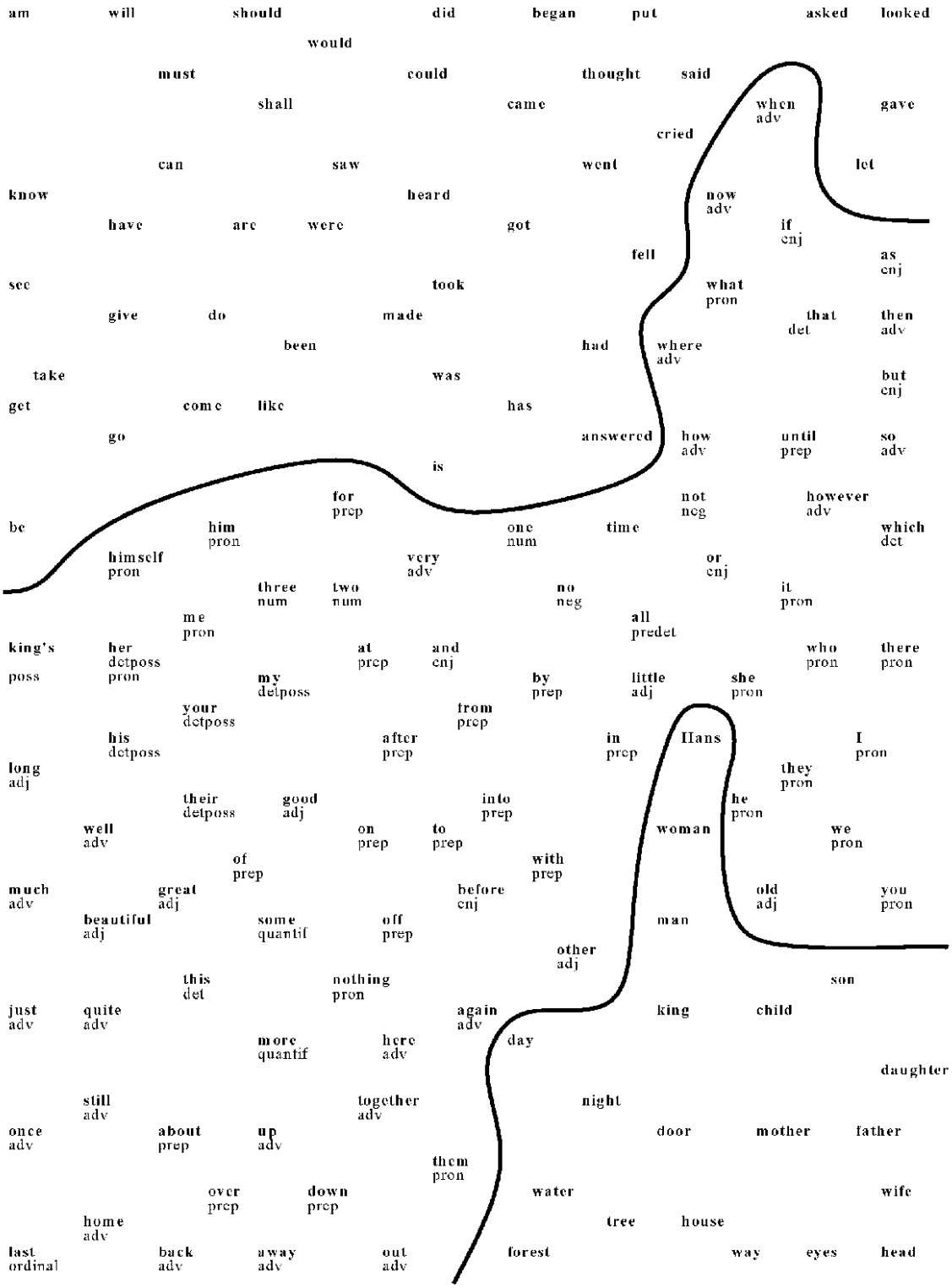
Conceptuele ruimte met behulp van Self-Organizing Maps
Het is mogelijk om een conceptuele ruimte te creëren met behulp van een SOM. Aan de aan de hand van een voorbeeld wordt dit proces uitgelegd. De input-data die in dit voorbeeld werd gebruikt betreft een verzameling van 200 Engelstalige sprookjes van de gebroeders Grimm (Honkela e.a. 1995) in totaal bevatte deze bijna 250.000 woorden met een vocabulaire van 7000 woorden.
Als voorbewerking werden alle verhalen achter elkaar gezet in één bestand. Leestekens werden verwijderd en hoofdletters werden omgezet in kleine letters. De woorden ‘a’, ‘an’ en ‘the’ werden verwijderd. Aan ieder woord werd een unieke 90-dimensionale vector toegekend. Deze vectoren waren statistisch onafhankelijk zodat er geen correlatie tussen bestond.
De inputvectoren voor de SOM werden opgebouwd uit de vectoren van drie achtereenvolgende woorden (een woordtriplet) in de tekst, waarbij het middelste woord het

sleutelwoord is. Deze drie 90-dimensionale vectoren werden simpelweg achter elkaar gezet, zodat de inputvectoren voor het SOM in totaal 270 dimensies kende. Dit is vereenvoudigd weergegeven in *figuur 3.10*.



Figuur 3.10: Voorbeeld van een inputvector voor het SOM-algoritme. Eerst wordt aan ieder woord een n-dimensionale vector toegekend. (In dit voorbeeld geldt n=4) Vervolgens wordt voor iedere woord-triplet in de tekst een 3n-dimensionale vector opgebouwd uit deze vectoren.

Voor de initiële vectoren van de SOM werd de lineaire initialisatie met behulp van eigenvectoren gebruikt. De SOM bestond uit een hexagonaal raster van 42 bij 36 eenheden.
In de tekst werden alle woordtriplets opgezocht waarvan het sleutelwoord behoorde tot de 150 meest voorkomende woorden. De bijbehorende inputvectoren werden aan de SOM gevoerd. De tekst werd op deze manier 1 miljoen keer doorlopen. Het doel van dit proces was het ordenen van de woorden zodat woorden die in ongeveer dezelfde context (de twee omliggende woorden) voorkomen, dicht bij elkaar in het raster te plaatsen.
In *figuur 3.11* is weergegeven wat het resultaat was. Voor ieder van de 150 sleutelwoorden is bepaald welke eenheid in alle voorkomende contexten het meest als winnende eenheid werd gevonden. Aan deze eenheid is dit sleutelwoord als label toegekend.
We zien in *figuur 3.11* dat de resulterende SOM inderdaad als een tweedimensionale conceptuele ruimte is op te vatten. In het figuur zijn de belangrijkste gevonden clusters na afloop aangegeven; boven vinden we de werkwoorden, rechtsonder de zelfstandig naamwoorden en in het midden de overige linguïstische categorieën. Binnen deze syntactische indeling kunnen we weer semantische clusters vinden.



Figuur 3.11: De 150 meest voorkomende woorden in de Grimm-sprookjes. De woorden zijn weergegeven op de plaats die zij innemen op het raster. De eenheden van het raster zijn zelf niet weergegeven. Bij de woorden uit linguïstische categorieën anders dan werkwoorden en zelfstandig naamwoorden is de categorie aangegeven (Honkela e.a. 1995, p.5).

3.4 Conclusies

In dit hoofdstuk zijn enkele methoden van kennisrepresentatie in computers behandeld, van symbolische kennissystemen tot conceptuele ruimte. Uiteraard is deze opsomming verre van volledig. Door echter enkele referentiepunten in het conceptuele vlak van kennisrepresentatie te plaatsen is het de bedoeling dat een beter beeld van dit gebied is ontstaan.
Symbolische kennissystemen vragen om een intensieve inzet van menselijke experts op het domein en de implementatie ervan wordt bemoeilijkt door het feit mensen hun kennis moeilijk expliciet kunnen formuleren. Symbolische systemen hebben daarnaast wel als voordeel dat zij 'strikt' zijn en daardoor geschikt zijn voor hogere cognitieve activiteiten. "In fact , a common criticism levelled against connectionism is that although it may be appropriate for modelling 'low level' cognitive activities and 'approximate" effects such as semantic priming and associative recall, it is unsuitable for dealing with 'high level' problems related to representation and reasoning" (Shastri, p.113). Connectionistische systemen hebben echter als voordeel dat zij veelal automatisch kennis uit data kunnen vergaren en daardoor makkelijker inzetbaar zijn.
Voor een informatie-zoeksysteem is het verder niet noodzakelijk dat de kennis die het heeft geschikt is voor redeneren. Dit systeem moet informatie hebben over de relaties tussen de termen deze informatie mag best slechts **bij benadering** juist zijn.
Geschikte kandidaten voor kennisrepresentatie die automatisch kan worden opgebouwd en welke geschikt zijn voor een informatie-zoeksysteem zijn dus Connectionistische Semantische Netwerken en conceptuele ruimte. Voor kennisrepresentatie in conceptuele ruimte heeft het Self-Organizing Map-algoritme verder de voorkeur boven het Simpele Recurrente Netwerk aangezien deze laatste (nu nog) slechts toe te passen is op zeer kleine vocabulaires en eenvoudige grammatica.
Interessant om te vermelden is dat er pogingen worden gedaan om symbolische en connectionistische kennissystemen te integreren om op deze manier van de voordelen van beide methoden gebruik te maken (Fu). Hierbij worden onder andere systemen ontworpen waar kennis van een neuraal netwerk kan worden overgezet naar een symbolisch kennissysteem en omgekeerd. Tevens wordt gewerkt aan systemen die beide kennisvormen tegelijk aanhouden.

Hoofdstuk 4: Informatie-zoeksystemen

"The purpose of an Information Retrieval system (IR system) is to ***retrieve information*** *items according to a* ***query*** *expressing the information need of a user...First of all, the system has to determine the semantic meaning of the query, i.e. the* ***concept or concepts*** *addressed by the* query. *Secondly and likewise, the* ***concept or concepts*** *addressed by the stored* ***information items*** *have to be known by the IR system."* (Schäuble, p.254).

De informatie-zoeksystemen die momenteel beschikbaar zijn kennen een grote diversiteit. In dit hoofdstuk zullen de belangrijkste klassen besproken worden. Als eerste zal beschreven worden aan welke voorwaarden een informatie-zoeksysteem moet voldoen (4.1), waarna de Boolse zoeksystemen (4.2) worden beschreven. Vervolgens komen enkele methoden aan bod waarmee een informatie-zoeksysteem meer inzicht in de semantische betekenis van zoekopdracht en documenten kan krijgen (4.3) en enkele methoden die specifiek gebruik maken van neurale netwerken (4.4). Als afsluiting volgen enkele conclusies (4.5).

4.1 Prestaties van informatie-zoeksystemen

Het scala aan informatie-zoeksystemen is groot en deze grote diversiteit maakt het moeilijk om de verschillende systemen met elkaar te vergelijken. Gezien de functie van een informatie-zoeksysteem, namelijk het vinden van de relevante documenten voor de gebruiker in een collectie, kunnen we de prestaties van een gegeven systeem op de volgende aspecten beoordelen
(vertaald van Rijsbergen, H7):

1. Het **bereik van de collectie**, oftewel de mate waarin het systeem verwijzingen naar relevante materie bevat
2. De **vertraging**, oftewel de gemiddelde interval tussen het moment dat de zoekopdracht is gegeven en het moment dat het antwoord wordt gegeven.
3. De **vorm van de presentatie** van de output.
4. De **moeite** die de gebruiker moet doen om de antwoorden te krijgen van het systeem.
5. De **vindkracht** ('recall') van het systeem; oftewel dat deel van de relevante informatie dat daadwerkelijk door het systeem wordt gevonden als resultaat op een zoekopdracht.
6. De **precisie** van het systeem, oftewel dat gedeelte van het gevonden materiaal dat daadwerkelijk relevant is voor de gebruiker.

Aspect 1 tot en met 4 zijn gemakkelijk te waarderen. Het zijn vooral de vindkracht en precisie die bepalen hoe effectief een systeem is (Rijsbergen, H7). Met andere woorden: ze vormen een maatstaf voor de vaardigheid van het systeem in het vinden van relevante documenten waarbij niet-relevante documenten voor de gebruiker buiten zicht blijven.
Teneinde de begrippen vindkracht en precisie nader te analyseren verdelen we de documentencollectie als weergegeven in *tabel 4.1*.

	Relevant	Niet-relevant	
Gevonden	$A \cap B$	$\overline{A} \cap B$	B
Niet gevonden	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$	$\overline{B}$
	A	$\overline{A}$	N

N=Totaal aantal documenten in het systeem

A=De verzameling relevante documenten

B=De verzameling documenten die door het systeem zijn gevonden

Tabel 4.1: Verdeling van de collectie

We kunnen nu de begrippen vindkracht en precisie als volgt definiëren:

$$\text{Precisie} = \frac{|A \cap B|}{|B|} \qquad (0 \le \text{Precisie} \le 1)$$

$$\text{Vindkracht} = \frac{|A \cap B|}{|A|} \qquad (0 \le \text{Vindkracht} \le 1)$$

($|\,.\,|$ geeft het aantal elementen [documenten] in de deelverzameling, dus:
$|N| = |A| + |\overline{A}| = |B| + |\overline{B}|$)

Indien een informatie-zoeksysteem een perfecte prestatie zou leveren dan zouden alle documenten die door dit systeem zijn gevonden ook daadwerkelijk relevant zijn voor de gebruiker. Hiernaast zouden er ook geen andere relevante documenten onopgemerkt blijven door het systeem. Deze hypothetische situatie is weergegeven in *tabel 4.2*.

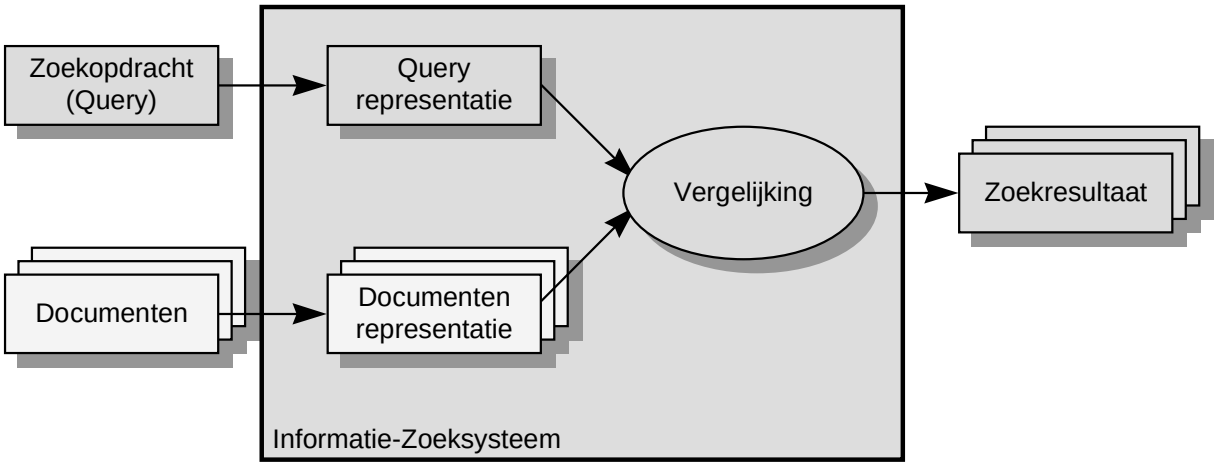
	Relevant	Niet-relevant	
Gevonden	A = B	∅	B
Niet gevonden	∅	$\overline{A} = \overline{B}$	$\overline{B}$
	A	$\overline{A}$	N

Tabel 4.2: Verdeling van de collectie bij een perfecte zoekmachine

In dit ideale geval zouden zowel vindkracht als precisie gelijk zijn aan 1. Hoge precisie en vindkracht kenmerken dus een effectiever informatie-zoeksysteem.
Met deze maatstaven is dus de effectiviteit van een informatie-zoeksysteem te waarderen. Het aantal documenten is objectief vast te stellen, net als het aantal gevonden documenten. Het bepalen van de relevantie vormt echter een groot probleem en is zeer persoonsgebonden en dus subjectief.
De effectiviteit is echter niet het enige aspect waar we een informatie-zoeksysteem op af moeten rekenen. Ook de kosten die gemoeid gaan met het opzetten en onderhouden van het systeem alsmede de kosten van de benodigde opslagcapaciteit en processortijd zijn relevant in de waardering van een systeem. Een systeem waar alle gebruikers blij mee zijn maar meer kost dan een reis naar de maan is geen goed systeem.

4.2 Boolse zoeksystemen

De eerste informatiesystemen werden op dezelfde manier opgebouwd als bibliografieën: Per document werd de titel, auteur en omschrijving van de vorm opgeslagen, maar ook trefwoorden en samenvattingen. Om vervolgens de documenten te vinden werd gebruik gemaakt van de zogenaamde Boolean Retrieval (Ferber): de gebruiker kan aangeven welke woorden of 'zoektermen' in de documenten voor moeten komen en welke beslist niet voor mogen komen. Een verfijning van dit systeem kan zijn dat men aan kan geven waar de zoekterm voor moet komen, zoals in de titel of in de auteursnaam. Verder zou de gebruiker zijn zoekacties uit kunnen breiden door de logische operatoren 'AND' en 'OR' te gebruiken op de zoektermen.



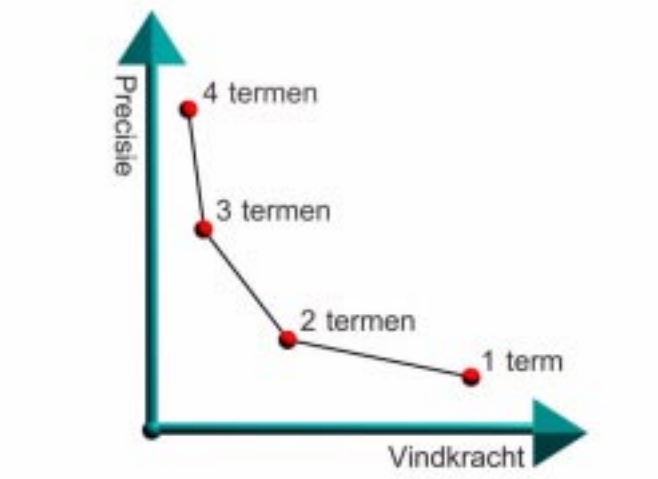
Figuur 4.1: Een eenvoudig informatie-zoeksysteem (aangepast van Crestani, p.456)

In *figuur 4.1* is een informatie-zoeksysteem weergegeven zoals in de vorige alinea is besproken. Het systeem vergelijkt de (interne representatie van de) zoekopdracht met de representatie die van de documenten is opgeslagen, zoals titel, auteursnaam en samenvatting. Indien er genoeg overeenkomsten worden gevonden zal het desbetreffende document worden opgenomen in het zoekresultaat.
Het vinden van de overeenkomende zoektermen kan gebeuren via de 'n-gram' methode: de complete opgeslagen tekst wordt sequentieel doorlopen waarbij iedere keer n achtereenvolgende tekens worden vergeleken met de zoektermen. Voor grotere documentencollecties zou dit echter te veel tijd vergen. Voor iedere zoekactie zou het systeem dan de complete tekst teken voor teken door moeten lopen!
Als oplossing hiervoor zijn geïnverteerde indexen[1] (inverted indices) ontwikkeld. Bij deze methode wordt een zeer grote index bijgehouden van bijna alle woorden in de representaties van de documenten, waarbij bij elk van deze indextermen wordt vermeldt in welke van deze representaties zij voorkomen. Woorden met weinig informatiewaarde, zoals 'de' en 'het' kunnen hierbij buiten beschouwing worden gelaten. Het informatie-zoeksysteem hoeft nu alleen de zoektermen in deze gestructureerde index op te zoeken om te weten te komen in welke documenten ze voorkomen.

[1] De internet-zoekmachine 'Alta-Vista' is een uitstekend voorbeeld van een informatie-zoeksysteem dat gebruik maakt van een geïnverteerde index

Een geïnverteerde index zorgt ervoor dat het zoekproces aanzienlijk wordt versneld. Een nadeel is echter dat een geïnverteerde index veel ruimte in neemt, soms net zoveel als de totale tekst zelf.

Op de eerste vier aspecten die in paragraaf 4.1 zijn genoemd presteert het boolse zoeksysteem goed: het bereik kan, afhankelijk van de implementatie, alle tekstuele documenten omvatten. De methode is snel en een enkele zoekactie vergt weinig moeite van de gebruiker. De vorm van de output kan ook volledig duidelijk zijn voor de gebruiker. Deze kan ook inzien waarom een document wel of niet is opgenomen in het resultaat.
De effectiviteit oftewel de verhouding tussen de vindkracht en de precisie laat echter veel te wensen over:
Bij het Boolse zoeksysteem is de effectiviteit, dus de precisie en vindkracht, afhankelijk van de door de gebruiker toegepaste zoekstrategie. De gebruiker kan namelijk zelf bepalen hoeveel zoektermen zij/hij gebruikt. Bij het gebruik van slechts één zoekterm zal het systeem een groot aantal documenten vinden die door die term worden gekenmerkt. Voor de gebruiker zullen hier zeker relevante documenten tussen zitten, de vindkracht is in dit geval groot. Echter, door het gebruik van slechts één zoekterm zal een groot deel van de gevonden documenten niet voor de gebruiker relevant zijn; de precisie is laag.
Een voorbeeld: Iemand is op zoek naar informatie over klassieke westerse filosofie. Het resultaat van de zoekactie met de term ‘philosophy’ geeft bij de zoekmachine Alta-Vista 2.030.690 gevonden documenten. Dit resultaat bevat echter ook documenten die over niet-westerse filosofie gaan en zelfs over documenten die toevallig een keer het woord filosofie gebruiken. Om de zoekactie specifieker te maken wordt de query uitgebreid tot ‘philosophy Greece’. Dit heeft als gevolg dat het resultaat nog maar 889.440 documenten bevat. Echter, door deze beperking worden alle documenten gemist die niet expliciet het woord ‘greece’ bevatten maar wel klassieke griekse filosofie behandelen. Door het gebruik van meerdere termen is de precisie misschien wel toegenomen, maar de vindkracht is gedaald.
We vinden bij deze systemen dus een wisselwerking tussen precisie en vindkracht. In *figuur 4.2* is dit weergegeven.



Figuur 4.2: Wisselwerking tussen vindkracht en precisie

We kunnen zien dat boolse informatie-zoeksystemen vindkracht moeten opofferen om een hogere precisie te bereiken. Een zoekopdracht zal dus zelden zowel een hoge precisie als een hoge vindkracht tot gevolg hebben.

De tekortkomingen in de Boolse zoeksystemen worden nog duidelijker indien er bij de gebruiker sprake is van een vage informatiebehoefte, oftewel wanneer de gebruiker zelf nog niet precies weet wat hij zoekt.
Het systeem is echter wel duidelijk, eenvoudig en doorzichtig voor de gebruiker en ondanks de voorheen genoemde tekortkomingen is dit systeem vandaag de dag verreweg het meest toegepaste.

4.3 Representatie van de inhoud

In de vorige paragraaf hebben we gezien dat een Bools zoeksysteem niet erg effectief is. Dit is het gevolg van het feit dat een dergelijk systeem de tekst beschouwt als een grote zak met woorden, met hier en daar misschien nog een apart vakje voor de titel, auteursnaam en trefwoorden. Het systeem heeft geen 'benul' van de betekenis van deze woorden en nog minder van de zinnen die hiermee worden gevormd.
Om dit op te lossen zal het systeem uitgerust moeten worden met kennis. Als in het voorbeeld van de vorige paragraaf het systeem wist dat Griekse filosofie betrekking had op namen zoals Socrates, Plato en Aristoteles dan zouden documenten die deze namen bevatten en dus over Griekse filosofie gaan door het systeem worden gevonden, ook al zouden deze documenten niet specifiek de woorden 'philosophy' of 'Greece' bevatten.
In hoofdstuk 2 zijn enkele theorieën over kennis behandeld. Volgens de klassieke theorie bestaat kennis uit een hiërarchie van concepten. Bij de woorden 'philosophy' en 'Greece' zou in deze hiërarchie Griekse filosofie als onderdeel van het bovenliggend concept 'filosofie' worden gevonden. In deze structuur zullen dan ook andere concepten die verbonden zijn aan Griekse filosofie zoals de genoemde namen, geassocieerd worden.
In het licht van de moderne kennisideeën die in hoofdstuk 2 aan bod zijn gekomen kunnen we deze kennis opvatten als een netwerk van 'neuronen' welke in bepaalde combinaties zijn op te vatten als concepten. Het samenvallen van de woorden 'philosophy' en 'Greece' zullen in een dergelijk netwerk een bovenliggende structuur activeren, die op haar beurt weer andere concepten kan activeren.
We kunnen dan ook zeggen dat het informatie-zoeksysteem de **semantiek van de zoekopdracht** moet kunnen achterhalen. Dit kan worden gezien als het relateren van de query aan concepten. Dit relateren kan de vorm aannemen van het identificeren van de geassocieerde concepten in de hiërarchie van concepten of het activeren van de concepten in een neuraal netwerk. We zouden kunnen stellen dat indien een systeem de query op één van deze manieren in relatie kan brengen met een interne kennisrepresentatie dit systeem de query 'begrijpt.'
Vervolgens zal deze semantische betekenis van de query in relatie moeten worden gebracht met de (representaties van de) documenten. We kunnen dus stellen dat het systeem ook moet kunnen oordelen over de **semantiek van de documenten**.
In de afgelopen veertig jaar zijn al enkele stappen in deze richting genomen.

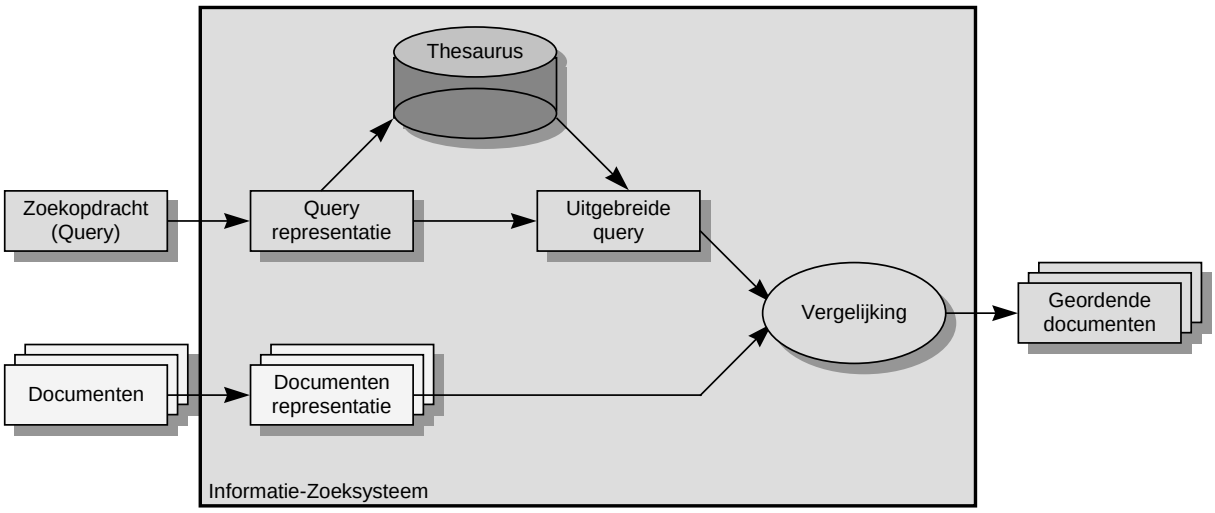
4.3.1 Woordstamreductie

Om een woord niet alleen als een reeks tekens te zien, maar ook om meer over de betekenis te weten te komen, kan men beginnen door niet naar de specifieke vorm te kijken waarin het woord staat. Woorden zoals 'philosophy' en 'philosophies' zullen normaal als twee verschillende woorden worden gezien, maar indien beide woorden tot hun stam worden herleid (in dit voorbeeld het woord 'philosophy') dan is voor het systeem duidelijk dat deze

woorden op hetzelfde concept betrekking hebben. Dit geldt uiteraard nog sterker voor werkwoorden. De zelfstandige naamwoorden ‘philosophy’ en ‘philosophies’ hebben uiteindelijk betrekking op twee verschillende dingen: één filosofie of een verzameling van filosofieën. Bij de voorbeelden ‘Ik adem’ en ‘hij ademt’ wordt echter exact dezelfde handeling, namelijk ademen, met twee verschillende woorden aangeduid, alleen omdat de persoonsvorm anders is.
De toepasbaarheid van deze methode verschilt echter per taal. Engels leent zich uitstekend voor deze methode, terwijl bijvoorbeeld de Duitse morfologie te veel uitzonderingen bevat (Ferber).

4.3.2 Thesauri

Een andere manier om meer te weten te komen over de betekenis van een woord is het gebruik van een thesaurus. Dit is een gestructureerde vocabulaire waar woorden zoals synoniemen en overkoepelende en onderliggende woorden aan elkaar worden gerelateerd. Vervolgens kunnen deze woorden worden toegekend aan een document als karakterisering van dat document, het zogenaamde indexeren. Bij een zoekactie wordt dan niet alleen gekeken of de zoektermen van de gebruiker overeenkomen met deze indextermen maar worden ook aan deze termen verwante woorden gezocht.



Figuur 4.3: Een informatie-zoeksysteem met thesaurus (aangepast van Crestani, p.457)

Zoals in *figuur 4.3* is getoond wordt de zoekopdracht of query van de gebruiker uitgebreid met behulp van aan de zoektermen gerelateerde woorden. Deze uitgebreide query wordt dan vergeleken met de interne representaties van de documenten.
Een thesaurus wordt over het algemeen handmatig aangelegd en indextermen worden dan ook handmatig toegekend aan de documenten. De thesaurus die dan ontstaat heeft in het algemeen de structuur van een semantisch netwerk zoals in paragraaf 3.1 is beschreven. Om deze op te stellen is echter uitgebreide kennis van de materie die de documenten behandelen vereist. Meestal vinden we een dergelijk informatie-zoeksysteem met thesaurus dan ook alleen bij gespecialiseerde, professionele informatiediensten.

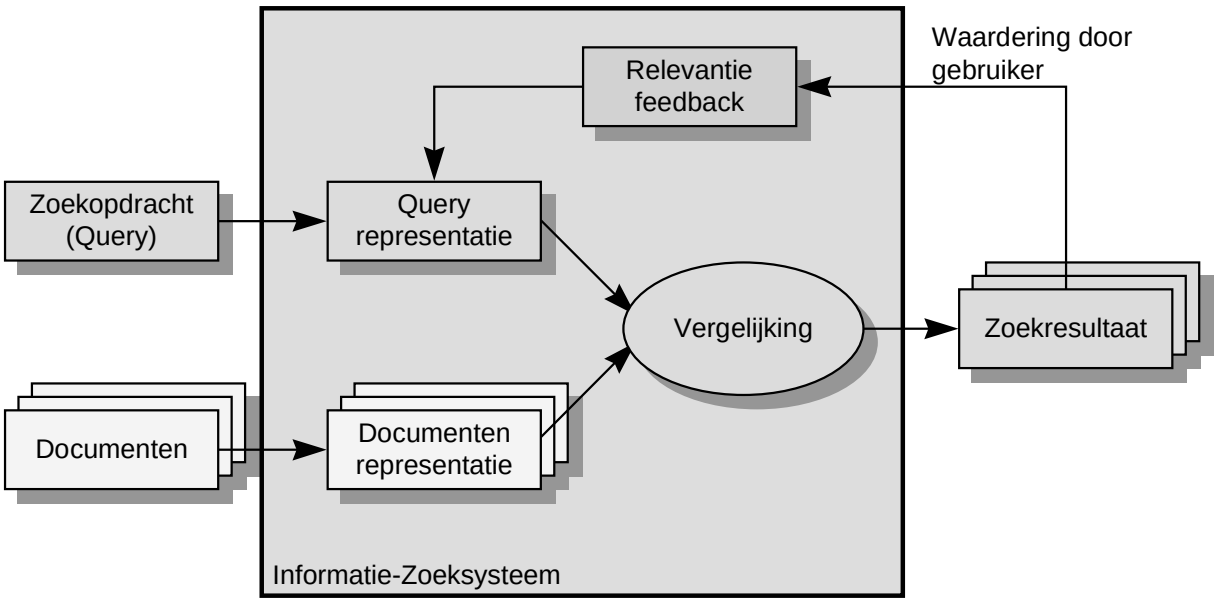
De thesaurus kan ook automatisch aangelegd worden, waarbij het systeem probeert associatieve verbindingen te vinden tussen de termen op basis van een statistische samenhang tussen de woorden. Een dergelijke thesaurus heeft niet de strikte structuur van een handgemaakte thesaurus, deze structuur is meer een connectionistisch semantisch netwerk zoals besproken in paragraaf 3.2, maar is wel makkelijker en sneller op te zetten en is voor ieder vakgebied inzetbaar.

4.3.3 Het vectorruimte-model

Bij het vectorruimte-model wordt ieder document voorgesteld als een lijst met n aantal termen waaraan per term een gewicht is toegekend. Deze lijst kan worden voorgesteld als een vector met n dimensies. De gewichten kunnen handmatig worden toegekend maar het is ook mogelijk dat deze automatisch worden toegekend. In het laatste geval zal het gewicht van een term aan de ene kant toenemen indien dit woord vaak in het document voorkomt. Aan de andere kant zal het gewicht weer afnemen als de term in veel verschillende documenten voorkomt en dus geen groot onderscheidend vermogen heeft.
De zoekopdracht kan vervolgens in bepaalde zoektermen worden gegeven, waarbij ook een gewicht aan de termen kan worden gegeven. De zoekopdracht vormt dan ook een vector. Het zoeksysteem kan de vectoren van de zoekopdracht vergelijken met die van de documenten. De documenten met de meest gelijkende vector worden dan in volgorde aan de gebruiker getoond.

4.3.4 Relevantie Feedback

Aangezien de gebruiker uiteindelijk maatgevend is voor de relevantie van de documenten is de “relevantie feedback”-methode ontwikkeld: nadat het systeem het resultaat van een zoekopdracht aan de gebruiker presenteert, kan de gebruiker aangeven welke documenten ook inderdaad relevant voor haar/hem zijn. Het systeem kan deze feedback dan gebruiken bij toekomstige zoekopdrachten, door bijvoorbeeld documenten die verwant zijn aan de door de gebruiker relevant bevonden stukken aan de gebruiker te tonen.



Figuur 4.4: Een informatie-zoeksysteem met relevantie feedback (Crestani, p.456)

In *figuur 4.4* is dit schematisch weergegeven. De relevantie feedback methode maakt de zoekactie **interactief**.

4.3.5 Enige conclusies

In deze paragraaf zijn oplossingen besproken ter verbeterin van de beperkte effectiviteit van de Boolse informatie-zoeksystemen. Het gebruik van woordstam-reductie, thesauri of toepassing van het vectorruimte-model of relevantie feedback zoals is besproken, kan inderdaad leiden tot een verhoging van de effectiviteit. Veelal kunnen deze methoden op twee manieren worden geïmplementeerd; met handmatige of met automatische kennisopbouw. De handmatige methode is in het algemeen slechts inzetbaar voor kleine documentencollecties die betrekking hebben op slechts enkele specifieke vakgebieden. Deze methode is ook arbeidsintensief en dus kostbaar.
Het automatisch ontwikkelen van kennis over de concepten waarop de zoekopdracht en de documenten betrekking hebben is dus wenselijk en voor de meeste documentencollecties zelfs noodzakelijk.

4.4 Informatie-zoeksystemen met neurale netwerken

In de afgelopen decennia zijn er informatie-zoeksystemen ontwikkeld die **specifiek** gebaseerd zijn op neurale netwerken. Deze systemen zijn speciaal gericht op het implementeren van het leermechanisme, zoals dat min of meer bij mensen opereert, in een informatie-zoeksysteem. Deze systemen zijn dus gespecialiseerd in het automatisch opbouwen van kennis.
De systemen die in de volgende paragrafen worden beschreven laten een ontwikkeling zien, waarbij ze in steeds hogere mate gaan voldoen aan de genoemde aspecten in 4.1. (Weustink, p.11-13)

4.4.1 Connectionistishe systemen

Een voorbeeld van een connectionistisch zoeksysteem is het **Interactief Activeringssysteem.** Dit systeem bestaat uit twee sets: een set met beschrijvingen van documenten en een set documenten. Deze beschrijvingen van de documenten zijn indextermen die automatisch afgeleid kunnen worden van de documenten. Als het systeem gebruikt wordt, dan wordt door middel van een query van de gebruiker de unit met beschrijvingen geactiveerd, die op zijn beurt de unit met de documenten activeert. Het neurale activeringsniveau wordt gebruikt om de relevantierangschikking van het document aan te geven. Het nadeel van dit systeem is dat de mogelijkheid voor het leren ontbreekt. Het is niet mogelijk om de gewichten tussen de twee sets aan te passen bij het toevoegen van nieuwe documenten.
Als oplossing hiervoor is het **Adaptive Information Retrieval System (AIR)** ontwikkeld. Dit systeem heeft dezelfde structuur als het vorige systeem, dat wil zeggen dat documenten en termen worden gedefinieerd als nodes die verbonden zijn door middel van gewichten. Wanneer een query geplaatst wordt, dan wordt door middel van een term een gerelateerd document geactiveerd. Wat nieuw is bij dit systeem is de mogelijkheid om te leren. Dit leer-algoritme heeft de volgende drie hoofdeigenschappen (Scholtes):

- Het toevoegen van nieuwe documenten resulteert in het aanpassen van de gewichten tussen de indextermen en de documenttermen. Dit heeft tot gevolg dat indextermen die meer voorkomen een lagere activeringsniveau genereren.
- Het systeem is zichbewust van contextuele relaties als gevolg van het aanpassingskarakter van de documentstructuren. Woorden die regelmatig in elkaars buurt voorkomen verhogen ook elkaars activeringsniveau.
- Gebruikers kunnen de mate van correctheid van de opgevraagde documenten aangeven. Hiermee worden de gewichten tussen de nodes aangepast. Deze methode van interactief bladeren is een zeer efficiënte manier van relevantie feedback.

4.4.2 Volledig gedistribueerde informatie-zoeksystemen

Het connectionistische systeem staat bekend om zijn implementatie- en onderhoudsproblemen (Scholtes). Begin jaren ’90 was er een verschuiving waar te nemen van connectionistiche systemen naar volledig gedistribueerde systemen. In deze paragraaf worden een paar voorbeelden gegeven.

- **Back-propagation network**.

Om een collectie van documenten beter te kunnen benaderen is het mogelijk de documenten op te delen in categorieën[2]. Het indelen van de documenten in de categorieën wordt meestal met de hand gedaan. Het Back-propagation algoritme (Hertz e.a, p.115 ff.) voor neurale netwerken is bij uitstek geschikt om gegevens te classificeren in categorieën, indien deze documenten gerepresenteerd worden als een activerings-patroon (vector).
Er zijn verschillende manieren om een document als vector weer te geven. Eén methode is de vectorruimte-methode zoals besproken in paragraaf 4.3.3. Een andere methode is die van Mitzman en Giovannini (Scholtes), waarbij gebruik wordt gemaakt van een **n-gram** om documenten te representeren.

[2] De internet-zoekmachine Yahoo is hier een goed voorbeeld van.

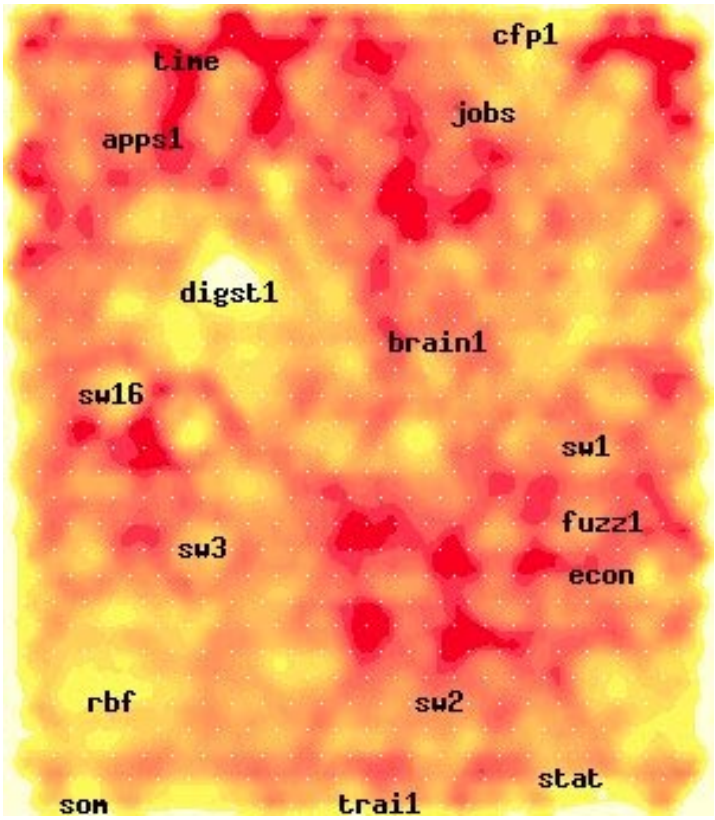
Een voorbeeld van een n-gram representatie is de bi-gram methode. Een document wordt gerepresenteerd als een vector van x bij x elementen, waarbij x het aantal verschillende leestekens voorstelt die worden gedetecteerd. Ieder element van de vector geeft aan hoe vaak een bepaalde twee-letter-combinatie in het document voorkomt.
Bij een test werden 5000 documenten handmatig geclassificeerd. Aan het back-propagation netwerk werd tijdens training als input een document-vector uit deze set gepresenteerd, en als verwachte output de handmatig bepaalde categorie. Middels het back-propagation algoritme wordt tijdens de training de gewichten van de interne verbindingen in het neurale netwerk zodanig aangepast dat het netwerk na verloop van tijd zelf redelijk in staat is de classificatie uit te voeren. Na de training werd het systeem dan ook onderworpen aan een test, waarbij het systeem 95% van een verzameling van 25.000 documenten juist wist te classificeren.
Dit systeem heeft echter als nadeel dat het slechts een oppervlakkige tekstanalyse uitvoert en dus niets over de semantiek van de woorden weet. Ook is het bijna onmogelijk dit systeem verder uit te breiden naar bijvoorbeeld een tri-gram, aangezien dit een explosieve groei van de vector en dus ook het neurale netwerk tot gevolg heeft. Hiernaast heeft dit systeem ook als nadeel dat de categorieën van te voren vast moeten worden gesteld.

- **Simple recurrent networks**

Evenals het vorige systeem probeert Stefan Wermter documenten te classificeren in categorieën (Scholtes). In tegenstelling tot het vorige systeem wordt hier echter gebruik gemaakt van **unsupervised learning** (Hertz e.a., p.197 ff.). Hierdoor is het mogelijk om tijdens het trainingsproces nieuwe categorieën aan te maken.
Om de grote hoeveelheid van neuronen te beperken wordt alleen gebruik gemaakt van de woorden in de titels van de documenten. Hierdoor wordt de capaciteit van het systeem beperkt.

- **Self-Organizing maps**

Een vorm van unsupervised learning is het Self-Organizing Map (SOM) algoritme (Kohonen). In plaats van het verdelen van de documenten in categorieën worden ze bij de WEBSOM methode ingedeeld in clusters van semantisch gelijkende documenten (Kohonen e.a.1).
De gebruikersinterface van deze methode geeft de gebruiker de mogelijkheid om zijn query te formuleren door een punt te selecteren op een tweedimensionale kaart zoals weergegeven in *figuur 4.4*. Alle documenten zijn op deze kaart ondergebracht bij een cluster, waarbij documenten met ongeveer dezelfde semantische inhoud in dezelfde cluster zijn ingedeeld. Aangrenzende clusters hebben betrekking op ongeveer dezelfde onderwerpen. Ook kan op de kaart worden afgelezen welke clusters meer documenten bevatten: lichte gebieden op de kaart bevatten relatief meer documenten dan donkere.



Explanation of the symbols on the map

apps1 - applications: face, speech
brain1 - brain sized NN
cfp1 - conferences
digst1 - Neuron Digest, CFPs
econ - finance
fuzz1 - fuzzy logic
jobs - vacancies
rbf - radial basis function networks
som - Kohonen SOMs
stat - NN vs statistics
sw1 - implementations
sw16 - software
sw2 - software
sw3 - source code
time - time series
trai1 - training, testing

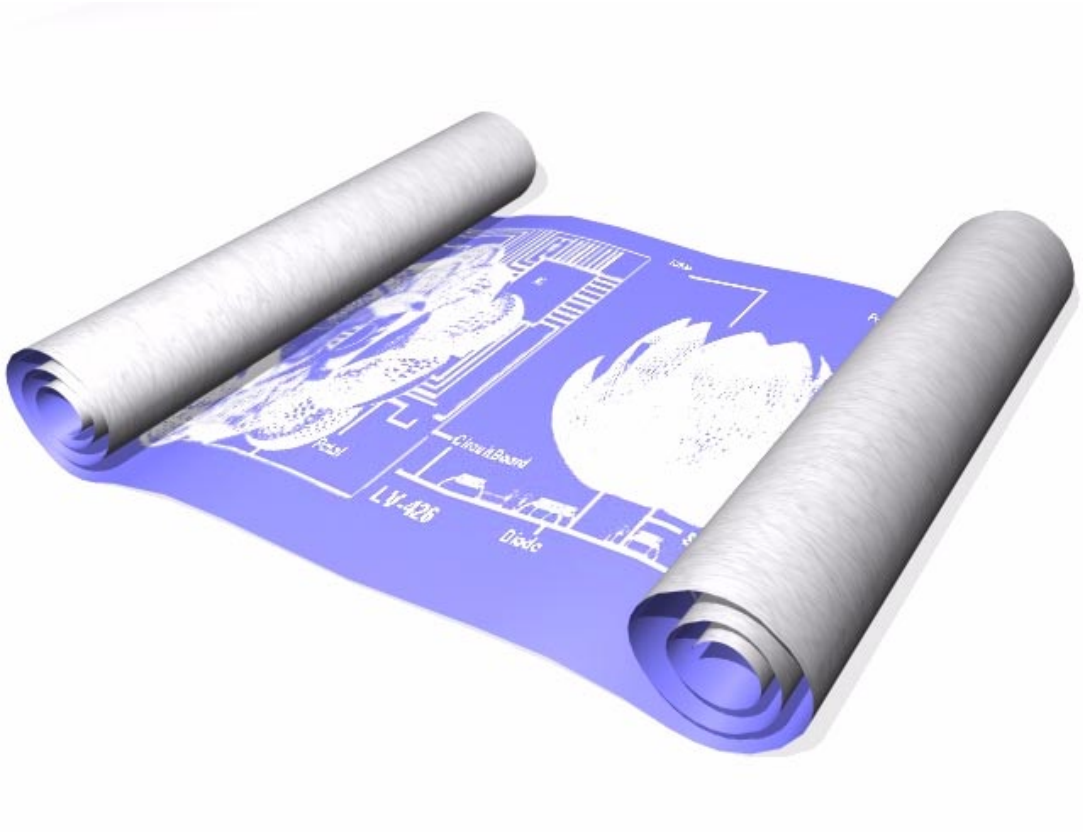
Figuur 4.5: Demonstratie van het WEBSOM algoritme met gebruik van de documenten uit comp.ai.neural-nets. (http://websom.hut.fi/websom/comp.ai.neural-nets/html/root.html)

De WEBSOM afbeelding in *figuur 4.5* is een afbeelding met meerdere niveaus om het geheel overzichtelijk te houden. Indien de gebruiker geïnteresseerd is in een deel van de kaart dan kan zij/hij hierop inzoomen, waarbij meer detail kan worden getoond van het deelgebied.
In hoofdstuk 5 zal de werking van het WEBSOM-algoritme in groter detail worden uitgelegd. Wel kan alvast worden opgemerkt dat het WEBSOM algoritme gebruik maakt van de volledige tekst van een document en daardoor slechts inzetbaar is voor collecties met kleine documenten

4.5 Afsluitende conclusies

In dit hoofdstuk is gebleken dat de effectiviteit van een informatie-een functie is van precisie en vindkracht. Vervolgens is een ontwikkeling beschreven die informatie-zoeksystemen in de afgelopen decennia hebben doorgemaakt waarbij steeds geprobeerd is deze effectiviteit te verbeteren.
Boolse zoeksystemen zijn momenteel nog het meest in de praktijk toegepast, maar missen de kennis om inzicht te krijgen in de semantische betekenis van zoektermen en documenten. De meest veelbelovende nieuwe ontwikkelingen, met een hoge effectiviteit tegen betaalbare kosten, vormen de automatische gegenereerde thesauri en de WEBSOM methode. Deze laatste heeft nog als extra voordeel dat de gebruiker via een unieke interface

de documenten kan zoeken. Hierdoor kan het systeem zelfs met vage gebruikersintenties goed werken.
Zoals al opgemerkt heeft deze methode momenteel een groot nadeel: het kan alleen kleine documenten verwerken. De rest van deze scriptie zal dan ook gewijd zijn aan het ontwikkelen van een systeem dat deze tekortkoming niet heeft en verder niet voor het WEBSOM algoritme onderdoet.



Hoofdstuk 5: Naar een architectuur van een knowledge-based Informatie Zoeksysteem

“Nu, laten we dan volgens onze gebruikelijke methode beginnen met vast te stellen dat we in verband met elke groep voorwerpen waaraan we dezelfde naam geven, het bestaan van een algemeen type aannemen.” (Plato, p.245)

Nadat in hoofdstuk 2 is geprobeerd het begrip ‘kennis’ te definiëren is in hoofdstuk3 behandeld hoe kennis kan worden opgeslagen in een computer. Hoofdstuk 4 geeft een overzicht van informatie-zoeksystemen. In dit hoofdstuk kan nu worden bekeken hoe informatie-zoeksystemen, die kennis hebben over de semantiek van documenten, eruit zouden moeten zien. De inhoud van de vorige hoofdstukken speelt hierbij een belangrijke rol.
Eerst zullen een tweetal voorbeelden van zulke knowledge-based informatie-zoeksystemen aan bod komen, namelijk de WEBSOM-methode (5.1) en de Aqua-browser (5.2). Aan de hand van deze voorbeelden zal een algemene architectuur voor dergelijk systemen worden afgeleid(5.3).
Het doel van deze scriptie is het ontwerpen van een nieuw informatie-zoeksysteem met kennis dat kennis kan extraheren uit de indexen van boeken in plaats van uit de volledige

tekst. Dit systeem zal ook de volgens de gevonden algemene architectuur moeten zijn opgebouwd. Een eerdere poging tot het ontwerpen van zo'n systeem waarbij nog geen rekening is gehouden met de hier voorgestelde architectuur, het INDEXSOM, zal dan ook beschreven worden, samen met de problemen die hierbij zijn gerezen (5.4).

5.1 Het WEBSOM-algoritme

In het vorige hoofdstuk is al kort stil gestaan bij het WEBSOM-algoritme. In deze paragraaf zal dieper in worden gegaan op de precieze werking van dit algoritme.
In hoofdstuk 3 is behandeld hoe Self-Organizing Maps gebruikt kunnen worden om een conceptuele ruimte op te zetten. Dit vormt de basis voor het WEBSOM-algoritme dat werkt met documenten die met name zijn opgebouwd uit ASCII-karakters. Het WEBSOM-algoritme is een uitbreiding op het in paragraaf 3.3.3 behandelde SOM-algoritme en heeft een aantal specifieke eigenschappen:

5.1.1 Voorbewerking

Als eerste worden alle documenten achter elkaar gezet in één groot in bestand. Vervolgens worden alle niet-relevante karakters en woorden verwijderd (Honkela e.a., 1996):

- ASCII-tekeningen worden verwijderd. Ook alle getallen en andere niet-alphabetische karakters worden weggelaten.
- Om de hoeveelheid rekenwerk te verlagen kunnen woorden die weinig voorkomen in de tekst (bijvoorbeeld minder dan 50 keer) gewist worden.
- Veel voorkomende algemene woorden zonder specifieke betekenis worden ook verwijderd. Hierbij valt te denken aan woorden zoals 'echter', 'doch', 'dus', 'en', 'uiteraard' en lidwoorden.

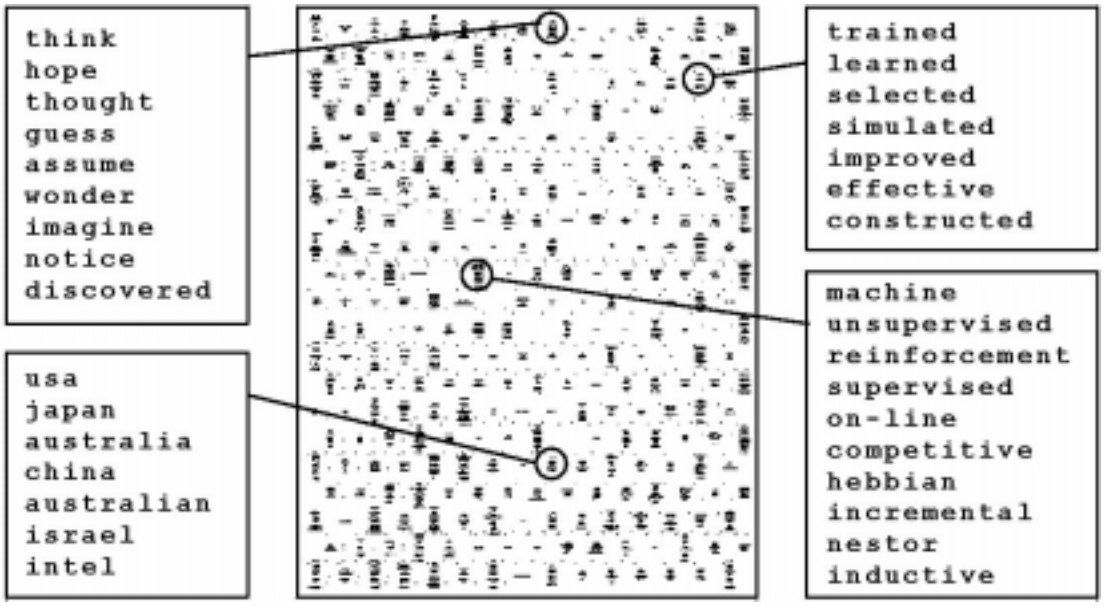
5.1.2 Woord-cluster-afbeelding

De woord-cluster-afbeelding is een SOM zoals beschreven in paragraaf 3.3.3 en beschrijft de relaties tussen woorden op basis van hun nabije context, namelijk het woord ervoor en het woord erna. Woord i wordt gerepresenteerd door de n-dimensionale vector x_i met willekeurige getallen als componenten. Hiervan wordt de gemiddelde context-vector opgesteld (Honkela e.a., 1996, p.6) :

$$X(i) = \begin{bmatrix} E\{ x_{i-1} \mid x_i \} \\ \varepsilon\, x_i \\ E\{ x_{i+1} \mid x_i \} \end{bmatrix}$$

Hierin is E de conditionele verwachting van de waarde, gebaseerd op de relatieve frequentie van het woord in de totale tekst: Oftewel, iedere keer dat woord x_i in de tekst voorkomt wordt genoteerd welke vector x_{i-1} eraan vooraf gaat. Door deze vectoren x_{i-1} bij elkaar op te tellen en te delen door het aantal wordt de gemiddelde conditionele vector bekend: Dit gemiddelde noemen we $E\{ x_{i-1} \mid x_i \}$. Door gebruik te maken van het gemiddelde hoeft niet iedere woordcombinatie apart aan de SOM gevoerd te worden en wordt de rekentijd verminderd.

ε is een klein scalair getal. Dit getal zorgt ervoor dat de nadruk bij woorden op hun context komt te liggen en niet het woord zelf. Nu vormt $\mathrm{X}(i) \in \Re^{3n}$ de input-vectors voor de SOM dat we verder de woord-cluster-afbeelding zullen noemen. Gebruikelijke waarden zijn $\varepsilon = 0.2$ en $n = 90$. De aanpassing van de referentievectoren geschiedt nu zoals is beschreven in paragraaf 3.3.3.
Door een SOM te kiezen dat minder eenheden bevat dan dat er woorden zijn in de (voorbewerkte) vocabulaire zal de SOM gedwongen zijn om meerdere woorden te representeren door één eenheid, oftewel verschillende woorden kunnen leiden tot de activering van dezelfde eenheid. We kunnen zeggen dat de SOM **clusters** onderkent in de conceptuele ruimte die wordt gedefinieerd door de context-vectoren. Woorden die in dezelfde cluster vallen zullen in het algemeen in dezelfde context voorkomen en hopelijk semantisch sterk aan elkaar verbonden zijn. In *figuur 5.1* is een voorbeeld weergegeven van zo'n woord-cluster-afbeelding, waar de inhoud van enkele van deze clusters is getoond.

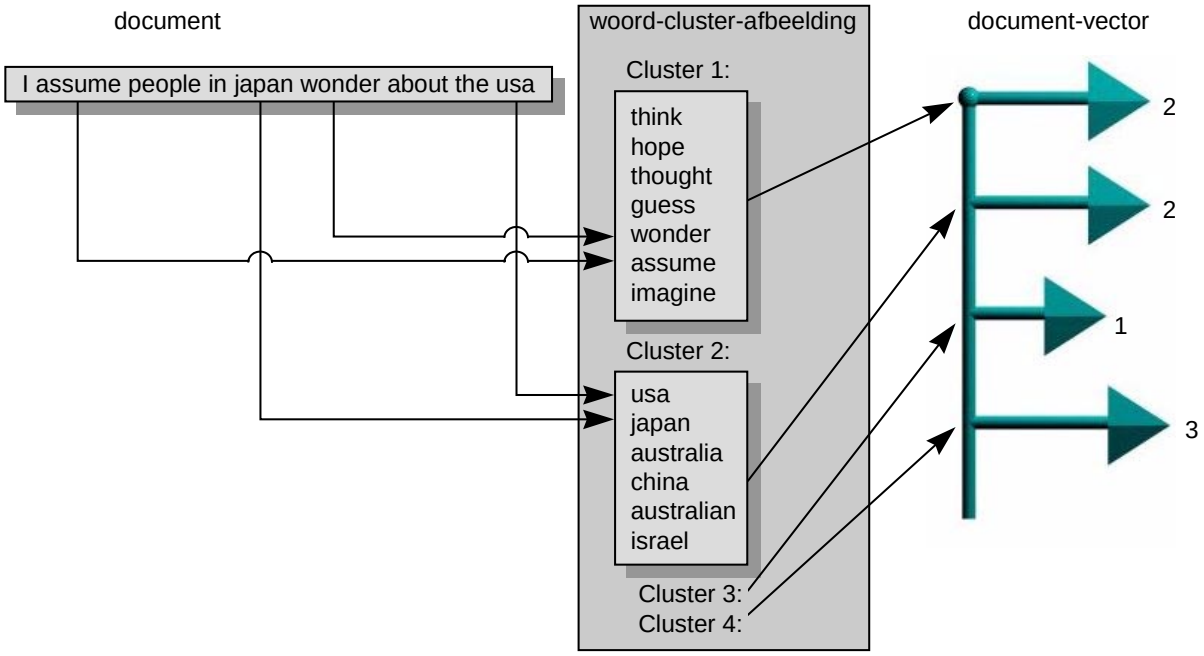


Figuur 5.1: Voorbeeld van een woord-cluster-afbeelding met enkele duidelijke categorieën. In het midden een overzicht van de gehele afbeelding. Aan de zijkant zijn een viertal clusters uitvergroot. (Honkela e.a., 1996, p.7)

Als we ervan uitgaan dat inderdaad de woorden in een cluster ongeveer dezelfde betekenis hebben dan maakt het voor de semantiek van het document niet veel uit als een woord uit deze cluster vervangen wordt door een ander woord uit dezelfde cluster.

5.1.3 Document-cluster-afbeelding

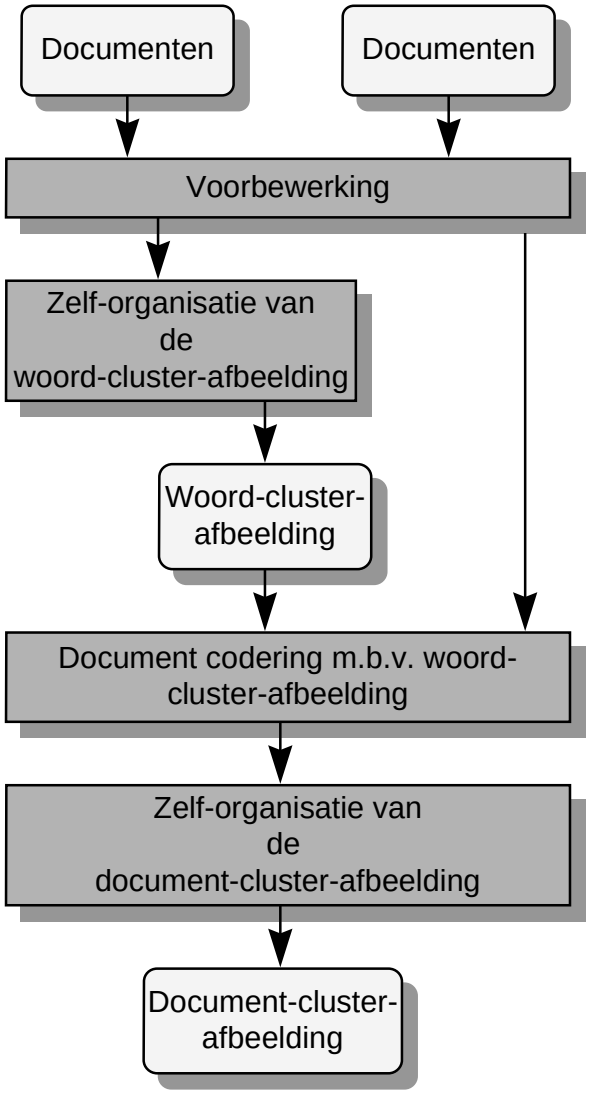
In paragraaf 4.3.3 is beschreven hoe bij het vectorruimte-model documenten voor worden gesteld als vectoren, waarbij een component uit deze vector verbonden wordt met een specifieke term en de waarde van deze component maatgevend is voor het aantal keer dat deze term in het document voorkomt. Het zou mogelijk kunnen zijn om deze vectoren als input te gebruiken voor een SOM waarbij documenten geordend zouden worden naar hun inhoud. Wat dan ontstaat zullen we een **document-cluster-afbeelding** noemen: Documenten die qua inhoud aan elkaar gerelateerd zijn zullen op deze afbeelding veelal dicht bij elkaar liggen. Bij een documentencollectie van redelijke omvang is de gebruikte vocabulaire echter al snel honderdduizenden woorden groot. Zelfs na een voorbewerking zoals in paragraaf 5.1.1 blijven er nog zo'n 6.000 woorden over. (Kohonen e.a.1) Dit grote aantal leidt ertoe dat een SOM slecht convergeert en veel processortijd en geheugen nodig heeft (Lagus).
Om dit probleem op te lossen is het mogelijk de werkelijke woorden uit de tekst te vervangen door een verwijzing naar de cluster waartoe dit woord behoort in de woord-cluster-afbeelding zoals weergegeven in *figuur 5.2***.**



Figuur 5.2: Codering van de documenten met behulp van de woord-cluster-afbeelding

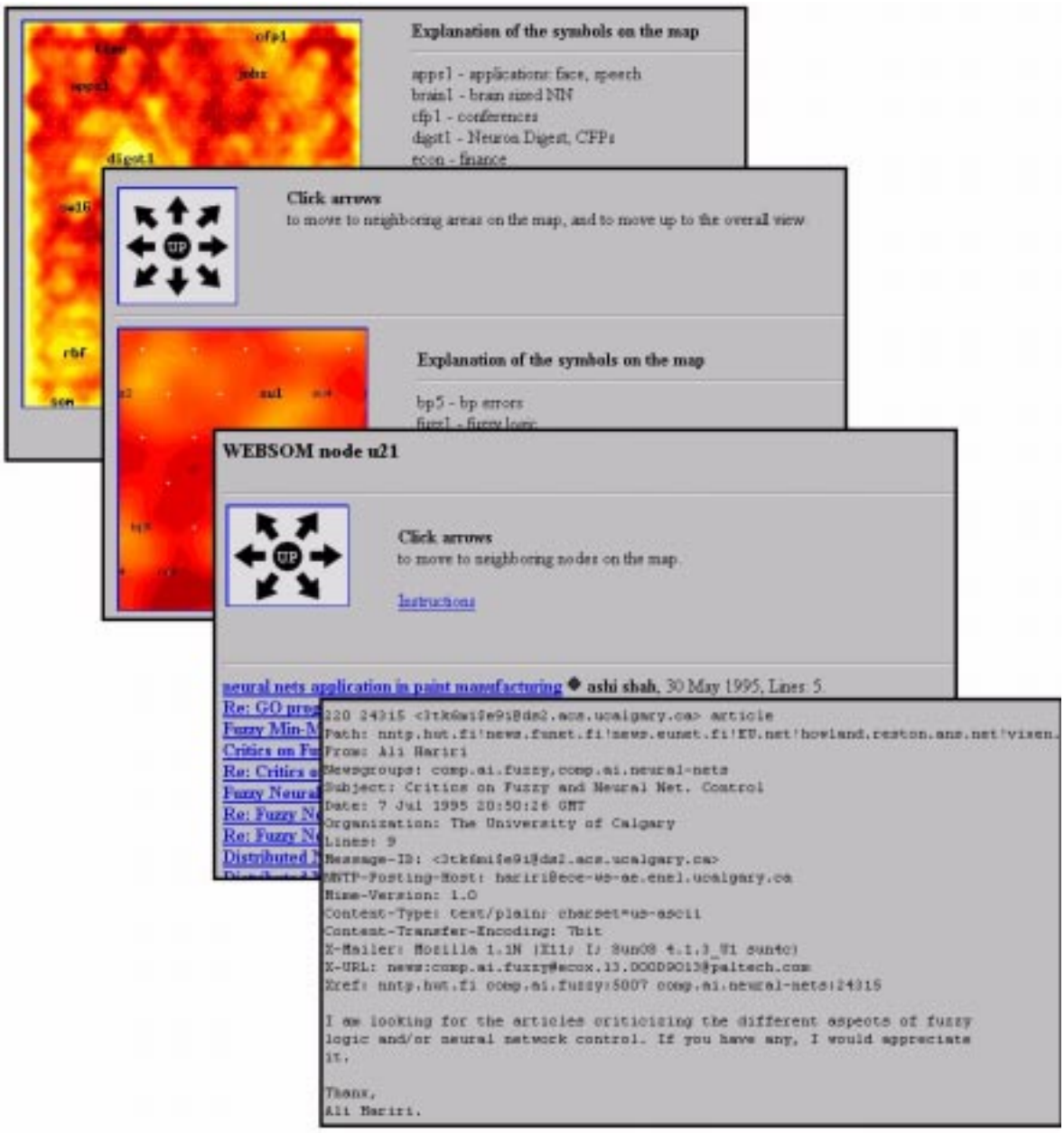
Hierdoor kan het aantal elementen in de vector worden gereduceerd tot enkele honderden, zodat deze beter geschikt is voor een SOM (Kohonen e.a.1). Het aantal componenten in de inputvector voor de document-cluster-afbeelding is gelijk aan het aantal eenheden in de woord-cluster-afbeelding. De waarde van de component is dan afhankelijk van de frequentie waarin woorden die behoren tot één eenheid voorkomen in het betreffende document.
In *figuur 5.3* is een overzicht weergegeven van het WEBSOM-algoritme.
De clusters op de document-cluster-map kunnen worden voorzien van een label. De gebruiker kan nu op de afbeelding die cluster selecteren die zij/hij interessant vindt. Documenten in deze cluster zullen vaak hetzelfde onderwerp behandelen en documenten in nabije clusters zullen qua semantische inhoud vaak ook gerelateerd zijn aan deze

documenten. Indien de document-cluster-afbeelding erg groot is dan kan deze in meerdere detail-niveaus worden opgebouwd, zodat de gebruiker op de afbeelding kan inzoomen waarbij steeds meer details worden getoond.
In *figuur 5.4* is weergegeven hoe de gebruikersinterface van de WEBSOM[3] eruit ziet. In de documenten-cluster-afbeelding is met behulp van kleur-gradaties weergegeven hoeveel documenten zich in één cluster bevinden: een lichte kleur duidt op een hoge concentratie en een donkere kleur duidt op een lage concentratie van documenten.



Figuur 5.3: De architectuur van de WEBSOM methode. De document-cluster-afbeelding is georganiseerd op basis van de documenten die gecodeerd zijn met behulp van de woord-cluster-afbeelding. Beide afbeeldingen worden gemaakt met het SOM-algoritme.

[3] Een werkende demonstratie van het WEBSOM-algoritme kan gevonden worden op het internet: http://websom.hut.fi/websom/comp.ai.neural-nets/html/root.html



Figuur 5.4: De vier verschillende niveaus van de WEBSOM zoek-interface. Linksboven is de volledige document-cluster-afbeelding weergegeven. Hiernaast is een ingezoomde afbeelding weergegeven, waarna de documenten zijn afgebeeld die tot één eenheid of cluster behoren. Als laatste is weergegeven hoe één van de desbetreffende documenten is afgebeeld.

5.1.4 Evaluatie van het WEBSOM-algoritme

Met behulp van de zes criteria die in paragraaf 4.1 zijn behandeld is het enigszins mogelijk de prestaties van het WEBSOM-algoritme te evalueren:

1. Het **bereik van de collectie**: Aangezien ieder document woord voor woord moet worden doorzocht is het alleen mogelijk dit algoritme toe te passen op kleine documenten (Weustink, p.30,31). Dit vormt één van de voornaamste probleempunten van het WEBSOM-algoritme.
2. De **vertraging**: Aangezien het vormen van de beide cluster-afbeeldingen al van te voren gebeurt is de vertraging tussen de initiatie van zoekactie door de gebruiker en het genereren van het resultaat door de computer zeer klein.
3. De **vorm van de presentatie** van de output: Het WEBSOM-algoritme heeft een grafische gebruiksinterface waardoor de gebruiker gemakkelijk kan zoeken.
4. De **moeite** die de gebruiker moet doen om de antwoorden te krijgen van het systeem: Door de unieke gebruikersinterface is het mogelijk voor de gebruiker om slechts met enkele klikken van de muis de voor haar/hem interessante documenten te vinden. Aangezien de gebruiker kan kiezen uit een aantal van te voren zichtbare clusters is het voor de gebruiker ook mogelijk te zoeken zonder een duidelijke informatiebehoefte. De gebruiker kan dan die cluster uitkiezen die het meest relevant lijkt en later een meer gedetailleerde ‘query’ geven, gebaseerd op de clusters die dan verschijnen.
5. De **vindkracht** van het systeem: De WEBSOM-methode is alleen nog toegepast in een prototype. Onderzoek naar de vindkracht en precisie moet nog worden verricht. Gegeven het feit dat dit systeem middels het gebruik van de woord-cluster-afbeelding rekening houdt met de semantische betekenis van woorden is af te leiden dat de vindkracht van dit systeem in het algemeen hoger ligt dan bijvoorbeeld bij geïnverteerde indexes: Synoniemen en andere sterk gerelateerde woorden zullen door het WEBSOM als gelijk worden behandeld. Hierdoor zal het systeem minder snel een relevant document missen.
6. De **precisie** van het systeem: De documenten worden ingedeeld in clusters waarbij documenten binnen een cluster over het algemeen dezelfde onderwerpen behandelen. Als een document uit een cluster relevant is voor de gebruiker dan is de kans groot dat de overige documenten uit deze cluster ook relevant zijn. We kunnen dan stellen dat een groot deel van de gevonden documenten relevant is oftewel dat het systeem een hoge precisie behaalt.

5.2 De Aqua-browser

De Aqua-browser (Veling) is een informatie-zoeksysteem dat is ontwikkeld door Medialab, een onderdeel van Origin. Dit systeem is gebaseerd op een Connectionistisch Semantisch Netwerk (CSN) zoals beschreven in paragraaf 3.2. Eerst zal worden beschreven hoe dit CSN tot stand komt (5.2.1), waarna de Aqua-browser zelf wordt besproken (5.2.2). Als laatste zal dit systeem kort worden geëvalueerd (5.2.3).

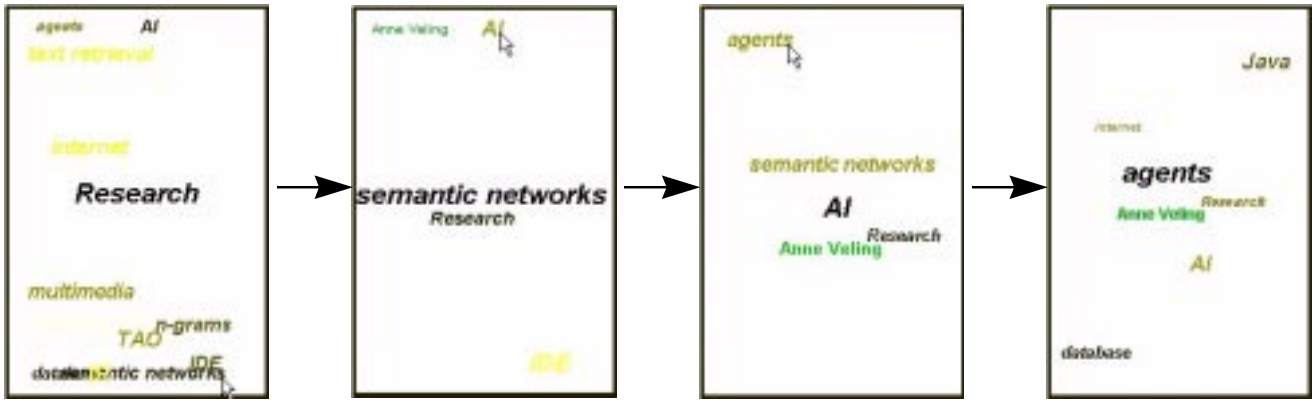
5.2.1 Opbouw van het CSN

Het semantisch netwerk kan op meerdere manieren worden opgebouwd. Zo kan de inhoud van dit netwerk gebaseerd zijn op een bestaand kennissysteem dat hetzelfde domein betreft als de documentencollectie, of een bestaande thesaurus.
Ook kan het netwerk worden ingevuld met behulp van een analyse van de documenten zelf. In dit geval worden nadat in een voorbewerking alle niet-relevante woorden en tekens zijn verwijderd de overgebleven woorden in een semantisch netwerk geplaatst. De gewichten van de verbindingen worden nu als volgt bepaald: Indien in de tekst twee woorden uit het semantisch netwerk dicht bij elkaar worden aangetroffen dan wordt de verbinding tussen deze woorden versterkt, afhankelijk van de afstand tussen deze woorden. Indien de woorden dichter bij elkaar liggen zal de verbinding sterker worden gemaakt.
Een factor die ook wordt meegenomen is de frequentie waarmee de woorden voorkomen in de tekst. Als een woord vaak voorkomt in tekst zal dit anders resulteren in een grote hoeveelheid sterke verbindingen met dit woord, misschien zelfs met alle andere woorden in het netwerk. Via spreading activation zou dan iedere knoop binnen twee stappen te bereiken zijn vanuit iedere willekeurige andere knoop. We kunnen zeggen dat teveel sterke verbindingen aan een knoop weinig informatie toevoegt en daarom zal een verbinding met een woord dat vaak voorkomt zwakker moeten worden gemaakt.

5.2.2 De browser

Het opbouwen van een CSN op basis van de woordanalyse zoals in de vorige paragraaf is beschreven is niet uniek aan de Aqua-browser. Hoe de gegevens van het CSN worden gebruikt voor een informatie-zoeksysteem echter wel.
In *figuur 5.5* is weergegeven hoe de gebruikersinterface van deze browser eruit ziet. In het midden is de term weergegeven waarop op dat moment de nadruk ligt. In het begin kan dit een willekeurige term zijn. De knoop die aan deze term is gekoppeld wordt geactiveerd en als gevolg hiervan worden middels de verbindingen andere knopen geactiveerd volgens het spreading-activation principe. De termen die aan deze knopen zijn verbonden worden ook weergegeven, waarbij geldt dat een term met hogere activering meer op de voorgrond staat, wat in dit geval betekent dat deze met een duidelijkere kleur en een groter lettertype is weergegeven.
De gebruiker kan nu één van deze andere termen selecteren met de muis. Hierdoor verandert de nadruk op deze term en krijgt deze term de hoogste activatie. De oude activering blijft echter gedeeltelijk bestaan en wordt nu gecombineerd met de activering van het nieuwe woord. Hierdoor worden vooral die woorden geactiveerd die zowel sterk gerelateerd waren aan het vorige woord als aan het huidige woord.

Op deze manier kan de gebruiker het semantische netwerk doorzoeken door iedere keer de meest relevante term te selecteren. Uiteindelijk zal de gebruiker dan de voor haar/hem meest relevante term bereiken.



Figuur 5.5: Concepten zoeken in een CSN met de Aqua-browser. De cursor geeft aan welk woord aangeklikt is om naar de volgende afbeelding te komen. Het semantisch netwerk is gevuld met gegevens van de website van medialab.
(http://www.medialab.nl/documents/research/aqua/aquaframes.htm)

Aan een term kan dan één of meerdere documenten worden gekoppeld. Of beter gezegd: Aan een document worden enkele van de termen toegekend. Dit indexeren kan handmatig gebeuren of kan geschieden op basis van het aantal keren dat de term in dit document voorkomt. Ook kunnen de woorden waarvan de knopen aan het einde van de zoekactie de hoogste activering hebben als input worden gebruikt voor een Boolse zoekactie.

5.2.3 Evaluatie van de Aqua-browser

1. Het **bereik van de collectie**: Voor de automatische opbouw van het CSN geldt, net als voor de WEBSOM-methode, dat de collectie uitsluitend uit kleine documenten mag bestaan. Dit aangezien het aantal termen anders te groot wordt en de gebruiker te veel keuze-mogelijkheden krijgt.
2. De **vertraging**: Het CSN wordt van tevoren gevormd. De uitvoering van de spreading activation berekeningen tijdens de query is zeer kort en levert in ieder geval voor kleine documentencollecties geen vertraging op. Onduidelijk is hoe dit systeem zich bij grote collecties gedraagt.
3. De **vorm van de presentatie** van de output: Dit is één van de sterkste punten van de Aqua-browser. De gebruikersinterface is zeer intuïtief en makkelijk door de gebruiker te doorzien. De gebruiker kan op ieder moment zien welke concepten voor haar/hem belangrijk zijn (Bloem 1997).
4. De **moeite** die de gebruiker moet doen om de antwoorden te krijgen van het systeem: Ook hier geldt dat de gebruiker zelfs met vage informatiebehoeften aan de slag kan. Door te klikken met de muis op de voor de gebruiker interessante woorden kan zij/hij snel komen tot de relevante concepten en de daaraan gekoppelde documenten.
5. De **vindkracht** van het systeem: Helaas zijn er geen gegevens bekend over de vindkracht en precisie van dit systeem. Wel kan worden afgeleid dat deze afhankelijk zijn van de manier waarop de concepten zijn gekoppeld aan de documenten. Indien de

activering-niveaus van het CSN worden gebruikt voor een query in een Bools zoeksysteem dan kunnen we stellen dat de vindkracht in bijna alle gevallen hoger zal zijn met Aqua-browser dan wanneer hetzelfde Boolse zoeksysteem zonder browser wordt gebruikt. Dit aangezien de door de gebruiker relevant bevonden zoektermen worden uitgebreid met gerelateerde woorden uit het CSN.

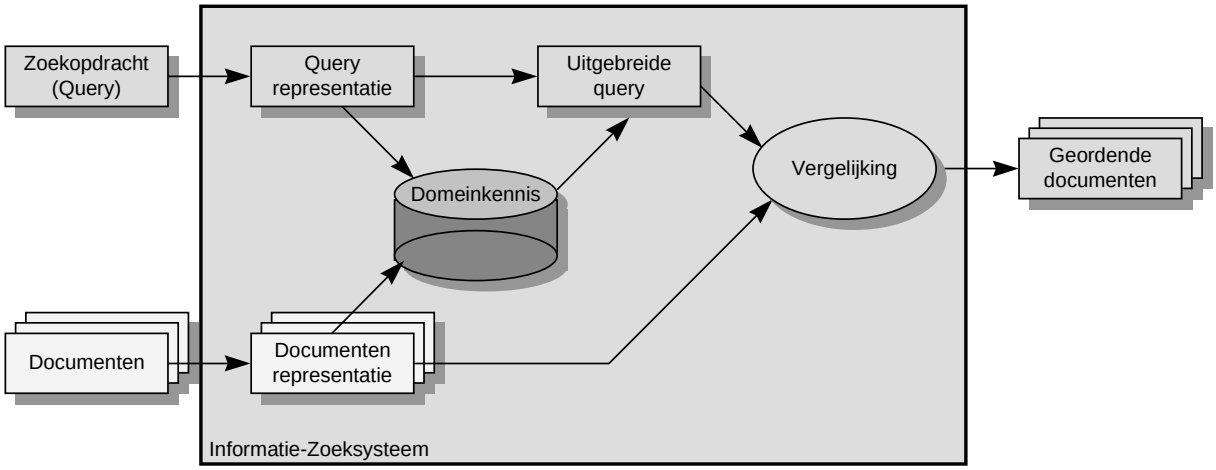
6. De **precisie** van het systeem: Gegeven het feit dat bij Boolse informatie-zoeksystemen de precisie en vindkracht onderling enigszins zijn uit te wisselen door de gebruikte zoekstrategie te wijzigen zoals behandeld in paragraaf 4.2, kan de hiervoor voorspelde toename van vindkracht omgezet worden in een hogere precisie.

5.3 Architectuur van een IZS met kennis

Nu enkele informatie-zoeksystemen met kennis zijn behandeld is het mogelijk een algemene structuur te definiëren voor een dergelijk systeem. Allereerst zal de complete architectuur worden behandeld (5.3.1) waarna de afzonderlijke componenten belicht zullen worden (5.3.2 tot en met 5.3.5)

5.3.1 Algemene architectuur

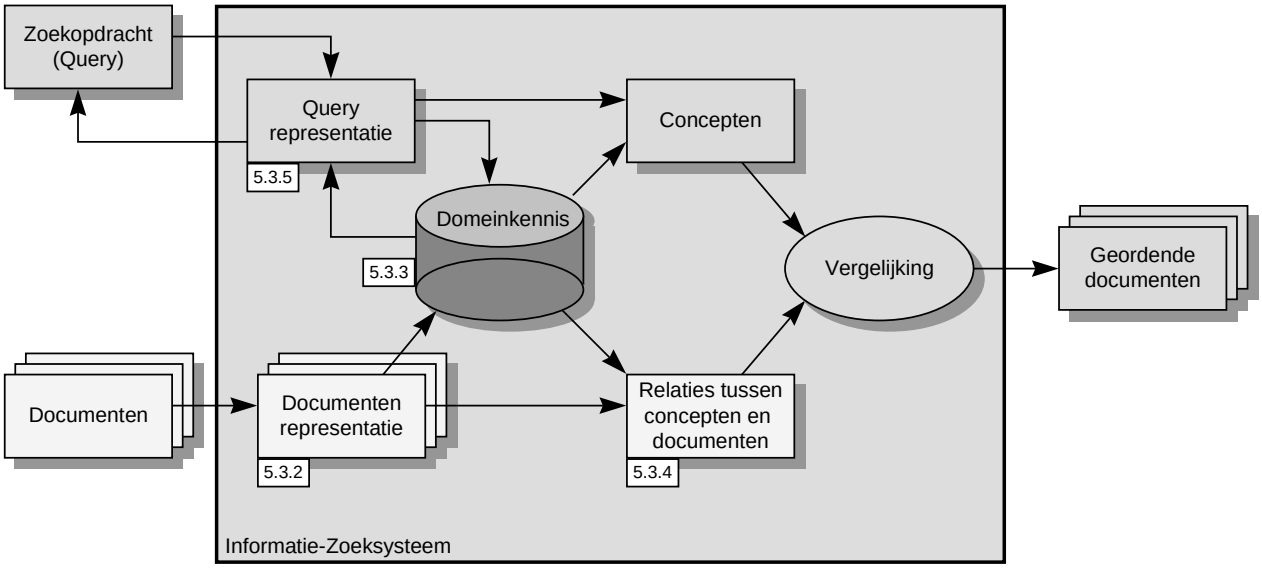
Elk informatie-zoeksysteem met zelfopgebouwde kennis moet logischerwijs over domeinkennis beschikken en over een manier om deze te extraheren uit de documenten. In het simpelste geval kan deze kennis worden gebruikt voor de uitbreiding van een query in een traditioneel informatie-zoeksysteem zoals een Bools zoeksysteem met geïnverteerde index. In *figuur 5.6* is dit weergegeven. Dit systeem komt grotendeels overeen met dat van een informatie-zoeksysteem (IZS) met thesaurus zoals beschreven in paragraaf 4.3.2. In plaats van een thesaurus is echter een brede noemer gebruikt, namelijk domeinkennis, om aan te geven dat deze kennis niet specifiek de vorm van een thesaurus aan hoeft te nemen. Daarnaast is, zoal weergegeven, deze domeinkennis in dit systeem afgeleid uit de (representatie van de) documentencollectie.



Figuur 5.6: Een eenvoudige informatie-zoeksysteem met eigen kennisopbouw

Deze architectuur is slechts een eerste stap. Omdat deze als een schil is toe te passen op een traditioneel IZS kan deze vorm makkelijk worden geïmplementeerd bij bestaande

systemen. In de vorige paragrafen is echter gebleken dat de nieuwste IZS nog een stap verder gaan. Niet alleen wordt kennis, opgebouwd uit onderling gerelateerde concepten, onttrokken aan de documenten; **ook worden de documenten gekoppeld aan deze concepten**. Hiernaast heeft de input van de gebruiker de vorm aangenomen van een interactief proces, waarbij de gebruiker de zoekactie initieert en het systeem met behulp van de domeinkennis de gebruiker ertoe zet de zoekactie verder te specificeren. Dit proces is voor te stellen als het zoeken van de gebruiker door het kennisdomein, waarbij de zoekactie resulteert in concepten die voor de gebruiker relevant zijn. Aan de hand van de koppeling tussen de concepten en de documenten kan het systeem nu de meest relevante documenten aan de gebruiker tonen. Een dergelijk IZS is weergegeven in *figuur 5.7.*



Figuur 5.7: Een uitgebreid IZS met zelf opgebouwde kennis. De nummers verwijzen naar de paragrafen waar de desbetreffende onderdelen worden behandeld.

We zien dat zo'n systeem documenten niet meer met de query vergelijkt op een woord-op-woord basis**. Van zowel de query als de documenten wordt een abstractere weergave gemaakt in de vorm van concepten** en op dit hogere abstractieniveau worden de query en de documenten nu vergeleken.
In de volgende paragrafen zullen de belangrijkste onderdelen van dit systeem worden behandeld.

5.3.2 Documentenrepresentatie

Van de documenten die binnen het bereik van het IZS liggen moet een representatie in het systeem worden gevoerd. Dit kan de complete tekst zijn van het document of bijvoorbeeld uitsluitend de index, titel en auteursnaam. Bij elektronische documenten is deze stap vaak eenvoudig: het betreft hier dan slechts een conversie van het ene bestandstype naar het andere. Indien echter de documenten uitsluitend op papier aanwezig zijn zullen deze moeten worden gescand. Scannen resulteert in beginsel in een grafische representatie van het document. Woorden en letters zijn niet meer dan een verzameling pixels op het scherm. Om de documenten toch leesbaar te maken voor de computer zullen deze beelden moeten

worden omgezet in leestekens. Dit gebeurt via Optical Character Recognition (OCR), waarbij de computer probeert letters te herkennen in het grafische bestand en vervolgens weer te geven in een tekstformaat zoals ASCII. Helaas is het proces van OCR verre van perfect. In 3-4% van de gevallen maakt de computer een fout bij het herkennen van een letter, zodat alle documenten nog steeds handmatig moeten worden nagekeken (Alexander 1997). Hierdoor liggen de kosten per te scannen pagina erg hoog.

5.3.3 Domeinkennis

In hoofdstuk 3 zijn enkele vormen van kennisrepresentatie behandeld waarmee kennis kan worden gerepresenteerd in een computersysteem. Vormen die in aanmerking komen voor een IZS dat zelf kennis opbouwt zijn Connectionistische Semantische Netwerken en conceptuele ruimte, aangezien deze vormen geheel zelfstandig door een computer zijn op te bouwen en de resulterende kennis, ook al is deze vaag en slechts een benadering, in ieder geval geschikt is voor een IZS.
Eerder in dit hoofdstuk hebben we dan ook twee systemen gezien die domeinkennis gebruiken in de vorm van respectievelijk conceptuele ruimte en connectionistische semantische netwerken:
Het WEBSOM-algoritme heeft een conceptuele ruimte in vorm van de woord-cluster-afbeelding, de Aqua-browser heeft een CSN.
Een aspect dat onlosmakelijk aan de kennisrepresentatie is verbonden is die van **kennisextractie**, oftewel: hoe wordt de kennis in de representaties van de documenten ontdekt en overgezet naar de ‘kennisbank’. Wanneer wordt gewerkt met de volledige tekst hebben we al twee verschillende methoden gezien: de korte context van de SOM en de woord-nabijheid-analyse van de Aqua-browser. De eerste methode gaat ervan uit dat semantisch gerelateerde woorden vaak worden omringd door dezelfde woorden. De tweede methode gaat ervan uit dat semantisch gerelateerde woorden dicht bij elkaar in de tekst staan.
Er zijn twee soorten relaties tussen woorden te onderkennen (Plaut):

1. Associatieve relaties: woorden waarvan de betekenis niet hetzelfde is maar die wel aan elkaar verbonden zijn. Bijvoorbeeld ‘Sleutel’ en ‘Slot’ of ‘Brood’ en ‘Boter’.
2. Semantische relaties: woorden waarvan de betekenis grotendeels overeenkomt. Bijvoorbeeld: ‘Brood’ en ‘Cake’ of ‘Boom’ en ‘Plant’.

Figuur 5.8: Voorbeeld van een associatieve en een semantische relatie.

In *figuur 5.8* is een voorbeeld gegeven van een associatieve relatie (‘gieter’ en ‘plant’) en een semantische relatie (‘plant’ en ‘boom’). We noemen de relatie tussen de concepten ‘plant’ en ‘boom’ semantisch omdat deze concepten elkaar overlappen. Zowel een plant als een boom zijn bijvoorbeeld levende dingen, kunnen groeien, hebben bladeren en wortels, zijn opgebouwd uit aminozuren en zijn meestal overwegend groen.

De concepten ‘gieter’ en ‘plant’ zijn ook aan elkaar gerelateerd aangezien een gieter meestal gebruikt zal worden voor het besproeien van planten. ‘Gieter’ en ‘plant’ zijn echter niet sterk semantisch gerelateerd: één van de weinige dingen die zij gemeen hebben is dat het in beide gevallen een object betreft. Associatieve relaties zijn meer subjectief in die zin dat ze per persoon verschillend kunnen zijn en zelfs bij één persoon van tijd tot tijd kunnen veranderen. Bijvoorbeeld iemand die gieters lange tijd heeft gebruikt voor schoonmaak-werkzaamheden zal ‘gieter’ eerder associëren met ‘sop’ dan met ‘plant’.
Het WEBSOM zal, door zijn nadruk op de korte context, vooral semantische relaties vinden. Dit aangezien semantisch gelijke woorden vaak op dezelfde manier worden toegepast in een zin en dus dezelfde context hebben. Bijvoorbeeld ‘Eet je -brood- op’ en ‘Eet je -cake- op’ of ‘ik gaf de -boom- water’ en ‘geef de -plant- water’.
Het CSN van de Aqua-browser zal daarentegen vooral associatieve relaties vinden. Woorden die associatief gerelateerd zijn zullen vaak in dezelfde (deel)zin worden gebruikt en daarom zal de afstand tussen deze woorden zelden groot zijn. Bijvoorbeeld ‘Hij stak de sleutel in het slot’ en ‘de sleutel paste in het slot’ of ‘ik las in een boek’ en ‘hij las voor uit een boek’.
Welke van deze relaties het meest vruchtbaar is om op te nemen in een IZS is niet geheel duidelijk en afhankelijk van de vorm van de overige onderdelen van het IZS. Waarschijnlijk zal het IZS beter presteren als het informatie bevat over beide soorten relaties.
Meer in het algemeen kunnen we stellen dat de methode van kennisrepresentatie en -extractie afhankelijk is van de volgende onderdelen:

- **De representaties van de documenten**. Wanneer bijvoorbeeld slechts de index beschikbaar is als representatie van een document dan zal dit andere eisen stellen aan de methode van kennisextractie en daarmee de bijbehorende kennisrepresentatie.
- **De query-representatie**. De manier waarop de interactie met de gebruiker plaats moet vinden is ook bepalend door de keuze van kennis-representatie. Een document-cluster-afbeelding is voor zover bekend alleen te maken op basis van een soort woord-cluster-afbeelding of iets dergelijks, maar in ieder geval niet op basis van een CSN.

5.3.4 Relaties tussen concepten en documenten

Nadat de domeinkennis is onttrokken uit de documenten zal een relatie tussen delen van deze kennis en de documenten moeten worden bepaald. In andere woorden: De documenten moeten worden gekarakteriseerd op basis van de uit deze documenten opgebouwde domeinkennis. Dit kan door de woorden waarmee de concepten worden geïdentificeerd op te zoeken in de documenten, zoals de Aqua-browser dit doet wanneer deze gekoppeld is aan een Bools IZS.
Bij het WEBSOM-algoritme zien we echter een andere mogelijkheid. Hierbij worden de documenten niet verbonden aan de aparte concepten maar wordt een vector gemaakt met het **patroon van concepten** dat in het desbetreffend document voorkomt. Dit wordt mogelijk gemaakt door het feit dat de termen in clusters zijn geplaatst in de woord-cluster-afbeelding, en deze methode is dus niet direct toe te passen op het CSN van de Aqua-browser.
Het belangrijkste hier is echter dat manier van koppelen van concepten aan documenten van de volgende aspecten afhankelijk is:

- **Kennisrepresentatie**: De manier waarop kennis gerepresenteerd is in de kennisbank.
- In mindere mate de **representaties van de documenten**. Afhankelijk van welk deel van de documenten is opgeslagen is het mogelijk deze documenten te koppelen aan de concepten. Een index geeft bijvoorbeeld andere informatie qua context over de termen dan een samenvatting.

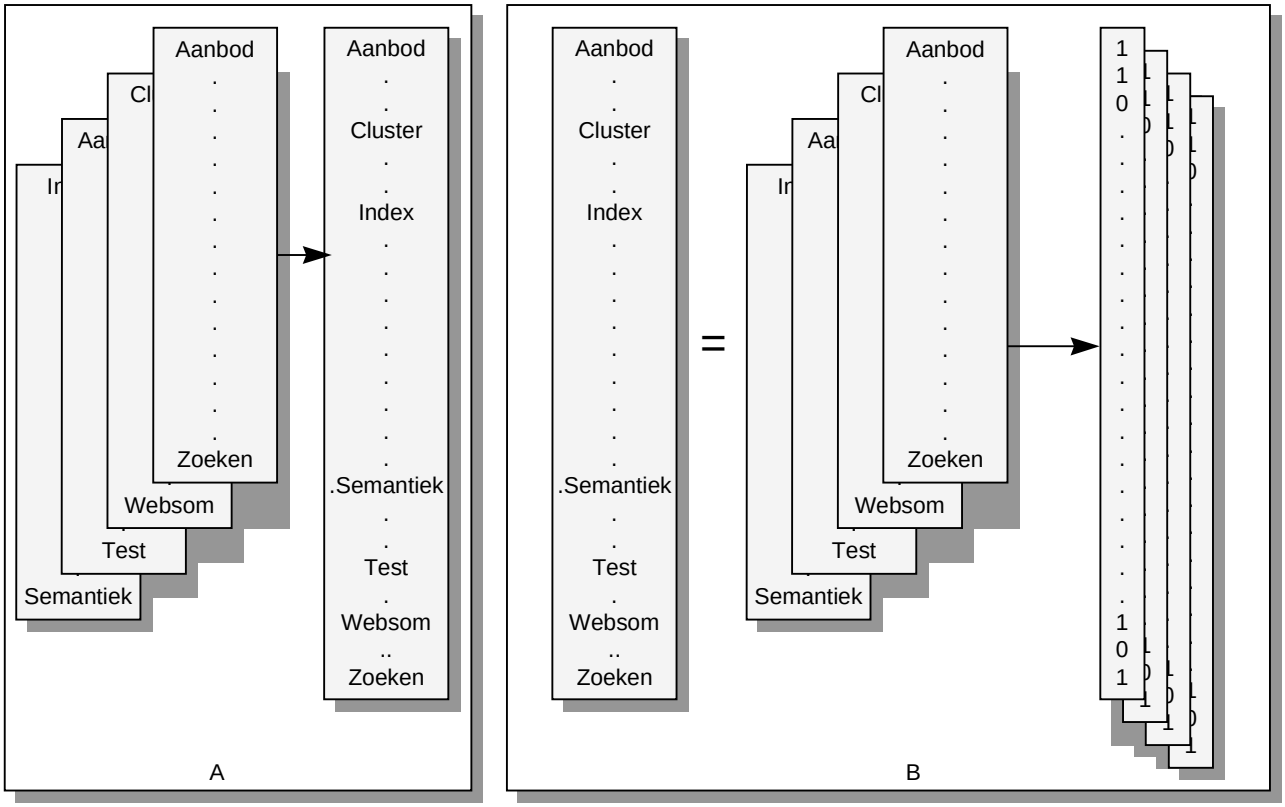
5.3.5 Query representatie

Gebruikers van een IZS weten niet altijd precies wat zij zoeken. Als een IZS over domeinkennis beschikt op het gebied van de documenten is het echter mogelijk geworden de gebruiker te helpen bij een vage gebruikersbehoefte. De zoekactie van de gebruiker is dan niet langer éénrichtingsverkeer: Een (initiële) vraag van de gebruiker zal leiden tot een vernauwing van de context, oftewel een deel- of aspectgebied van het kennisdomein, waarbinnen de concepten nu worden gezocht. Dit deelgebied kan inzichtelijk worden gemaakt voor de gebruiker die vervolgens binnen dit gebied een vervolgvraag kan definiëren of, indien het deelgebied nog maar gerelateerd is aan een beperkt aantal documenten , deze documenten bekijken.
Een triviaal lijkende beperking aan de gebruikersinterface is dat deze in het algemeen moet worden afgebeeld op een tweedimensionaal scherm. Dit betekent dat het te representeren kennisdomein ten alle tijden moet worden gereduceerd tot twee dimensies, wat een beperking legt op de complexiteit en hoeveelheid van de relaties die kunnen worden getoond. Ook moet rekening worden gehouden met de beperkingen van het menselijk brein. Een overvloed aan gegevens op één beeld leidt tot vermindering van de informatieve waarde van deze gegevens.
Over het algemeen is keuze van de query-representatie vrij, met als enige beperking:

- De keuze moet aansluiten bij de gekozen **vorm van kennisrepresentatie**.

5.4 Het INDEXSOM-algoritme

De in dit hoofdstuk tot nu toe genoemde informatie-zoeksystemen hebben beiden als beperking dat zij slechts toepasbaar zijn op collecties van kleine documenten aangezien deze algoritmen de complete tekst gebruiken. Tevens is niet altijd de volledige tekst in elektronisch formaat beschikbaar. Het ligt voor de hand een methode te ontwikkelen die niet de complete tekst gebruikt, maar slechts een deel of een afgeleide ervan. Zo zou bijvoorbeeld de samenvatting of de inhoudsopgave van een document kunnen worden gebruikt. Inhoudsopgave hebben echter als beperking dat zij zeer summier zijn. Voor samenvattingen zouden de tot nu toe beschreven methoden echter meteen toepasbaar zijn indien het document is voorzien van een samenvatting.
Veel documenten, en dan met name boeken, zijn in ieder geval voorzien van een index. Deze index is afgeleid van de tekst zelf en bevat dezelfde onderwerpen. Helaas is dit wel afhankelijk van de perceptie van de schrijver, net als bij een samenvatting. Toch zou het gebruiken van een index voor een informatie-zoeksysteem een duidelijke mogelijkheid bieden voor het toepassen van IZS met kennis op collecties van grote documenten.
Om deze reden is het INDEXSOM-algoritme ontwikkeld (Weustink) als eerste stap richting een IZS dat werkt op basis van indexen.
Aangezien een index geen mogelijkheid biedt tot de analyse van de context van een woord komt de woord-cluster-afbeelding te vervallen. In plaats hiervan worden de indexwoorden direct omgezet tot vectoren die als input dienen voor de woord-cluster-afbeelding. Dit gaat als volgt:



Figuur 5.9: Het genereren van inputvectoren gebaseerd op de individuele indexen van boeken. De indextermen van alle boeken worden geplaatst in een hoofdindex (A). Daarna worden alle indexen vergeleken met de hoofdindex met als resultaat de bitvectoren (B). (Weustink, p.33)

In *figuur 5.9* is weergegeven hoe de indexen van alle boeken worden samengevoegd tot één grote index, waarbij B_i de index is van boek i met n termen b_{ij}, oftewel:

$B_i = \{b_{i\,1}, b_{i\,2}, b_{i\,3}, \ldots, b_{i\,n-1}, b_{i\,n}\}$

B_* is dan de totale index van de collectie met m termen b_{*j}:

$B_* = \bigcup_i B_i = \{b_{*\,1}, b_{*\,2}, \ldots, b_{*\,m-1}, b_{*\,m}\}$

Met behulp van deze index wordt nu een bitvector X_i opgesteld voor ieder boek i, waarbij de waarde 1 wordt toegekend indien de indexterm voorkomt in de index van het boek:

$X_i = \{x_{i\,1}, x_{i\,2}, \ldots, x_{i\,m-1}, x_{i\,m}\}$

$b_{*i} \in B_i \rightarrow x_{ij} = 1$
$b_{*i} \notin B_i \rightarrow x_{ij} = 0$

Deze bitvectoren dienen nu als input voor de SOM zoals beschreven in paragraaf 3.3.3. Na training wordt per bitvector de winnende eenheid in het raster bepaald en wordt het document aan deze eenheid toegekend. Zo ontstaat een document-cluster-afbeelding, vergelijkbaar met die van het WEBSOM.

Helaas levert een bewerking van een collectie documenten al snel een groot aantal verschillende indextermen op, bijvoorbeeld m>10.000 bij 50 documenten. Dit levert convergentie-problemen op voor het SOM-algoritme: het algoritme doet veel te lang over het generen van een acceptabele ordening. Om dit op te lossen is het mogelijk alleen die indextermen te gebruiken die ook in de samenvattingen voorkomen of in de inhoudsopgaven. Deze laatste methode kan bijvoorbeeld het aantal indextermen reduceren met 80%. Op deze manier is een document-cluster-afbeelding te creëren. Experimenten hebben echter aangetoond dat deze methode geen semantische ordening vertoont (Weustink, p.40).
De slechte semantische clustering zou aan een aantal factoren toe kunnen worden geschreven:

- De indexen die zijn gebruikt voor het experiment hadden onderling weinig termen gemeen. Hierdoor werd het bijzonder moeilijk voor de SOM om semantische relaties aan te tonen. Bij een analyse van 8 boeken bleek dat deze slechts 1% van de indextermen gemeen hadden.
- Bij het INDEXSOM-algoritme wordt geen rekening gehouden met extra informatie die een index kan verschaffen, zoals de paginanummers waaraan de indextermen gerelateerd zijn.
- Sommige indexen bevatten veel meer termen dan andere. Hierdoor wordt hun aandeel in de bitvector zo groot dat zij de andere documenten overschaduwen en een goede ordening in de weg staan.

Als we het INDEXSOM-algoritme analyseren aan de hand van de algemene architectuur welke in dit hoofdstuk is gevonden blijkt dat nog een factor van belang zou kunnen zijn voor de slechte semantische ordening:

- Het ontbreken van de woord-cluster-afbeelding. Het INDEXSOM-algoritme heeft geen woord-cluster-afbeelding en kan dus niets zeggen over de semantische of associatieve relatie tussen individuele woorden. Ieder woord vormt een apart component van de vector en een onderling verband tussen deze woorden wordt hierdoor niet gerepresenteerd. Indexen hebben maar weinig termen gemeen ook al beschrijven ze ongeveer dezelfde onderwerpen. Dit betekent dat verschillende woorden worden gebruikt om ongeveer dezelfde concepten aan te duiden. Indien deze relatie tussen de woorden wordt meegenomen kunnen de indexen beter met elkaar worden vergeleken.

In hoofdstuk 6 zal een nieuw algoritme aan bod komen waarmee onder andere is geprobeerd rekening te houden met deze factoren.

Hoofdstuk 6: Het ACR-WEBSOM algoritme

*"Unsupervised learning can only do anything useful when there is **redundancy** in the input data. Without redundancy it would be impossible to find any patterns or features in the data, which would necessarily seem like random noise. In this sense **redundancy provides knowledge**." (Hertz e.a., p.197)*

In dit hoofdstuk zal een nieuw algoritme gepresenteerd worden dat is opgebouwd uit een combinatie van architecturen die besproken is in hoofdstuk 5 en dat kan functioneren op basis van een interne representatie van de documenten die slechts bestaat uit een index.
Als eerste zal worden bekeken of het mogelijk is kennis te extraheren uit de index van een boek (6.1), waarna de vorm van kennisrepresentatie wordt besproken die geschikt is voor een dergelijke kennisextractie (6.2). Hierna wordt de precieze methode van kennisextractie uit een index besproken (6.3) en het complete algoritme behandeld (6.4). Als laatste zal de programmacode van het prototype kort worden besproken (6.5).

6.1 Woordrelaties in een index

Eén van de grootste moeilijkheden bij het onttrekken van kennis aan een index is het ogenschijnlijk ontbreken van gegevens waarop de relaties tussen woorden kunnen worden geschat. Indexen zijn op alfabetische volgorde gerangschikt, waardoor ieder semantisch of associatief verband tussen de woorden verloren lijkt. Een voordeel van een index is echter wel dat de woorden die erin voorkomen door de schrijver als relevant zijn geacht. Zij hebben dus altijd een informatieve waarde, veronderstellend dat de auteur zijn werk goed heeft gedaan. Er zal bijvoorbeeld nooit in een index een verwijzing staan naar woorden met een lage informatieve waarde zoals 'uiteraard', 'doch' of 'dus'. Dit brengt wel weer als nadeel mee dat we hiermee afhankelijk zijn van de subjectieve perceptie van de schrijver.
Eén manier om de achter de relaties tussen de woorden te komen is door te kijken naar de paginanummers waar de woorden aan gerelateerd zijn. Als we veronderstellen dat op één pagina slechts één onderwerp wordt behandeld dan kunnen we zeggen dat de woorden die verwijzingen hebben naar dezelfde pagina gerelateerd zijn aan hetzelfde onderwerp en dus aan elkaar zijn gerelateerd. Als we verder de veronderstelling maken dat het onderwerp van één pagina niet veel verschilt van het onderwerp van de daarop volgende pagina dan kunnen we ook stellen dat woorden die op nabije pagina's voorkomen enigszins gerelateerd zijn.
De index van een boek is in feite een geïnverteerde index van de tekst: van ieder (belangrijk) woord is weergegeven op welke bladzijde deze staat. Met behulp van de paginanummers is het enigszins mogelijk deze inversie ongedaan te maken. We kunnen de geïnverteerde index inverteren. We de definiëren B_i als de verzameling van n indextermen b_{ij} uit boek i of, meer formeel:

$B_i = \{b_{i\,1}, b_{i\,2}, b_{i\,3}, \ldots , b_{i\,n-1}, b_{i\,n}\}$

Bij iedere term b_{ij} hoort een verzameling van paginaverwijzingen P_{ij} welke bestaat uit één of meer paginaverwijzingen p_{ijk}. p_{ijk} is een bladzijdenummer. (i verwijst naar het boek, j naar de term en k naar de paginaverwijzing bij de term.)

S_i = de bewerkte index van B_i met m deelverzamelingen S_{it}, waarbij m gelijk is aan het aantal pagina's in het boek, i verwijst naar het boek en t naar de pagina.

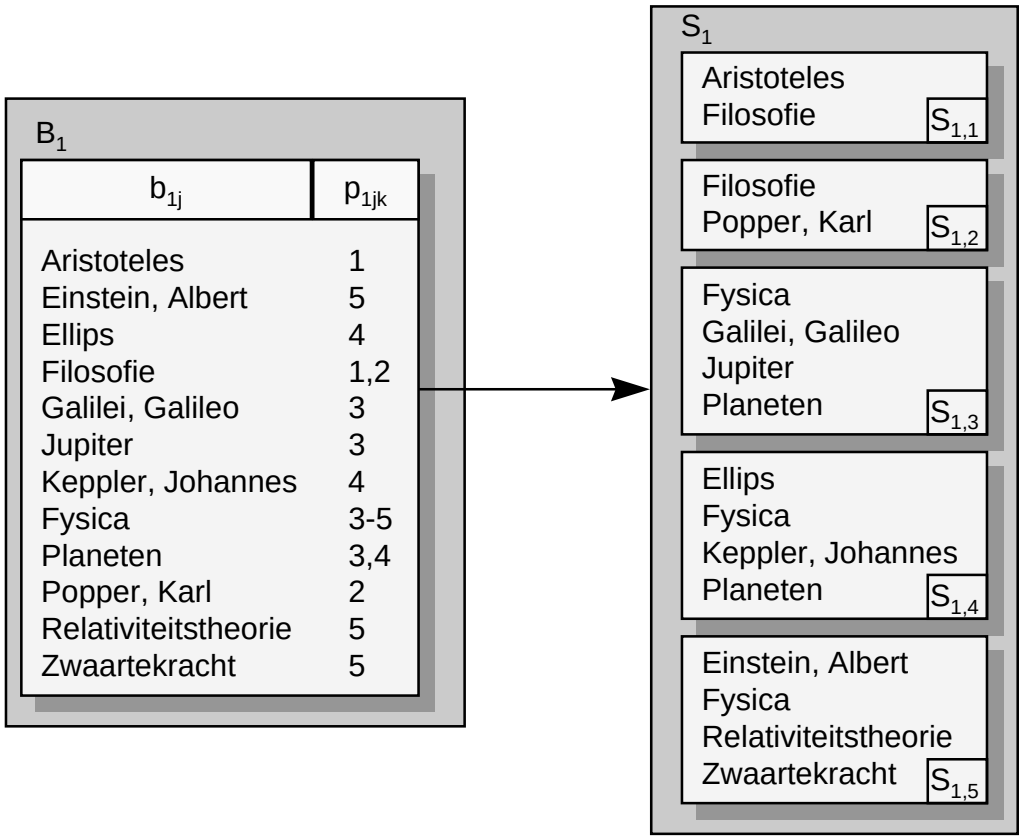
$S_i = \{S_{i1}, S_{i2}, S_{i3}, \ldots , S_{i\,m-1}, S_{im}\}$

De elementen van de deelverzamelingen S_{it} zijn indextermen b_{ij}. De bewerking van de index verloopt nu als volgt:

$\exists\ k, P_{ijk} = t \rightarrow b_{ij} \in S_{it}$

Oftewel, een woord b_{ij} wordt toegekend aan deelverzameling S_{it}, dus aan bladzijde t van boek i, als dit woord een verwijzing p_{ijk} heeft naar deze bladzijde t.

In *figuur 6.1* is een voorbeeld weergegeven. Het betreft hier de bewerking van de index van 'boek 1'.



Figuur 6.1: Voorbeeld van de inversie van een index. Aan de hand van de paginanummers wordt per woord bepaald op welke bladzijde(n) het woord moet voorkomen.

Als we kijken naar de eigenschappen van de bewerkte index is duidelijk dat iedere zinsstructuur afwezig is. Alleen de kernwoorden van de tekst zijn overgebleven en op een pagina S_{it} is de volgorde waarin de woorden voorkomen arbitrair (in dit voorbeeld op alfabetische volgorde). Het is derhalve in deze bewerkte index niet mogelijk de relaties te bepalen aan de hand van de korte context zoals gebruikelijk bij de WEBSOM-methode aangezien niet duidelijk is hoe de woorden in een zin gebruikt worden. De omliggende woorden in de werkelijk tekst zijn niet via de index te achterhalen en hierdoor verliest de korte-context-analyse zijn geldigheid.
De woord-nabijheid-analyse zoals beschreven in paragraaf 5.2.1 kan daarentegen wel gebruikt worden bij deze bewerkte index. Zoals gezegd zullen woorden die voorkomen op dezelfde of nabije pagina's veelal betrekking hebben op hetzelfde onderwerp. In een CSN kunnen de connecties tussen de woorden die in de bewerkte index dicht bij elkaar liggen worden versterkt en op deze manier kan kennis worden onttrokken aan de index.

6.2 Vorm van de kennisrepresentatie

Zoals in de vorige paragraaf is beschreven kan met behulp van de index van een boek een CSN worden opgebouwd. Zo'n CSN zou als basis kunnen dienen voor de Aqua-browser, zodat een knowledge-based IZS ontstaat waarvoor de index van boeken voldoende informatie verschaft.
Zoals in paragraaf 5.3.4 staat vermeld onderscheiden het WEBSOM-algoritme en de Aqua-browser zich onder andere op de manier waarop de concepten zijn gekoppeld aan de documenten. Bij de Aqua-browser worden concepten direct aan de woorden gekoppeld in een document, bij de WEBSOM wordt gekeken naar het patroon van concepten dat in het document voorkomt.
Boeken zijn veelal omvangrijke documenten die niet beperkt zijn tot één enkel onderwerp of concept. We kunnen stellen dat zij een patroon van onderwerpen of concepten bevatten. Gegeven het feit dat het WEBSOM-algoritme documenten karakteriseert als patronen van concepten poneren we de hypothese dat dit algoritme beter in staat is de juiste boeken te vinden bij een gegeven query. Onderzoek naar de mogelijkheid om de Aqua-browser te koppelen aan een CSN dat is opgebouwd vanuit de indexen van gaat echter te ver voor deze scriptie.
Gegeven dat kennisextractie uit een index via de woord-nabijheid-analyse van het CSN moet geschieden en dat de document-karakterisering van de WEBSOM waarschijnlijk beter voor boeken geschikt is lijkt een combinatie van deze twee elementen gewenst. Hierin schuilt echter een probleem: **het WEBSOM-algoritme vereist vectoren als input**. Een verzameling vectoren definieert een ruimte dus de domeinkennis moet de vorm hebben van een conceptuele ruimte en niet van een netwerk zoals het CSN. Het is echter niet mogelijk om de kennis in een CSN te vertalen naar zo'n conceptuele ruimte.

6.2.1 Associatieve Conceptuele Ruimte

Een oplossing voor het probleem zoals hiervoor geschetst is het opzetten van een conceptuele ruimte op basis van de leerregels die gelden in een CSN. Deze vorm van kennisrepresentatie zullen we in het vervolg Associatieve Conceptuele Ruimte (ACR) noemen omdat dit een conceptuele ruimte is waarbij de weergegeven relaties tussen de woorden hoofdzakelijk associatief zullen zijn aangezien zij worden gevormd op basis van de woord-nabijheid-analyse. In deze ACR geldt, indien we uitgaan van een Euclidische ruimte:
Er is een verzameling van n woorden $W = \{w_1, w_2, w_3, \ldots, w_{n-1}, w_n\}$. Deze woorden zijn één-op-één gerelateerd aan de referentievectoren in verzameling X, $X = \{x_1, x_2, x_3, \ldots, x_{n-1}, x_n\}$. Deze één-op-één relatie vloeit voort uit de vereenvoudiging zoals gemaakt in paragraaf 3.3.1. waar is gesteld dat één woord slechts één concept identificeert. De referentievectoren zijn opgebouwd uit d componenten, afhankelijk van het aantal dimensies d van de conceptuele ruimte. $x_i = \{x_{i1}, x_{i2}, x_{i3}, \ldots, x_{i\,d-1}, x_{id}\}$. Component x_{ij} is een reëel getal.
De relatie tussen concepten kan nu echter niet meer worden weergegeven door verbindingen zoals in een CSN. De relatie tussen concepten dient dan ook op een andere manier te worden weergegeven:

De afstand tussen referentievectoren geeft de mate van associatie aan tussen de concepten die behoren bij deze vectoren

In andere woorden: de sterkte van de verbinding tussen twee concepten is gelijk aan de afstand tussen de referentievectoren van deze concepten:

associatie tussen w_i en w_j = $|| x_i - x_j ||$ ($|| . ||$ geeft de Euclidiaanse afstand)

De vraag is nu hoe de correcte associaties tot stand moeten komen in deze conceptuele ruimte. Hiervoor moet het netwerk leren:

6.2.2 Het leermechanisme

Als we de leerregel van Hebb (zie paragraaf 2.2.2) toepassen betekent dit dat als twee concepten tegelijk geactiveerd worden de verbinding tussen deze twee moet worden versterkt. De implementatie van dit activeren zal later worden besproken in paragraaf 6.4. Het versterken van de verbinding houdt in dat de afstand tussen de referentievectoren van deze woorden moet worden verkleind:

De referentievectoren van concepten die tegelijkertijd worden geactiveerd worden dichter naar elkaar gebracht.

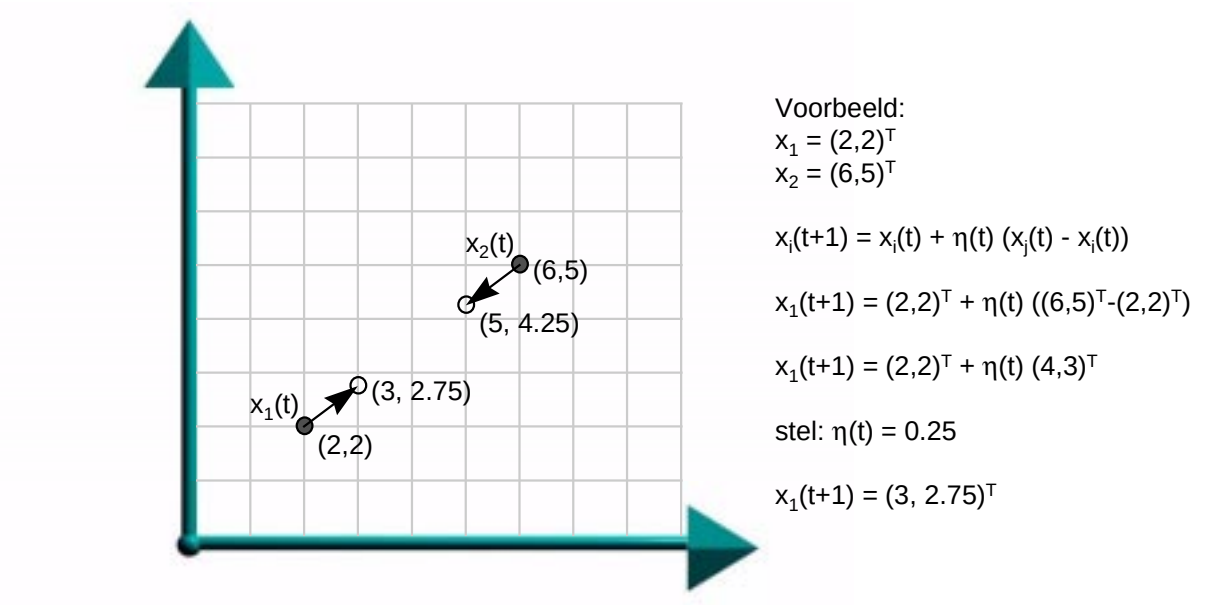
w_i, w_j actief $\rightarrow$ $|| x_i(t+1) - x_j(t+1) || < || x_i(t) - x_j(t) ||$

Dit kan worden bereikt door de afstand met een fractie te verkleinen :

$x_i(t+1) = x_i(t) + \eta(t) (x_j(t) - x_i(t))$ (associatieregel)

$x_j(t+1) = x_j(t) - \eta(t) (x_j(t) - x_i(t))$

Hierin is $\eta(t)$ de learning-rate welke de grootte van de fractie bepaalt waarmee de afstand wordt verkleind. In *figuur 6.2* is een rekenvoorbeeld gegeven van deze regel. In dit voorbeeld is berekend hoe x_1 wordt aangepast indien w_1 en w_2 tegelijk actief zijn.



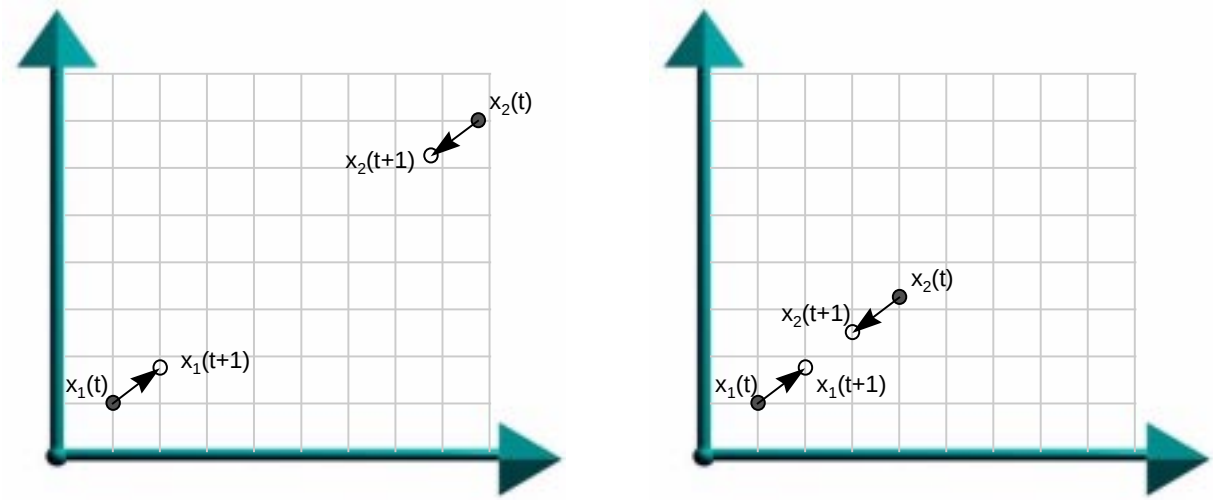
Figuur 6.2: Voorbeeld van de associatieregel.

Bij het toepassen van deze associatieregel zal vaak al snel de absolute afstand tussen alle vectoren zeer klein worden. Bij een goede kennisrepresentatie zal echter niet alleen de afstand tussen gerelateerde woorden klein moeten zijn; ook zal de afstand tussen niet-gerelateerde woorden groter moeten zijn. We moeten dan ook kijken naar de **relatieve afstand**. Het toepassen van de associatieregel zal de afstand tussen twee vectoren verkleinen, maar daarmee worden alle andere afstanden relatief groter ten opzichte van deze afstand.
We zien dat door toepassing van de associatieregel vectoren die ver van elkaar verwijderd zijn met grote stappen naar elkaar toe worden gebracht en vectoren die dicht bij elkaar liggen een minimale aanpassing krijgen. Dit kan echter tot gevolg hebben dat één enkele activering van een ver verwijderde vector een lokaal verband dat door vele herhalingen is ontstaan compleet kan doen verdwijnen. Iedere associatie zal in beginsel gelijk moeten worden behandeld en zal dus een even groot effect moeten hebben. We kunnen de aanpassingsvector normaliseren zodat iedere aanpassing op dezelfde schaal plaatsvindt:

$$x_i(t+1) = x_i(t) + \eta(t) \frac{x_j(t) - x_i(t)}{\| x_j(t) - x_i(t) \|}$$ (genormaliseerde associatieregel)

$$x_j(t+1) = x_j(t) - \eta(t) \frac{x_j(t) - x_i(t)}{\| x_j(t) - x_i(t) \|}$$

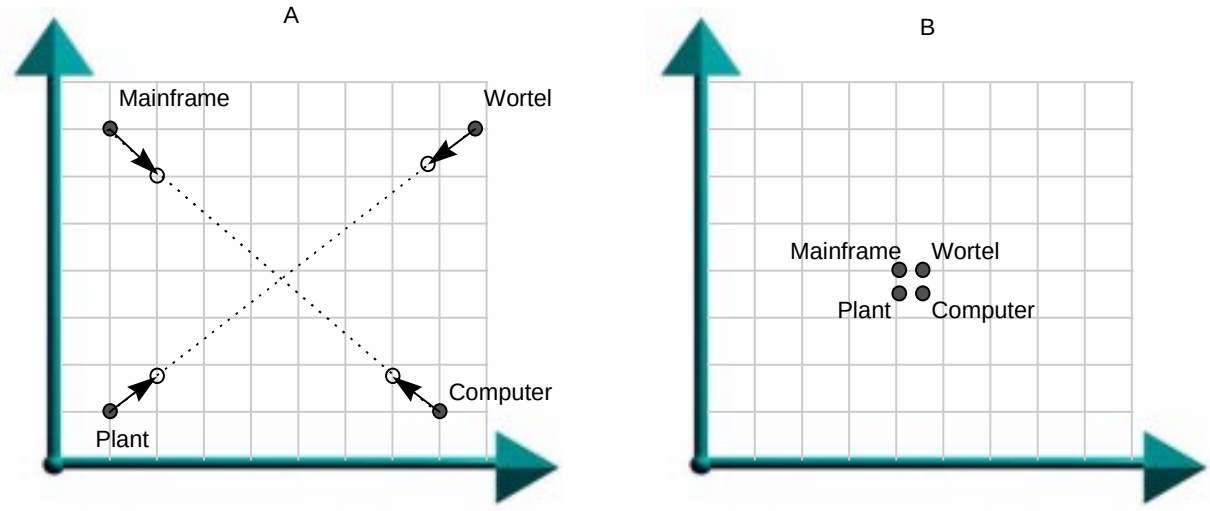
Het verschil tussen de twee vectoren wordt genormaliseerd en vermenigvuldigd met de learning-rate $\eta(t)$. Vervolgens wordt deze nieuwe vector opgeteld bij de oorspronkelijke vector.



Figuur 6.3: Voorbeeld van de genormaliseerde associatieregel: Zowel associatie over een grote afstand (links) als over een kleine afstand (rechts) leidt tot een gelijke aanpassing.

6.2.3 Actief vergeten

In de vorige paragraaf is gesteld dat de (relatieve) afstand bepalend is voor de associatie tussen twee woorden. In *figuur 6.4* is echter een voorbeeld gegeven waaruit blijkt dat de hierboven beschreven methode van leren tot verkeerde ordeningen kan leiden en daarmee tot verkeerde associaties. In dit voorbeeld wordt begonnen met 4 woorden waarvan de referentievectoren gelijk zijn verdeeld over de conceptuele ruimte (*figuur 6.4a*) . De woorden ‘Plant’ en ‘Wortel’ worden tegelijk geactiveerd en worden naar elkaar toegebracht. Ook de woorden ‘Computer’ en ‘Mainframe’ worden aan elkaar geassocieerd. Na verloop van tijd blijkt dat het algoritme inderdaad de associaties heeft geleerd (*figuur 6.4b*). De referentievectoren van de geassocieerde woorden liggen dicht bij elkaar: ‘Plant’ bij ‘Wortel’ en ‘Computer’ bij ‘Mainframe’. We zien echter dat door puur toeval er een schijnassociatie is ontstaan tussen deze twee verder niet gerelateerde woordenparen.



Figuur 6.4: Voorbeeld van een foute associatie a.g.v. de (genormaliseerde) associatieregel. Links: voor de aanpassing. Rechts: na meerdere malen de associatieregel te hebben toegepast

Deze situatie is niet wenselijk aangezien deze geen informatie verschaft. Er moet dus ook een manier zijn om de afstand toe te laten nemen. Voor een CSN gebeurt dit via een Ricatti-Type leerwet (Kohonen, p67-70). De verbindingen worden afgezwakt middels een vorm van **actief vergeten**. Dit actieve vergeten kunnen we ook introduceren in het model van Associatieve Conceptuele Ruimte. Actief vergeten betekent dat alle verbindingen afgezwakt worden, oftewel dat de afstanden tussen alle referentievectoren onderling vergroot dienen te worden:

$\forall x_i, x_j \in X$, $|| x_i(t+1) - x_j(t+1) || > || x_i(t) - x_j(t) ||$

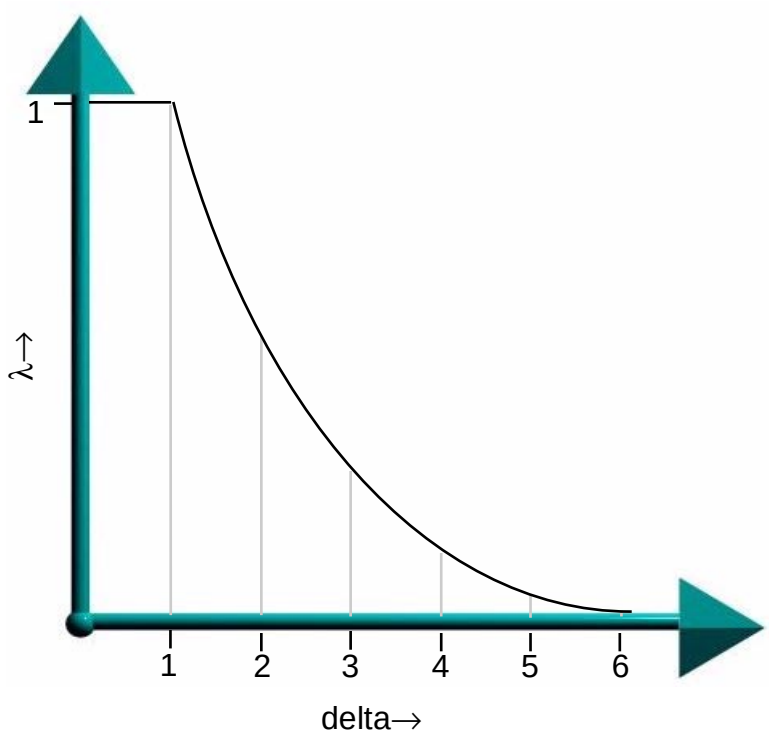
We kunnen zeggen dat de referentievectoren een onderlinge afstootkracht moeten hebben. Net als voor de associatieregel geldt ook hier dat de mate waarin de vectoren worden verplaatst moet worden genormaliseerd om ervoor te zorgen dat op grotere afstanden de aanpassing geen extreme gevolgen heeft waardoor een ontstane ordening verloren gaat. Dit kan op deze manier bereikt worden:

$$x_i(t+1) = x_i(t) - \lambda(delta)\ \frac{x_j(t) - x_i(t)}{delta} \qquad \text{(vergeetregel)}$$

$$delta = || x_j(t) - x_i(t) ||$$

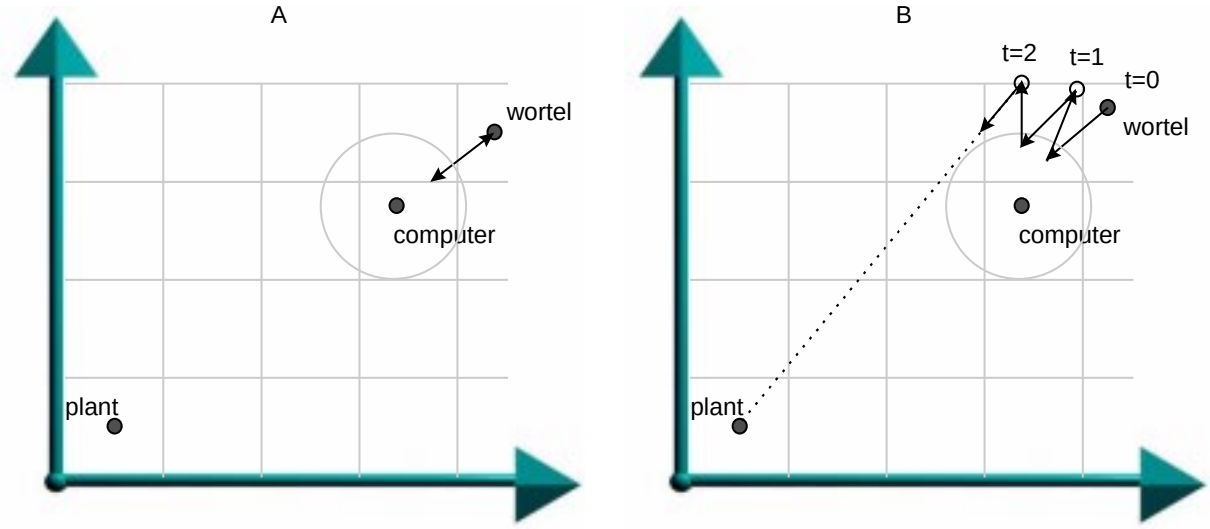
$\lambda(delta)$ is de afstootfunctie en is afhankelijk van de onderlinge afstand. Indien de conceptuele ruimte complexe kennisstructuren bevat is het wenselijk dat het effect van de vergeetregel lokaal is. Dit houdt in dat de afstootkracht dient af te nemen naarmate de afstand tussen de vectoren groter wordt. Dit kan worden bereikt door deze kracht omgekeerd evenredig te stellen aan de afstand: $\lambda(delta) = 1/delta$. Indien nu twee vectoren echter dicht bij elkaar genaderd zijn ($delta \rightarrow 0$) zal de afstootfunctie echter leiden tot extreme afstoting met $\lambda(delta) \rightarrow \infty$. Hypothetisch lijkt het beter om binnen een bepaald bereik de afstoting uniform te stellen:

$$\lambda(\text{delta}) = \begin{cases} 1 & \text{voor delta} < 1 \\ 1/\text{delta} & \text{voor delta} \geq 1 \end{cases} \qquad \text{(afstootfunctie)}$$



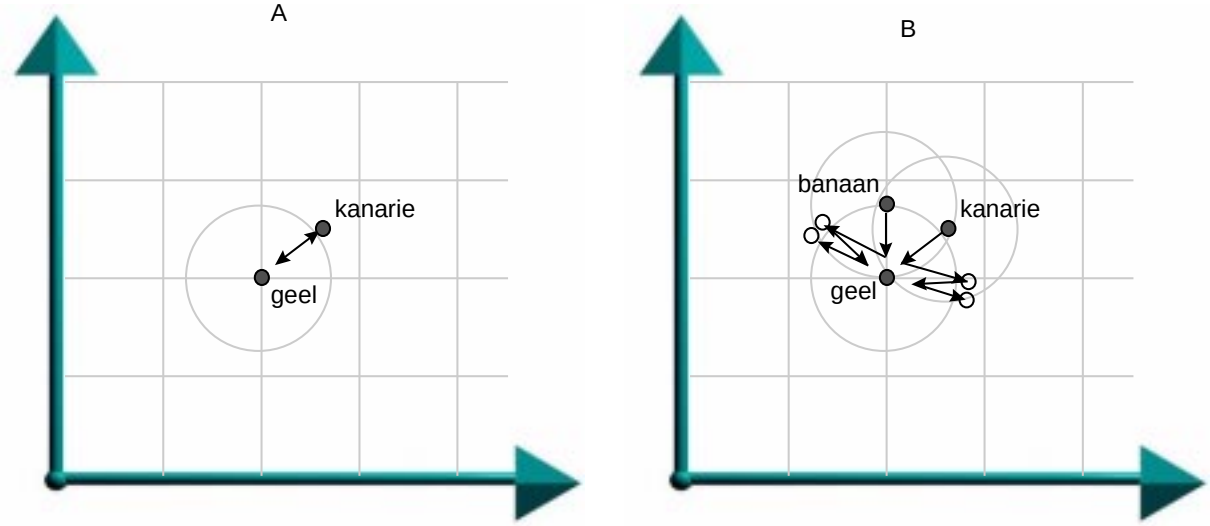
Figuur 6.5: De afstootfunctie

Door toepassing van de vergeetfunctie kan zowel een schijnassociatie als een aangeleerde associatie teniet worden gedaan. De kans bestaat zelfs dat de referentie-vectoren na toepassing van zowel de associatieregel en de vergeetregel op dezelfde positie terugkeren. In *figuur 6.6a* is zo'n 'fixed-point' sterk vereenvoudigd weergegeven zonder de aanpassingen op vectoren te tonen anders dan die van het woord 'wortel'. Na toepassing van het algoritme keert de referentievector terug naar zijn oorspronkelijke positie en is het woord 'wortel' nog steeds gerelateerd aan 'computer'. Indien het systeem in een dergelijke fixed-point is gevangen zal er nooit een verandering op treden. We zien echter dat deze fixed-point een onstabiele situatie weergeeft indien een woord wordt afgestoten door een ander woord dan waar het door wordt aangetrokken. Een minuscule wijziging in de startpositie van de vector zorgt ervoor dat de vector na enige iteraties van het algoritme vrij is van de niet gerelateerde vector. Schijnassociaties kunnen dus nog steeds optreden in een algoritme met vergeetfunctie. Ze zijn echter structureel instabiel en bij toenemende complexiteit met een groter aantal vectoren zal het niet lang duren voordat zo'n minuscule wijziging in een vector zich voordoet als gevolg van een interactie met andere vectoren.



Figuur 6.6: Voorbeeld van de werking van de vergeetfunctie. De cirkel om 'computer' geeft aan dat alleen de afstotende werking van deze vector in dit voorbeeld wordt beschouwd. In A leidt de toepassing van de vergeetregel tot het tenietdoen van het effect van associatieregel. In B convergeert het systeem naar een betere oplossing.

In tegenstelling tot schijnassociatie zal een gedegen associatie wel aanleiding geven tot een stabiele situatie. Voor twee woorden die werkelijk zijn geassocieerd zijn een reeks fixed-points te vinden die wel stabiel zijn. In *figuur 6.7a* is zo'n punt weergegeven. Een klein wijziging in de startpositie zal al snel convergeren naar hetzelfde of een nabij gelegen fixed-point.
In *figuur 6.7b* zien we twee woorden 'banaan' en 'kanarie' die onderling niet geassocieerd zijn maar wel beide aan een ander woord 'geel' zijn gerelateerd. In dit geval zal na enkele iteraties van het algoritme een situatie ontstaan waarin deze niet gerelateerde woorden zover mogelijk van elkaar verwijderd zijn terwijl zij hun associatie met het woord 'geel' blijven behouden.

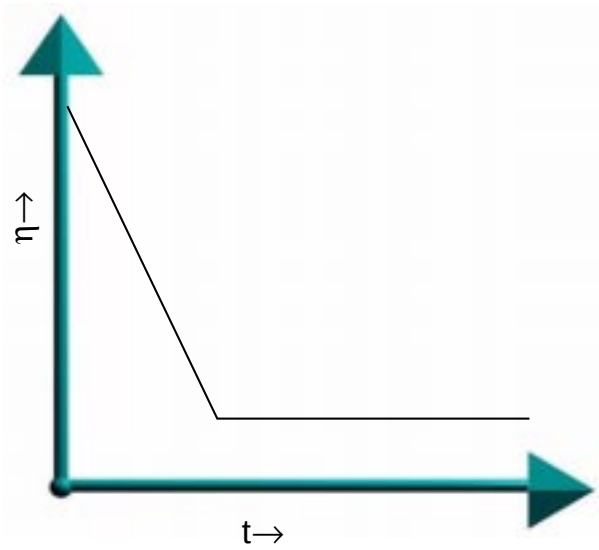


Figuur 6.7: Voorbeeld van ordening bij gerelateerde woorden. 'banaan' en 'kanarie' zijn onderling niet gerelateerd maar beide zijn ze wel gerelateerd aan 'geel'.

Op deze manier kunnen complexe kennisstructuren ontstaan. Het aantal onderling niet gerelateerde woorden die aan een ander woord kunnen worden geassocieerd zonder dat er een onderlinge schijnassociatie ontstaat is afhankelijk van het aantal dimensies van de conceptuele ruimte. Hiernaast geeft een hogere dimensionaliteit een betere kans dat een referentievector om een niet gerelateerde vector kan worden geleid zoals in *figuur 6.6b*. Aan de andere kant moet de dimensionaliteit zo laag mogelijk worden gehouden om de tijdscomplexiteit van het algoritme te beperken.

6.2.4 Het algoritme in de tijd

De conceptuele ruimte zal een beginsituatie moet hebben. Hiervoor kunnen de referentievectoren willekeurige waarden toegewezen krijgen. Door toepassing van het algoritme zal hopelijk dan vanuit bijna iedere beginpositie uiteindelijk een soortgelijke ordening ontstaan. Deze hypothese zal worden gestaafd in paragraaf 7.3.2. In de toekomst zal eventueel moeten worden onderzocht of er een mogelijkheid bestaat een andere beginsituatie te genereren van waaruit het algoritme sneller tot een optimale situatie kan komen.
Om de rekentijd te beperken is het in ieder geval wenselijk vanuit de willekeurige beginpositie snel naar een situatie te gaan waar al enige ordening heeft plaatsgevonden. Vooral de cirkelbeweging om niet-gerelateerde referentievectoren, zoals beschreven in de vorige paragraaf, vergt veel iteraties. In paragraaf 7.3.3 is beschreven hoe experimenten hebben aangetoond dat zich sneller een ordening voordoet wanneer de aanpassing van de vectoren in het begin zo groot is dat deze de vergeetfunctie overschaduwt qua schaal.. De learning-rate $\eta(t)$ zal dus in het begin zeer groot moeten zijn en moet later afnemen naar een waarde die in verhouding staat tot de vergeetfunctie.
Er moet wel een ondergrens worden gesteld aan de learning-rate aangezien een te kleine learning-rate als gevolg zal hebben dat de vergeetfunctie alle associaties teniet doet. De functie $\eta(t)$ zal dan over de tijd als volgt verlopen:



Figuur 6.8: De learning-rate over de tijd

6.3 Associatieve conceptuele ruimte van een index

In paragraaf 6.1 is behandeld hoe een index van een boek omgezet kan worden naar een verzameling $S_i = \{S_{i1}, S_{i2}, S_{i3}, \ldots , S_{i\,m-1}, S_{im}\}$ waarbij i naar het boek verwijst met m pagina's. Iedere S_{ij} bevat een verzameling van woorden die volgens de index op pagina j voorkomen. Indien alle verzamelingen S_{ij} achter elkaar worden geplaatst en binnen iedere verzameling de woorden in een arbitraire, bijvoorbeeld alfabetische, volgorde worden gezet, ontstaat één lange keten van woorden T_i ,$T_i = \{t_{i1}$,$t_{i2}, \ldots ,t_{i\,k-1}$,$t_{ik}\}$ waarbij i verwijst naar het boek. Alle ketens T_i kunnen voor alle boeken in willekeurige volgorde achter elkaar worden gezet. Dit resulteert in één grote keten $T_{*,}T_* = \{T_1, T_2, \ldots , T_{l-1}$,$T_l\} = \{t_{1\,1}$,$t_{1\,2}$,$t_{1\,3}$, ... ,$t_{l\,k-1}$,$t_{l\,k}\}$
Tevens kan de vocabulaire van deze woordenketen worden samengesteld uit de indexen worden vastgesteld aan de hand van de afzonderlijke indexen. We definiëren B_i als de index van boek i met n indextermen b_{ij} of, meer formeel:

$B_i = \{b_{i\,1}, b_{i\,2}, b_{i\,3}, \ldots , b_{i\,n-1}, b_{i\,n}\}$

We definiëren verder B_* als de verzameling van alle indextermen van de collectie met p termen b_{*j}

$B_* = \bigcup_i B_i = \{b_{*\,1}, b_{*\,2}, \ldots , b_{*\,p-1}, b_{*\,p}\}$

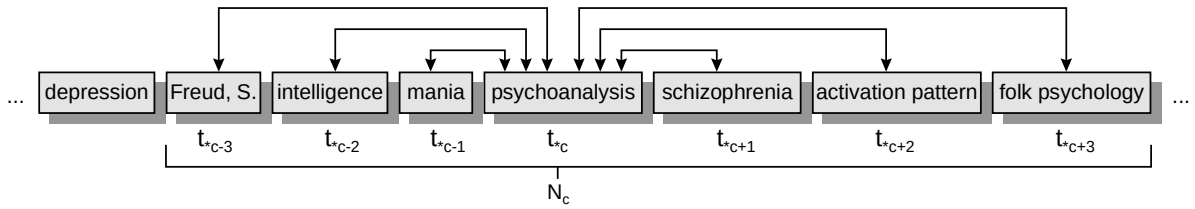
We kunnen nu voor ieder woord uit deze vocabulaire B_* in het Associatieve Conceptuele Ruimte-algoritme (ACR-algoritme) een referentievector definiëren zoals beschreven in paragraaf 6.2.1, oftewel $W = B_*$. De associatie tussen woorden kan nu worden gevonden door te bepalen welke van de woorden uit B_* vaak in elkaars buurt voorkomen in T_*. Dit kan bereikt worden door de keten T_* woord voor woord door te lopen en voor ieder woord t_{*c} dat woord en alle woorden die in de buurt van dat woord liggen te activeren. We kunnen zeggen dat de te activeren woorden deel uit moeten maken van de buurt van t_{*c}, oftewel :

$t_{*i} \in N_c$

waarin Nc de buurtverzameling is van t_{*c}. Een logische keuze voor N_c is n woorden voor en n woorden na t_{*c}, ook wel een (woord)window genoemd:

$N_c = \{t_{*c-n}$, ... ,$t_{*c-2}, t_{*c-1}, t_{*c+1}, t_{*c+2}, \ldots ,t_{*c+n}\}$

Ieder woord in deze verzameling wordt apart verbonden met t_{*c} zodat 2n paren van woorden ontstaan die tegelijkertijd worden geactiveerd zodat de twee corresponderende referentievectoren kunnen worden aangepast volgens de associatieregel van het ACR-algoritme. In *figuur 6.9* is een voorbeeld gegeven van een buurtfunctie met n=3 en de paren die worden gevormd als input voor het ACR-algoritme.



Figuur 6.9: Associatie binnen de buurtverzameling met n=3.(Voorbeeld uit de index van Churchland)

Hierbij kan rekening worden gehouden met de frequentie waarin de woorden voorkomen. Woorden die vaak voorkomen hebben minder informatieve waarde dan woorden die minder vaak voorkomen. De associatieregel zal er dan als volgt uitzien:

$$x_i(t+1) = x_i(t) + \eta(t) \frac{x_j(t) - x_i(t)}{\| x_j(t) - x_i(t) \|} \cdot \alpha(i,j)$$

$$x_j(t+1) = x_j(t) - \eta(t) \frac{x_j(t) - x_i(t)}{\| x_j(t) - x_i(t) \|} \cdot \alpha(i,j)$$

$$\alpha(i,j) = \frac{2.freq_*}{freq_i + freq_j}$$

$freq_i$ = frequentie van woord i

$$freq_* = \frac{|T_*|}{|B_*|}$$

| . | geeft het aantal elementen in de verzameling

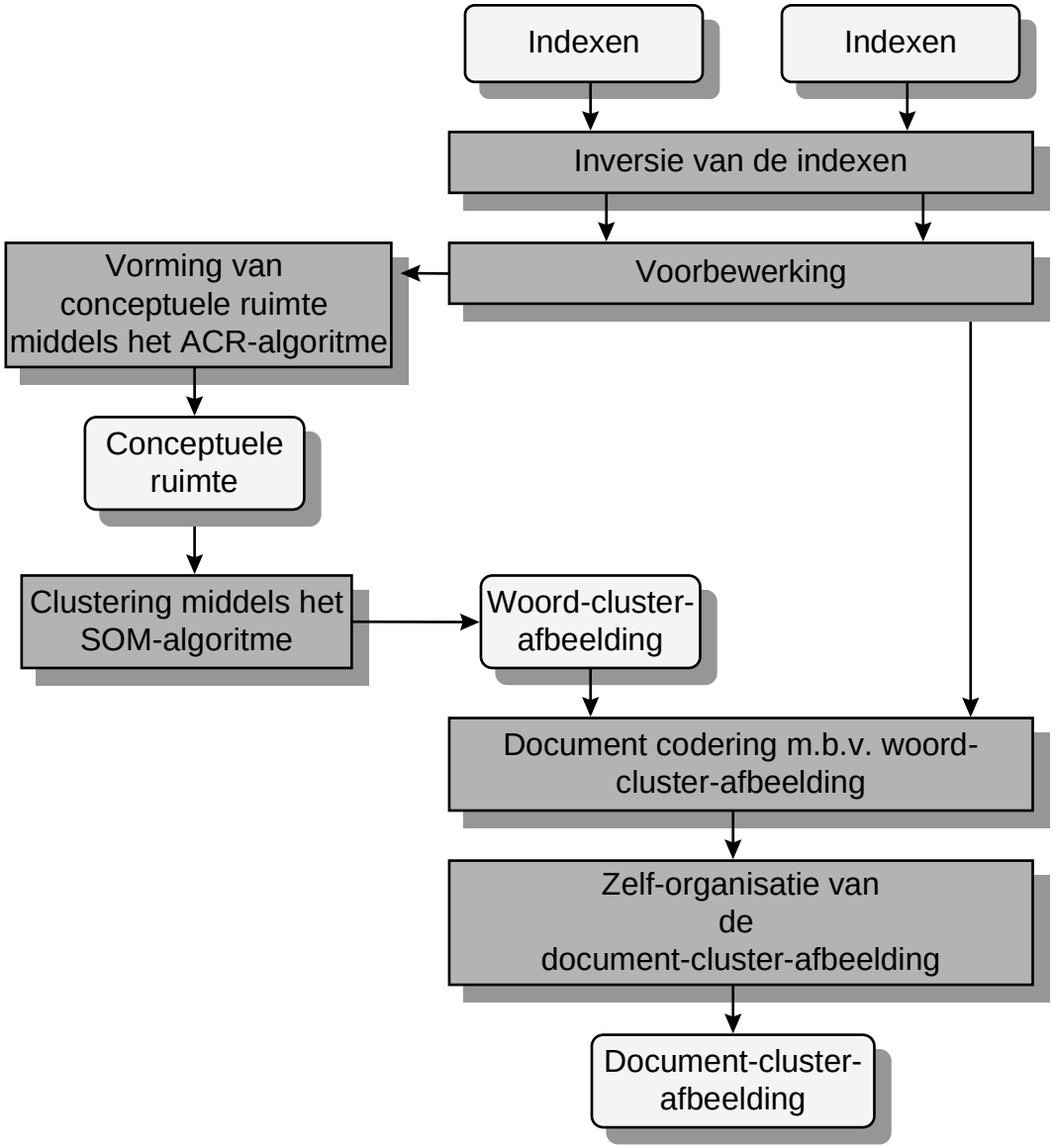
$freq_*$ is de gemiddelde frequentie van alle woorden in de indexen. Indien beide woorden w_i en w_j precies een gemiddelde frequentie hebben dan is de waarde van functie $\alpha(i,j)$ gelijk aan 1. Indien beide frequenties hoger zijn dan zal $\alpha(i,j)$ een lagere waarde aannemen.
Hypothese: Het resultaat van het ACR-algoritme is een conceptuele ruimte Cr waarin woorden worden gerepresenteerd als punten in Cr en waarbij de woorden die in een gegeven verzameling van boeken **geassocieerd** voorkomen (dat wil zeggen in elkaars nabijheid optredend), dicht bij elkaar liggen.

6.4 Van conceptuele ruimte naar document-cluster-map

Nu met behulp van het ACR-algoritme op basis van de indexen van boeken een conceptuele ruimte is opgezet zou deze ruimte gebruikt kunnen worden in een informatie-zoeksysteem. Het is mogelijk om, net als in het WEBSOM-algoritme, eerst een woord-cluster-afbeelding te creëren en op basis hiervan een document-cluster-afbeelding.
De output van het ACR-algoritme beschrijft een conceptuele ruimte aan de hand van een verzameling woorden met bijbehorende referentievectoren. Het SOM-algoritme kan nu niet alleen worden gebruikt om de dimensionaliteit van deze ruimte te reduceren tot bijvoorbeeld twee. Nog belangrijker is dat het SOM-algoritme clusters van woorden kan onderkennen in de conceptuele ruimte. Binnen één cluster bevinden zich dan woorden die sterk zijn geassocieerd.
De woord-cluster-afbeelding die dan ontstaat is verder identiek aan de normale woord-cluster-afbeelding van het WEBSOM. Met behulp van deze afbeelding kunnen de documenten gecodeerd worden op dezelfde manier als bij het WEBSOM-algoritme zoals beschreven in paragraaf 5.1.3: Ieder document wordt gerepresenteerd door een document-vector. Het aantal componenten in deze vector is gelijk aan het aantal clusters op de woord-cluster-afbeelding. Voor ieder woord in de bewerkte index S_i van boek i wordt bepaald tot welke cluster deze behoort.
Er zijn nu een tweetal opties:

1. **Continue**: Per cluster wordt bepaald hoe vaak een woord uit deze cluster in de tekst voorkomt en dit aantal wordt de waarde van het desbetreffende component van de document-vector.
2. **Bitvector**: Zodra in het document één van de woorden uit een cluster voorkomt krijgt het desbetreffende component de waarde één. Als geen termen uit de cluster in het document voorkomen dan krijgt de desbetreffende component de waarde nul.

Uit experimenten die beschreven zijn in paragraaf 7.3.5 blijkt dat in ieder geval voor kleinere documentencollecties de bitvector-representatie leidt tot een betere clustering.
De documentenvectoren vormen de input voor de document-cluster-afbeelding.
In *figuur 6.10* is een overzicht gegeven van het volledige algoritme van indexen tot document-cluster-afbeelding. Dit volledige algoritme zullen we in het vervolg **het ACR-WEBSOM-algoritme** noemen.
Het ACR-WEBSOM-algoritme begint met de inversie van de indexen. Vervolgens kan hier een voorbewerking op plaats vinden zoals het verwijderen van woorden die weinig voorkomen. Op basis van deze voorbewerkte geïnverteerde indexen wordt middels het ACR-algoritme een conceptuele ruimte gevormd. Met behulp van het SOM-algoritme zal hier een woord-cluster-afbeelding worden gemaakt welke gebruikt wordt voor de codering van de documenten tot documentenvectoren. Deze vectoren dienen weer als input voor de document-cluster-afbeelding. Op dezelfde manier als bij het WEBSOM-algoritme kan de gebruiker op deze afbeelding clusters selecteren die voor haar/hem relevant lijken. Binnen deze clusters zullen zich semantisch gerelateerde documenten bevinden.



Figuur 6.10: Het volledige ACR-WEBSOM algoritme.

6.5 De programmacode

In bijlage 1 is de programmacode gegeven van het belangrijkste object van het ACR-WEBSOM-algoritme . Dit prototype is in Delphi 2.0 geschreven. De keuze van deze programmeertaal is gebaseerd op mijn bekendheid met deze taal. Een andere taal is misschien beter geschikt voor een operationele versie. Zo zou het dynamische geheugenbeheer van C++ beter geschikt zijn voor dit algoritme omdat dan de omvang van de arrays runtime zou kunnen worden bepaald, i.p.v. de huidige situatie waar dit voor compilatie moet gebeuren.
Dit programma gebruikt als input ASCII-bestanden welke indexen bevat die al zijn geïnverteerd door een ander programma dat niet in detail zal worden behandeld.
Het object Main waarvan de code is gegeven in bijlage 1 wordt gestart met behulp van de procedure Main.run. Vanuit deze procedure wordt eerst de **initialisatie-procedure** (Main.Initialize) opgestart waarin o.a. de vectoren willekeurige waarden krijgen. Vervolgens wordt de **lexicale analyse** (Main.LexicalAnalysis) uitgevoerd, oftewel de bewerkte indexen worden ingelezen. In de array van records <Lexicon> wordt de vocabulaire B_* opgeslagen met bij ieder woord het aantal keren dat deze voorkomt in de totale tekstketen. In <Wordstring> wordt de tekstketen T_* zelf opgeslagen in de vorm van een array van verwijzingen naar de woorden in Lexicon.
Hierna vindt de **voorbewerking** (Main.PreProcessing) plaats. Woorden die een lagere frequentie hebben dan de constante <LowerThreshold> worden verwijderd uit <Lexicon> en <Wordstring>.
Hierna wordt de **semantische analyse** (Main.SemanticAnalysis) uitgevoerd. Dit is een iteratief proces dat <NumberOfEpochs> keer wordt uitgevoerd.
Binnen de semantische analyse wordt per woord de buurtverzameling <Hood> bepaald. Tussen het woord waar de focus op ligt en de overige woorden in de buurtverzameling wordt vervolgens de associatie versterkt met behulp van de **Connect** procedure (Main.Connect). Nadat de tekstketen volledig is doorlopen wordt de **vergeet-procedure** (Main.Forget) uitgevoerd waarin de afstand tussen alle referentievectoren wordt vergroot met behulp van de **afstoot-functie** (Main.RepulseFunction).
Nadat de iteraties van de semantische analyse zijn voltooid worden de **resultaten opgeslagen** (Main.SaveResults).

Hoofdstuk 7: Praktijktests

“The real purpose of scientific method is to make sure Nature hasn’t misled you into thinking you know something you don’t actually know.’ (Pirsig, p.94)

In dit hoofdstuk zullen de belangrijkste hypothesen die ten grondslag liggen aan het ACR-WEBSOM-algoritme worden onderworpen aan een praktijktest. Als eerste zal de test-opstelling worden besproken (7.1) waarna de testmethodiek wordt besproken waarmee de verscheidene hypothesen zullen worden getoetst (7.2). Hierna volgt een beschrijving van de experimenten en de resultaten die hieruit voortvloeien (7.3).

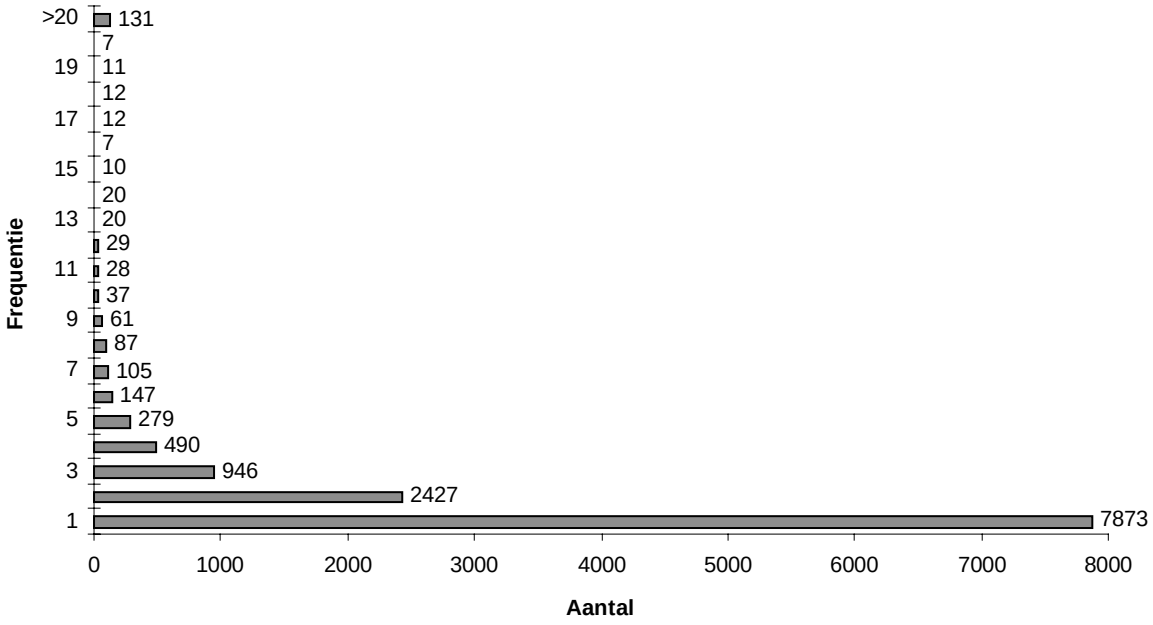
7.1 Testopstelling

Alle experimenten zijn gedaan op een seriële computer met een 133Mhz Pentium processor en 48Mb intern geheugen met als operating systeem Windows-95. De voorbewerking van de dataset, toepassing van het ACR-algoritme en de codering van de documenten met behulp van de woord-cluster-afbeelding zijn uitgevoerd met programma’s die geschreven zijn in Borland Delphi 2.0, een ontwikkelomgeving die Object-Pascal combineert met een visuele gebruikersinterface. Het vormen van de woord-cluster-afbeelding en de document-cluster-afbeelding is uitgevoerd met het SOM-pakket (Kohonen e.a.2) dat is geschreven in de object-georiënteerde programmeertaal C++.

Om beter inzicht te krijgen in de resultaten van de experimenten zal eerst de gebruikte testset kort worden beschreven (7.1.1), waarna de implementatie van de voorbewerking (7.1.2) en de nabewerking (7.1.3) aan bod komen. De implementatie van het ACR-algoritme is gedetailleerd weergegeven in de vorm van de programmacode in bijlage 1 en is al kort behandeld in paragraaf 6.5 zodat deze in dit hoofdstuk verder niet aan de orde zal komen.

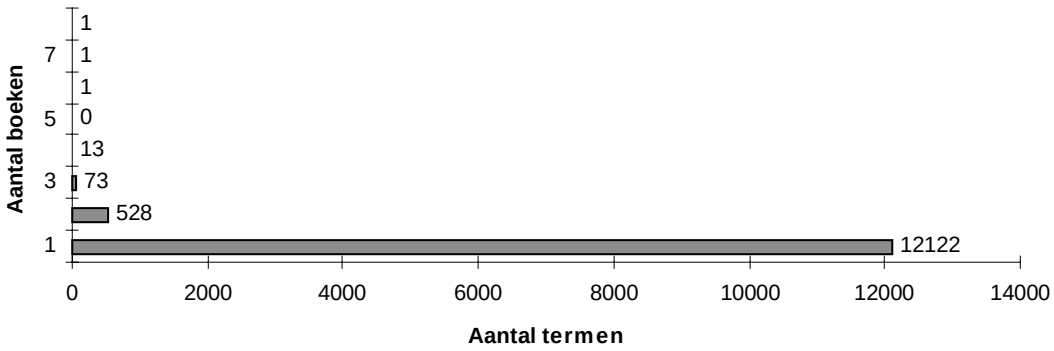
7.1.1 Testset

De oorspronkelijke testset bestaande uit ca. 50 indexen van boeken is verschaft door Niek Weustink. Hieruit zijn de indexen geselecteerd die behooren tot Nederlandstalige documenten en waarvan de indexverwijzing geschiedt op basis van paginanummers. De overgebleven set van 39 indexen is handmatig uitgebreid met nog één index, zodat de dataset uiteindelijk uit 40 documenten bestaat. Deze dataset komt grotendeels overeen met de set die gebruikt is voor de tests met het indexSOM-algoritme (Weustink).
Nadere analyse toont aan dat deze indexen in totaal 12.739 verschillende termen bevat, oftewel $| B_* | = 12.739$. Verder blijkt dat het grootste gedeelte, namelijk 7.873 woorden slechts één keer voorkomen in de tekst-keten T_*. De overige frequentieverdeling is getoond in *figuur 7.1*. Verder bleek dat $| T_* | = 33.730$.



Figuur 7.1: Histogram van de woordfrequentie over de gehele tekst-keten T_*

Belangrijk voor het kunnen vergelijken van diverse indexen is het aantal indextermen dat zij gemeen hebben. In *figuur 7.2* is weergegeven dat verreweg de meeste termen, namelijk 12.122 van de 12.739 indextermen (95%) uitsluitend in één index voorkomen.



Figuur 7.2: Aantal indextermen dat in één of meer boeken voorkomen.

De documenten uit de testset kunnen grofweg worden ingedeeld in 17 categorieën, te weten:

- Management : deze boeken behandelen managementtheorie in zijn algemeenheid
- Marketing : theorie achter het commercieel management
- Ondernemen : informatie over de verschillende facetten van het ondernemen
- Verslag : handboek over externe verslaggeving en controle
- Communicatie : communicatietheorie
- Arbo : handboek over de arbo-wetgeving in Nederland
- Stress : informatie over stress, voornamelijk op de werkplek
- New Age : boeken over populaire nieuwe spirituele stromingen
- Filosofie : documenten die filosofie in zijn algemeenheid behandelen
- Medisch : handboeken voor medici
- Tandarts : handboeken voor tandheelkundige
- Anatomie : beschrijvingen van de anatomie van de mens
- Biologie : naslagwerken over biologie en met name plantkunde
- Natuurkunde : boeken over natuurkunde in zijn algemeenheid
- Paard : informatie over paardendressuur en -verzorging
- Imker : informatie over het houden van bijen
- Wedstrijd : boek over wedstrijd-elementen en competitie

Ieder document is voorzien van een label waarin deze categorie vermeld is, bijvoorbeeld: ‘management1’ en ‘medisch3’.
Aangezien deze testset relatief veel verschillende onderwerpen bevat is later een **kleinere testset** van 10 indexen samengesteld die nog slechts 4 onderwerpen bevatte, te weten:

- Management
- Medisch
- Natuurkunde
- Anatomie

7.1.2 Voorbewerking

De indexen waren geleverd in ASCII-formaat. Iedere term is ontdaan van niet-alfabetische karakters waaronder spaties en beperkt tot 15 tekens mede om geheugen te besparen. In de indexen kwamen ook meerdere malen subtermen voor, zoals deze subtermen bij de indexterm ‘diffusieproces’:

...
diffusiekromme 256,258
diffusieproces 308,340
- op macro-niveau 341
- op micro-niveau 340
- op meso-niveau 341

direct product costs 363
...

Deze subtermen zijn bij de voorbewerking verwijderd.
Vervolgens zijn de indexen B_i bewerkt zoals beschreven in paragraaf 6.1 en 6.3 tot de tekstketens T_i.
Deze ketens vormen de input voor het programma waarvan de code is gegeven in bijlage 1. Dit programma voegt de afzonderlijke ketens T_i samen tot de totale keten T_* en leidt hier tevens de vocabulaire B_* en de frequentie $freq_i$ waarmee woord $b_i \in B_*$ voorkomt in T_* uit af.
Alle woorden met een frequentie lager dan een constante drempelwaarde worden verwijderd uit B_* en T_*.
Vervolgens wordt door dit programma het ACR-algoritme uitgevoerd. Als output levert dit programma dan de indextermen B_* met de bijbehorende verzameling referentievectoren X.

7.1.3 Nabewerking

Met behulp van het SOM_PAK wordt aan de hand van het outputbestand ‘output.txt’ van het hierboven genoemde programma een woord-cluster-afbeelding gevormd. Hiervoor zijn de volgende commando’s gebruikt:

- randinit -din output.txt -cout smap.txt -xdim 20 -ydim 20 -topol hexa -neigh bubble
- vsom -din output.txt -cin smap.txt -cout smap.txt -rlen 1000 -alpha 0.05 -radius 20
- vsom -din output.txt -cin smap.txt -cout smap.txt -rlen 10000 -alpha 0.02 -radius 3
- vcal -din output.txt -cin smap.txt -cout wordmap.txt -numlabs 100

Het eerste commando genereert een willekeurig geïnitialiseerde SOM met hexagonale toplogie onder de naam ‘smap.txt’. De volgende twee commando’s leiden tot aanpassing van deze referentievectoren met een learning-rate van respectievelijk 0.05 en 0.02, een radius van de topologische buurtfunctie van respectievelijk 20 en 3 en een aantal aanpassingsstappen van respectievelijk 1.000 en 10.000. Als laatste wordt voor iedere referentievector en bijbehorende indexterm bepaald tot welke cluster deze behoort met behulp van het vcal programma.
Deze commando’s resulteren in de woord-cluster-afbeelding ‘wordmap.txt’ waarop 20 bij 20 clusters van woorden zijn weergeven. Met behulp van dit bestand en de originele tekst-ketens T_i worden de documenten gecodeerd zoals beschreven in paragraaf 5.1.3. Deze

codering wordt opgeslagen in het bestand ‘vectors.txt’. Met behulp van dit bestand wordt de document-cluster-afbeelding gemaakt middels het SOM_PAK:

- randinit -din vectors.txt -cout smap.txt -xdim 5 -ydim 5 -topol hexa -neigh bubble
- vsom -din vectors.txt -cin smap.txt -cout smap.txt -rlen 1000 -alpha 0.05 -radius 5
- vsom -din vectors.txt -cin smap.txt -cout smap.txt -rlen 10000 -alpha 0.02 -radius 3
- vcal -din vectors.txt -cin smap.txt -cout documap.txt -numlabs 100
- umat -cin vectors.txt -ps >documap.ps

Dit resulteert in een afbeelding van 5 bij 5 eenheden. Het laatste commando visualiseert de document-cluster-afbeelding in een PostScript-bestand ‘documap.ps’.

7.2 Testmethodiek

In het vorige hoofdstuk is een nieuw algoritme geïntroduceerd. Dit algoritme is gebaseerd op een aantal hypothesen die nog geverifieerd dienen te worden. Hiernaast kent het algoritme een groot aantal parameters waarvoor een optimale waarde moet worden bepaald. We hebben hier echter te maken met een experimenteel prototype en het gaat voor deze scriptie dan ook te ver om gelijk ieder aspect van het ACR- en het ACR-WEBSOM-algoritme zorgvuldig te verifiëren en te optimaliseren. De praktijktests in dit hoofdstuk hebben dan ook voornamelijk een **verkennend karakter**.
Om het ACR-WEBSOM-algoritme te waarderen is één aspect van bijzonder belang: de semantische ordening, zowel van woorden in de woord-cluster-afbeelding als van documenten in de document-cluster-afbeelding. Deze **semantische ordening** is echter **moeilijk te kwantificeren** en tot op zekere hoogte **subjectief**. Daarom zal bij deze test hoofdzakelijk worden volstaan met een visuele inspectie van de clusters of de volledige afbeelding. In de toekomst is het wenselijk dat hier echter een meer objectieve maatstaf voor komt. Dit is echter een probleem dat niet alleen aan het ACR-WEBSOM-algoritme is toe te kennen maar ook aan het WEBSOM-algoritme.
Het ACR-WEBSOM-algoritme onderscheidt zich ten opzichte van het WEBSOM-algoritme door het vormen van een conceptuele ruimte uit een index. De eigenschappen van deze conceptuele ruimte zal ook onderzocht moeten worden. Een hoge dimensionale ruimte is echter moeilijk te visualiseren dus dit zal geschieden via een tweedimensionale weergave van deze ruimte namelijk de woord-cluster-afbeelding. Hierbij zal moeten worden gekeken of de clusters inderdaad geassocieerde woorden bevatten. Dit zal worden behandeld in paragraaf 7.3.1.
Een experiment moet herhaalbaar zijn. Hierbij moet worden geverifieerd of het algoritme inderdaad vanuit meerdere willekeurige beginposities tot een soortgelijke ordening komt zoals gesteld in paragraaf 6.2.4. Hiervoor zal de woord-cluster-afbeelding vanuit meerdere willekeurige beginposities worden gegenereerd en de resulterende clusters zullen worden vergeleken in paragraaf 7.3.2.
In paragraaf 6.2.4 is tevens gesteld dat uit praktijkervaring is gebleken dat een hogere learning-rate in de eerste leerstappen het vormen van de ordening versnelt. Dit zal worden onderbouwd in paragraaf 7.3.3 door woord-cluster-afbeeldingen te genereren respectievelijk met en zonder hoge startwaarden en deze afbeeldingen te vergelijken.
Actief vergeten behelst een groot aantal berekeningen en de noodzaak van deze omvangrijke vergroting van de tijdscomplexiteit zal dus moeten worden aangetoond. In paragraaf 7.3.4 zal dit geschieden aan de hand van woord-cluster-afbeeldingen welke zijn

gegenereerd zonder vergeetregel, waarbij wordt nagegaan of er een correcte clustering plaatsvindt.
Het belangrijkste is om na te gaan of het ACR-WEBSOM-algoritme überhaupt werkt. Dit betekent dat het algoritme in staat moet zijn de documenten te ordenen naar onderwerp. Dit gebeurt door visuele inspectie van de document-cluster-afbeelding. Hierbij zal worden gekeken of documenten uit dezelfde categorie dicht bij elkaar liggen op de document-cluster-afbeelding. Tevens is in paragraaf 6.4 gesteld dat bitvectors als document-representatie een betere ordening geven van de document-cluster-afbeelding. Dit is ook getoetst in paragraaf 7.3.5.
Een grote beperking aan het ACR-algoritme is de grote tijdscomplexiteit. Dit zal worden onderzocht in paragraaf 7.3.6.
Het ACR-WEBSOM-algoritme kent een aantal parameters waarvoor een optimale waarde moet worden gevonden. Het vergt voor deze scriptie te veel ruimte en tijd om iedere parameter in detail te bestuderen. In paragraaf 7.3.7 zal dan ook slechts beknopt bij de waarden van deze parameters worden stilgestaan.

7.3 Tests en resultaten

Zoals gezegd gaat het voor deze scriptie te ver om alle aspecten van het ACR-WEBSOM-algoritme te verifiëren en te optimaliseren. Bij de experimenten in deze paragraaf zullen dan ook enkele aspecten constant worden gehouden en verder niet worden onderzocht:

- Bij de testset van 40 documenten geldt een minimale woordfrequentie van 2. Bij de testset van 10 documenten is deze 1.
- De associatieregel is genormaliseerd zoals beschreven in paragraaf 6.2.2.
- De afstootfunctie verloopt zoals beschreven in paragraaf 6.2.3, met een constante waarde over het bereik [0,1] en een afnemende waarde voor het bereik ⊙ 1,∞ ◎.
- De vorm van de buurtfunctie zoals beschreven in paragraaf 6.3 blijft gelijk met n=4.
- Bij de associatieregel wordt de relatieve frequentie van de woorden meegenomen, zoals beschreven in paragraaf 6.3.
- De instellingen van het SOM-algoritme blijven gedurende de tests zoals beschreven in 7.1.3.

De keuze voor de invulling van deze aspecten wordt dus niet geverifieerd.
Een aantal parameters worden wel gewijzigd bij de experimenten. Bij iedere afbeelding in deze paragraaf is dan ook in tabelvorm weergegeven wat deze waarden zijn. In deze tabellen staat vermeldt:

- Aantal indexen : grootte van de testset.
- Learningrate : functie voor het verloop van de learningrate over de tijd, gemeten in epochs
- Epochs : aantal keer dat de volledige tekstketen wordt doorlopen
- Dimensies : aantal dimensies van de conceptuele ruimte
- Overige parameters: : overige variaties ten opzichte van het algoritme zoals het gepresenteerd is in hoofdstuk 6.

Zoals gezegd zal bij de experimenten worden voldaan met een visuele inspectie van de woord-cluster-afbeeldingen en de document-cluster-afbeeldingen. Bij de woord-cluster-

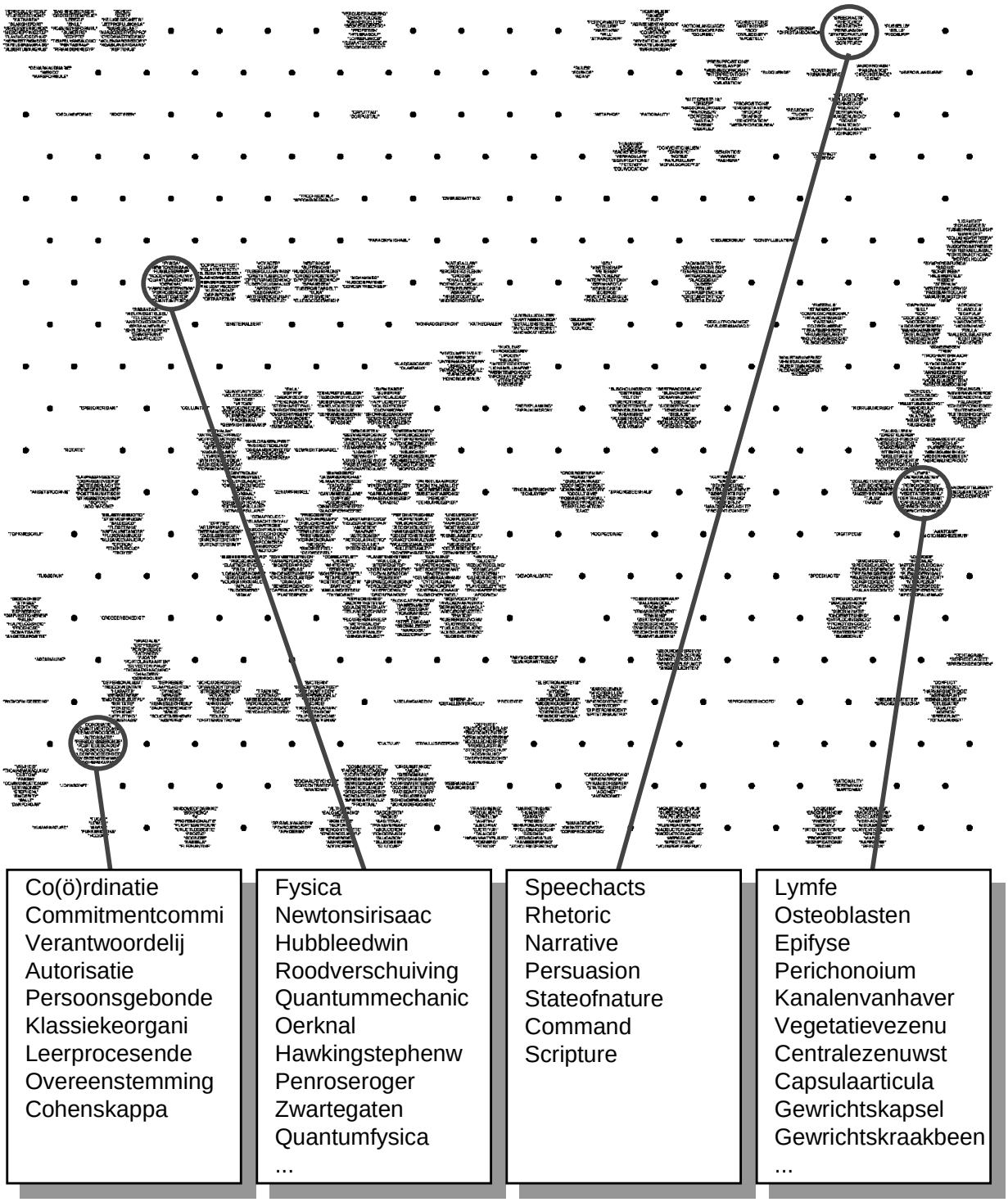
afbeeldingen zal hoofdzakelijk worden gekeken naar de inhoud van individuele clusters zoals is afgebeeld in *figuur 7.3*.
De document-cluster-afbeelding zal wel in zijn geheel worden beschouwd. Deze afbeelding is opgebouwd uit hexagrammen welke de concentratie van documentvectoren aangeven aan de hand van een kleurcode: een lichte kleur geeft aan dat de documentvectoren in het desbetreffende gebied dicht bij elkaar liggen. Een donkere kleur duidt op een lage dichtheid van documentvectoren. Verder zijn op deze afbeeldingen de labels van de documenten weergegeven op de plaats waar zich de eenheid bevindt op het raster van de SOM waaraan dit document is gerelateerd. Indien geen enkel document is gerelateerd aan een eenheid dan wordt deze eenheid weergegeven met een zwarte punt.

7.3.1 Clustering van woorden

Om na te gaan of het ACR-WEBSOM-algoritme inderdaad in staat is om een conceptuele ruimte op te zetten waarbij sterk geassocieerde woorden dicht bij elkaar liggen zal worden gekeken naar de tweedimensionale weergave van deze ruimte: de woord-cluster-afbeelding. In *figuur 7.3* is een woord-cluster-afbeelding weergegeven waarbij enkele clusters zijn weergegeven waarbinnen duidelijk zichtbare associatieve clustering heeft plaatsgevonden. Hiermee blijkt dat het algoritme inderdaad in staat is associatieve relaties tussen termen te vinden en te representeren. In *tabel 7.1* zijn de waarden van de vrije parameters weergegeven zoals die zijn toegepast bij deze testsessie.

Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
40	2/min(epoch,10)	250	5	-

Tabel 7.1: Waarden van de vrije parameters bij het genereren van figuur 7.3



Figuur 7.3: Woord-cluster-afbeelding

De associatieve ordening zoals weergegeven in *figuur 7.3* Is echter pas bruikbaar indien de termen waartussen deze associaties zijn gevonden uit verschillende documenten afkomstig zijn. Later onderzoek heeft aangetoond dat de termen die zijn weergegeven in *figuur 7.3* binnen een cluster vrijwel uitsluitend uit één boek afkomstig zijn. Dit is nadelig voor de werking van het algoritme, met name voor het vormen van de document-cluster-afbeelding. Indien ieder document gekenmerkt wordt door zijn eigen unieke clusters op de woord-cluster-afbeelding dan bestaat er geen overeenkomst tussen de vector-representaties van documenten, zelfs niet als deze documenten gerelateerd zijn. In een experiment is geprobeerd dit te verhelpen door een woord-cluster-afbeelding te kiezen met minder eenheden, zodat noodgedwongen meer termen in één cluster vallen.

Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
40	2/min(epoch,10)	250	5	Woord-cluster-afbeelding met 10 bij 10 eenheden

Tabel 7.2: Waarden van de vrije parameters bij het genereren van figuur 7.4

FYSICA	new age1				
EINSTEINALBERT	new age1	natuurkunde1	natuurkunde2		
DOPPLEREFFECT	new age1	natuurkunde1	natuurkunde2		
ELEKTROMAGNETIS	new age1	natuurkunde1	natuurkunde2		medisch4
FARADAYMICHAEL	new age1	natuurkunde1	natuurkunde2		
RELATIVITEITSTH	new age1		natuurkunde2		
QUANTUMMECHANIC	new age1				
OERKNAL	new age1	natuurkunde1			
ELEMENTAIREDEEL	new age1				
WEINBERGSTEVEN	new age1		natuurkunde2		
HAWKINGSTEPHENW	new age1				
KALUZATHEODOR	new age1		natuurkunde2		
KLEINOSKAR	new age1		natuurkunde2		
PENROSEROGER	new age1				
ZWARTEGATEN	new age1				
QUANTUMFYSICA	new age1		natuurkunde1		
BOHRNIELS		natuurkunde1	natuurkunde2	filosofie2	
NEUTRONEN		natuurkunde1	natuurkunde2		
PROTONEN		natuurkunde1	natuurkunde2		

Figuur 7.4: Termen in een woord-cluster (links) waarbij is vermeldt in welke documenten deze term voorkomt.

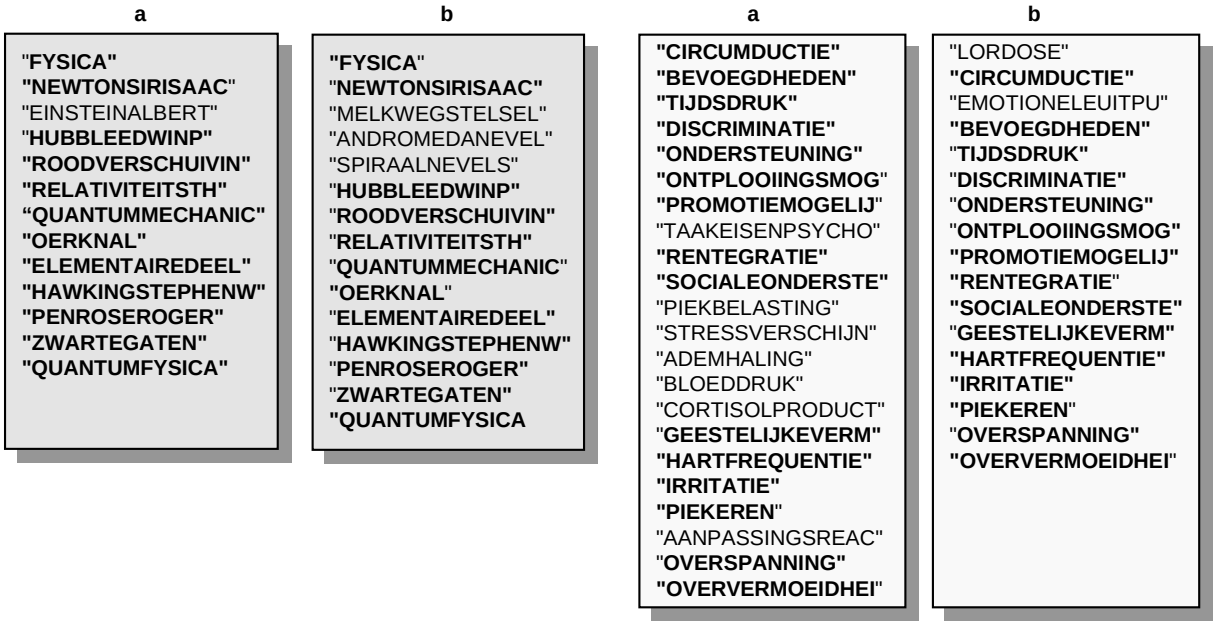
In *figuur 7.4* kunnen we zien dat bij dit experiment een cluster termen uit verschillende documenten bevat, waarbij de termen nog steeds associatief gerelateerd zijn. Voorlopige experimenten bevestigen dat een lager aantal eenheden in de woord-cluster-afbeelding inderdaad leidt tot een betere ordening van de document-cluster-afbeelding.

7.3.2 Herhaalbaarheid van het experiment

Om na te gaan of de clustering consistent is vanuit iedere willekeurige beginsituatie is het mogelijk de clusters uit twee aparte testsessies te vergelijken. De kans dat een cluster aan exact dezelfde eenheid in een SOM gerelateerd wordt is klein gegeven de willekeurige startpositie van een SOM. Daarom worden verwante clusters gevonden door te zoeken naar die clusters op beide afbeeldingen waar een van te voren bepaald woord in voorkomt.
In *tabel 7.3* zijn de vrije parameters weergegeven welke voor de twee tests zijn gebruikt. In *figuur 7.5* zijn clusters uit beide woord-cluster-afbeeldingen weergegeven, waarbij in beide afbeeldingen respectievelijk de term 'fysica' en 'tijdsdruk' voorkomen. De keuze voor deze termen was gedeeltelijk willekeurig. Daarbij werd echter wel rekening gehouden met het feit dat deze term in een cluster voorkwam die niet te groot (>25 termen) of te klein (<10 termen) was om een zinnig oordeel over te geven.

	Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
a	40	2/min(epoch,10)	250	5	-
b	40	2/min(epoch,10)	250	5	-

Tabel 7.3: Waarden van de vrije parameters bij het genereren van figuur 7.5



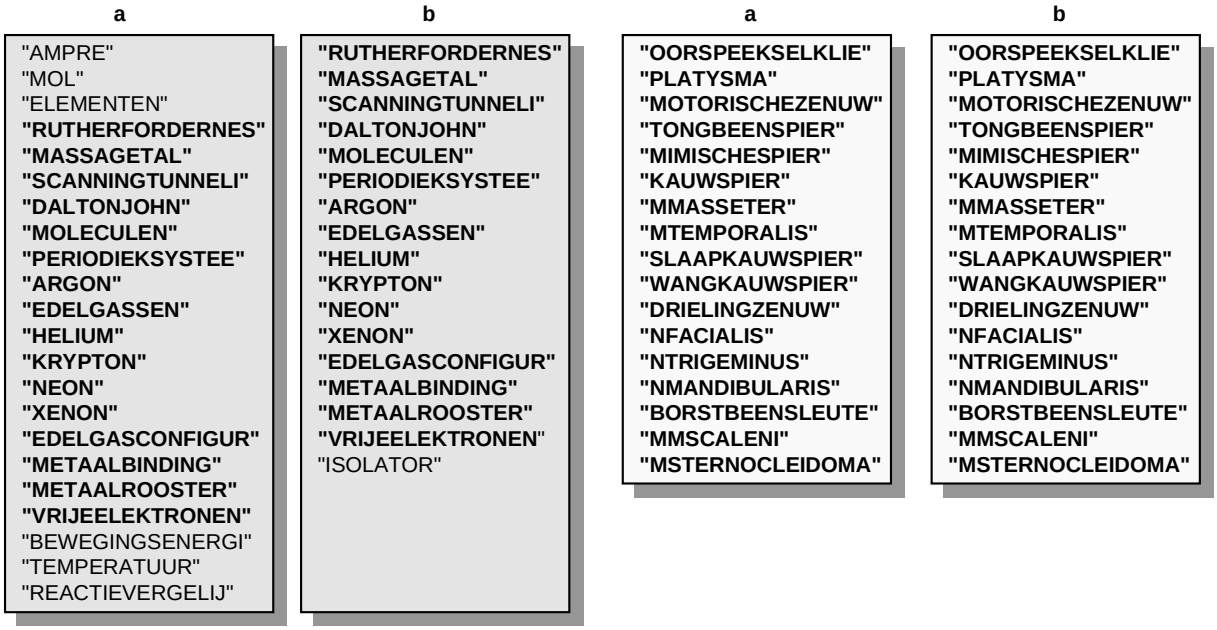
Figuur 7.5: Vergelijking van twee clusters uit twee individuele woord-cluster-afbeeldingen. De dik gedrukte termen komen in beide afbeeldingen in dezelfde cluster voor.

Uit *figuur 7.5* blijkt dat inderdaad de uiteindelijke clustering grotendeels consistent is vanuit verschillende startposities. De minimale verschillen tussen beide afbeeldingen kunnen ook veroorzaakt zijn door het vormen van de woord-cluster-afbeelding vanuit de conceptuele ruimte aan de hand van het SOM-algoritme. Daar de verschillen zo klein zijn zal dit echter niet verder worden onderzocht.
Ondanks de verschillen zijn alle woord-clusters echter min of meer correct in die zin dat de woorden in de clusters associatief gerelateerd kunnen worden genoemd.

Als extra verificatie kan het experiment herhaald worden met de beperkte testset van 10 documenten. In *tabel 7.4* zijn de waarden weergegeven waarmee *figuur 7.6* is gegenereerd. Deze keer zijn respectievelijk de woorden ‘edelgassen’ en ‘motorischezenuw’ gebruikt voor het identificeren van de clusters.

	Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
a	10	2/min(epoch,15)	250	5	-
b	10	2/min(epoch,15)	250	5	-

Tabel 7.4: Waarden van de vrije parameters bij het genereren van figuur 7.6



Figuur 7.6: Vergelijking van twee clusters uit twee individuele woord-cluster-afbeeldingen. De dik gedrukte termen komen in beide afbeeldingen in dezelfde cluster voor.

Uit *figuur 7.6* blijkt wederom dat de clustering consistent is. De clusters behorende bij de term ‘motorischezenuw’ zijn zelfs volkomen identiek.

7.3.3 Hogere startwaarden van de learningrate

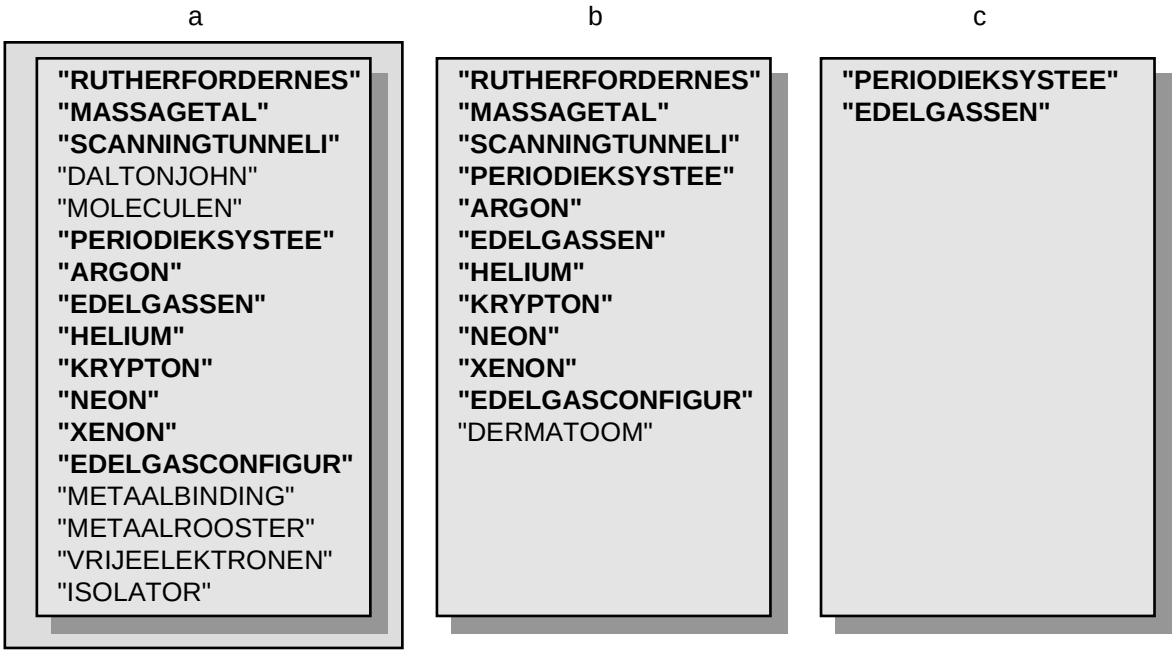
In paragraaf 6.2.4 is gesteld dat in het begin van een trainingssessie een hoge learning-rate is gewenst om snel vanuit de volledig willekeurige beginsituatie naar een globale ordening te komen. Tot nu toe is de learning-rate gesteld op:

learning-rate = 2/min(epoch,15)

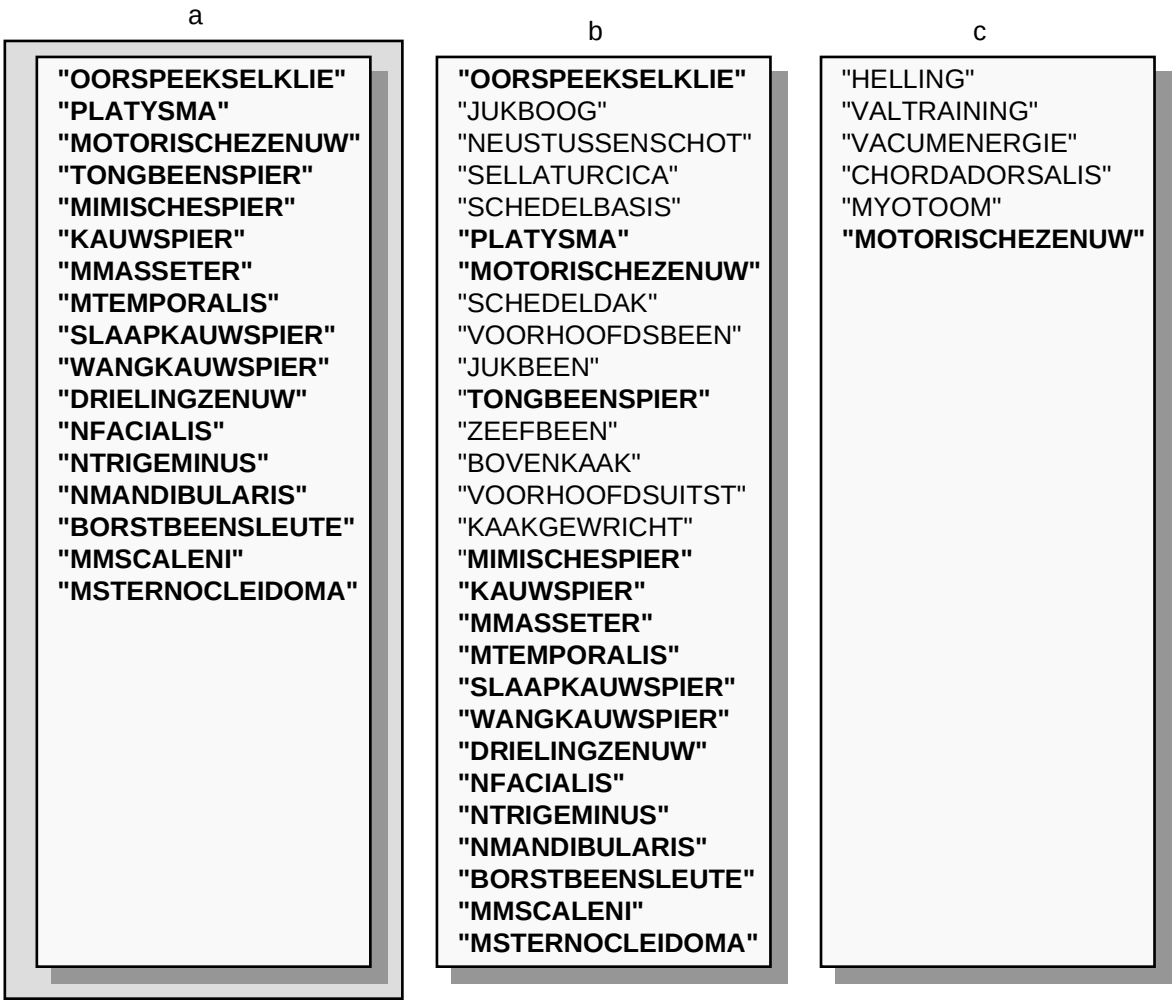
Dit betekent dat in 15 leerstappen de learning-rate terugloopt van 2 naar 2/15. Hierna blijft de learning-rate constant op deze lage waarde. De hoge startwaarde van de learning-rate heeft als gevolg dat de ordening sneller verloopt. Dit betekent dat het algoritme minder stappen nodig heeft om dezelfde ordening te bereiken. Na een groot aantal epochs is de ordening met of zonder hoge startwaarden echter gelijk. Experimenten die hier niet verder zullen worden beschreven tonen aan dat dit al geldt voor ca.100-150 epochs.
Het effect van de hogere startwaarden is dus het beste te zien bij een klein aantal epochs. In *figuur 7.7* en *figuur 7.8* zijn clusters vergeleken welke zijn gegenereerd na 50 epochs. Ter vergelijking zijn 'voltooide' clusteringen getoond. In *tabel 7.5* zijn de waarden voor de parameters van deze experimenten vermeld.

	Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
a	10	2/min(epoch,15)	250	5	-
b	10	2/min(epoch,15)	50	5	-
c	10	2/15	50	5	-

Tabel 7.5: Waarden van de vrije parameters bij het genereren van figuur 7.7 en figuur 7.8



Figuur 7.7: Vergelijking van clusters. Cluster a is een 'voltooide' cluster na 250 epochs. B en c zijn gevormd na 50 epochs, waarbij b is gegenereerd met hoge startwaarden voor de learningrate.



Figuur 7.8: Clustering zoals in figuur 7.7. Echter voor een andere cluster

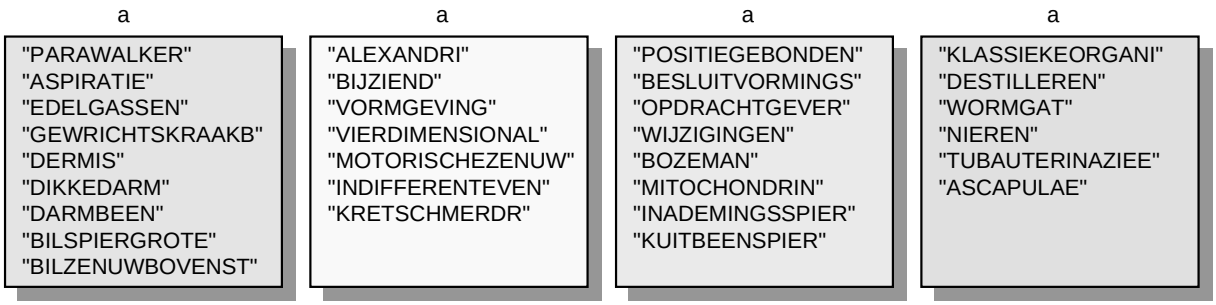
In respectievelijk *figuur 7.7* en *figuur 7.8* zijn wederom de termen ‘edelgassen’ en ‘motorischezenuw’ gebruikt om de clusters te identificeren. Beide figuren tonen aan dat hogere startwaarden een betere clustering geven. In *figuur 7.8* blijkt zelfs dat zonder hogere startwaarden de term ‘motorischezenuw’ in een cluster is geplaatst die verder met geen enkele term overeenkomt met de ’voltooide’ cluster.

7.3.4 Effect van de vergeetregel

De vergeetregel is een belangrijk onderdeel van het ACR-algoritme. Door een woord-cluster-afbeelding te genereren zonder de vergeetregel wordt het effect dat deze regel heeft zichtbaar gemaakt. In *tabel 7.6* zijn de waarden van de vrije parameters weergegeven waarmee *figuur 7.9* is gegenereerd. De parameters zijn gelijk aan die waarmee *figuur 7.6* in paragraaf 7.3.2, met uitzondering van de omissie van de vergeetregel.

	Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
a	10	2/min(epoch,15)	250	5	geen vergeetregel

Tabel 7.6: Waarden van de vrije parameters bij het genereren van figuur 7.9.



Figuur 7.9: Clustering zonder de vergeetregel

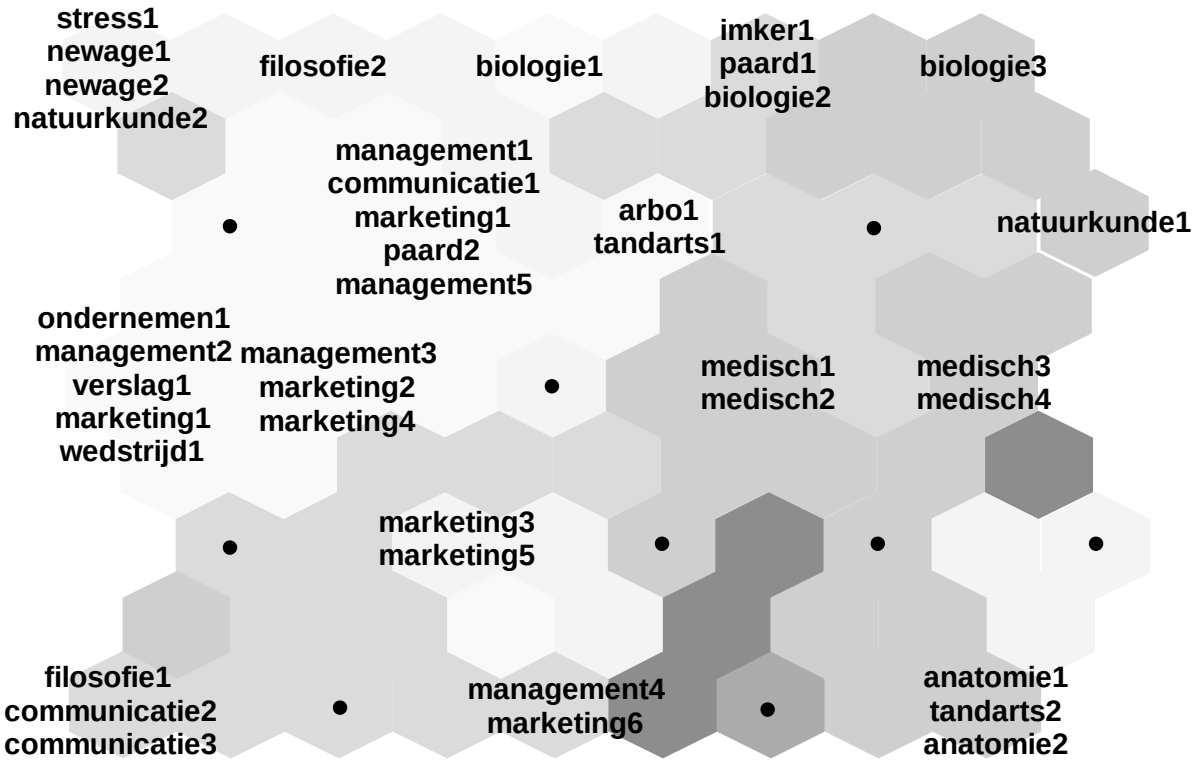
In *figuur 7.9* zijn enkele clusters weergegeven uit de woord-cluster-afbeelding. De linkse twee clusters zijn gevonden door wederom te zoeken naar de termen ‘edelgassen’ en ‘motorischezenuw’ zodat deze twee clusters vergeleken kunnen worden met de clustering van *figuur 7.6*. Hiernaast zijn twee willekeurig gekozen clusters weergegeven.
Zoals uit de figuur blijkt is de clustering zonder vergeetregel bijzonder slecht te noemen. Verdere experimenten die verder niet behandeld zullen worden bevestigen dit. Hiermee is de noodzaak van de vergeetregel aangetoond.

7.3.5 Ordening in de document-cluster-afbeelding

Het uiteindelijke resultaat van het ACR-WEBSOM-algoritme is de document-cluster-afbeelding. Op deze afbeeldingen dienen de documenten naar semantische inhoud te zijn geordend. In *figuur 7.10* is een document-cluster-afbeelding weergegeven.

Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
40	2/min(epoch,10)	100	5	bit-document-vector

Tabel 7.7: Waarden van de vrije parameters bij het genereren van figuur 7.10

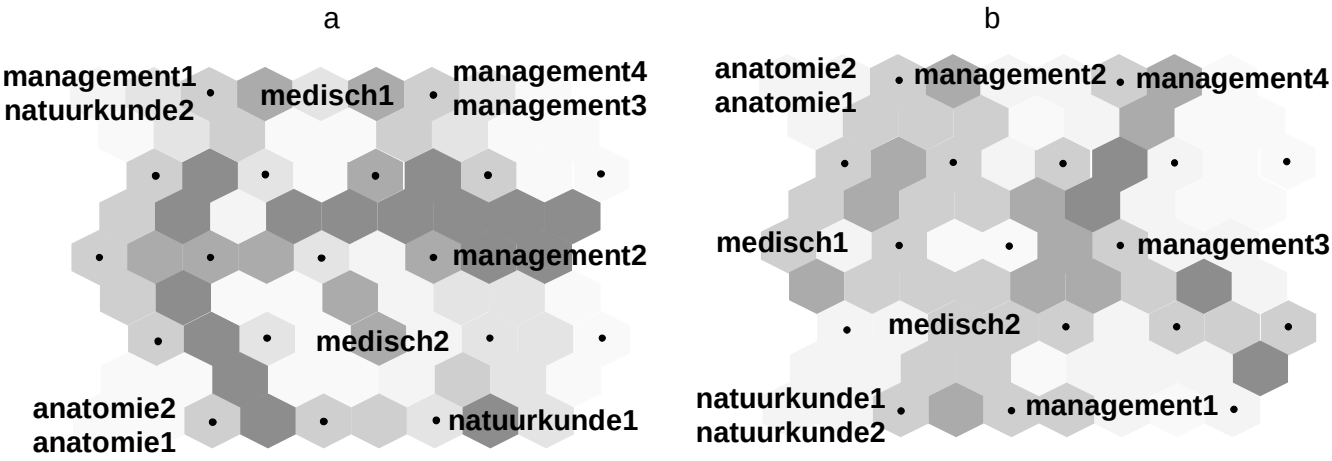


Figuur 7.10: De document-cluster-afbeelding

Uit *figuur 7.10* blijkt dat er inderdaad sprake is van een zekere ordening van documenten. Zo zijn alle 4 de boeken uit de categorie ‘medisch’ ingedeeld in twee naburige clusters. De drie boeken uit de categorie ‘biologie’ zijn naast elkaar te vinden, waarbij de gerelateerde categorie ‘imker’ is opgenomen in deze clusters. Opvallend is dat het onderscheid tussen de sterk gerelateerde categorieën ‘management’ en ‘marketing’ door het systeem niet is opgemerkt. Dit is te verklaren door het feit dat beide categorieën voor een groot gedeelte dezelfde onderwerpen behandelen.
Bij deze afbeelding is gebruik gemaakt van bitvector representatie voor de documenten zoals beschreven in paragraaf 6.4. Deze keuze kan worden onderbouwd door deze manier van representatie te vergelijken met de continue vectorrepresentatie zoals in *figuur 7.11*

	Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
a	10	2/min(epoch,15)	250	5	continue document-vector
b	10	2/min(epoch,15)	250	5	bitvector

Tabel 7.8: Waarden van de vrije parameters bij het genereren van figuur 7.11

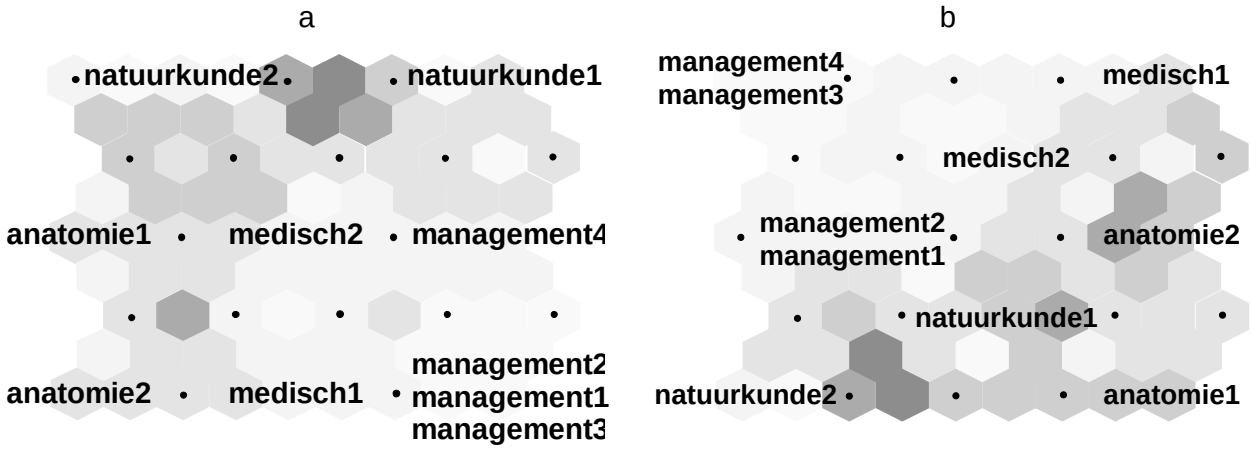


Figuur 7.11: Document-cluster-afbeelding met continue documentvector (a) en bit-documentvector (b).

Uit *figuur 7.11* blijkt dat een betere ordening ontstaat door gebruik te maken van bitvector-representatie. Dit valt te verklaren door het feit dat het onderscheid tussen documenten uit verschillende categorieën duidelijker wordt: behandelt een document een bepaald onderwerp wel, al is het maar een klein gedeelte, dan heeft de vector voor de desbetreffende cluster op de woord-cluster-afbeelding als waarde één. Wordt dit onderwerp niet behandeld dan heeft deze component van de documentvector de waarde nul. Bij continue documentvector-representatie is dit verschil meer genuanceerd en daardoor moeilijker op te merken door het SOM-algoritme. Het is echter niet uitgesloten dat voor grotere documentencollecties deze nuancering juist weer noodzakelijk is.
In *figuur 7.12* zijn nog een tweetal document-cluster-afbeeldingen weergegeven welke zijn gegenereerd met de waarden van *tabel 7.9*.

	Aantal indexen	LearningRate	Epochs	Dimensies	Overige parameters
a	10	2/min(epoch,15)	100	5	-
b	10	2/min(epoch,15)	250	5	-

Tabel 7.9: Waarden van de vrije parameters bij het genereren van figuur 7.12



Figuur 7.12: Document-cluster-afbeeldingen

7.3.6 Tijdscomplexiteit

De hoeveelheid berekeningen voor het ACR- en het ACR-WEBSOM-algoritme is groot: Het uitvoeren van het programma dat beschreven is in bijlage 1 kost veel tijd, zoals weergegeven in *tabel 7.10.* In deze tabel is weergegeven: het aantal trainingstappen, het aantal indexen, het aantal dimensies van de conceptuele ruimte, de minimale frequentie waarmee een woord moet voorkomen in de tekst-keten om opgenomen te worden in T_* , het aantal woorden dat hierna overbleef in de vocabulaire, het aantal woorden in de tekst-keten en de bewerkingstijd.

Epochs	Aantal indexen	Dimensies	Threshold	$\lvert B_* \rvert$	$\lvert T_* \rvert$	Tijd
250	10	5	1	1.407	5.729	2u43
100	10	5	1	1.407	5.729	1u08
50	10	5	1	1.407	5.729	0u34
100	10	3	1	1.407	5.729	0u45
100	10	2	1	1.407	5.729	0u29
250	40	5	2	2.439	20.621	8u10
100	40	5	2	2.439	20.621	4u25
50	40	5	2	2.439	20.621	1u38

Tabel 7.10: bewerkingstijd voor het ACR-algoritme

Tabel 7.10 laat zien dat grotere documentencollecties al snel een grote bewerkingstijd eisen. Nader onderzoek wijst uit dat 95 tot 96 procent van deze tijd in beslag wordt genomen door

de uitvoering van de vergeetregel. Het overige gedeelte is hoofdzakelijk toe te schrijven aan de uitvoering van de associatieregel.
De vergeetregel moet de afstand tussen alle referentievectoren vergroten en moet daarvoor $n^2/2$ maal dezelfde procedure uitvoeren, waarbij n gelijk is aan $| B_* |$, oftewel het aantal woorden in de vocabulaire. Ten opzichte van het aantal leerstappen gedraagt het systeem zich grotendeels lineair ten opzichte van de bewerkingstijd.
De overige stappen in het ACR-WEBSOM-algoritme vergen weinig tijd. In totaal zijn deze stappen in minder dan vijf minuten voltooid.

7.3.7 Optimale waarden voor de parameters

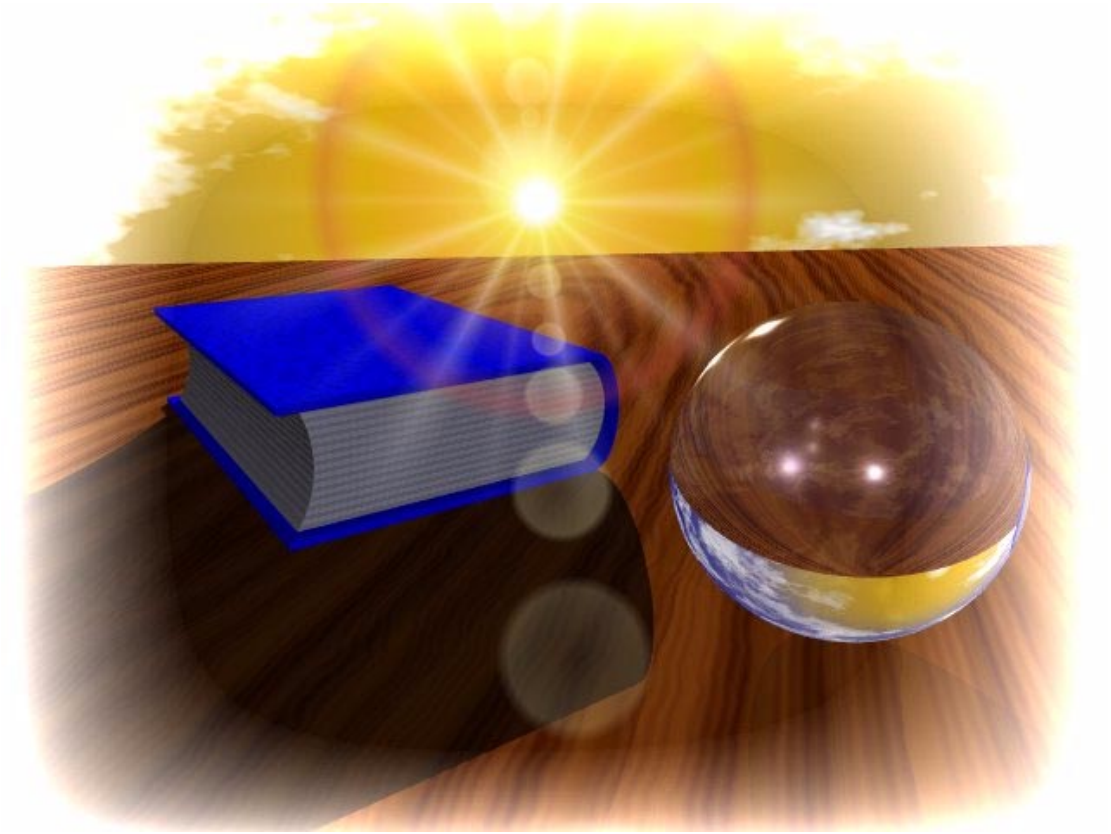
Het feit dat de prestaties van het ACR-WEBSOM-algoritme moeilijk zijn te kwantificeren leidt ertoe dat een optimale waarde voor de verschillende parameters moeilijk is te bepalen. Voor iedere parameter die hieronder behandeld wordt zijn enkele experimenten uitgevoerd om een indicatie te krijgen van de optimale waarde van deze parameter. Bij deze experimenten werd volstaan met een visuele inspectie van de woord-cluster-afbeeldingen en de document-cluster-afbeeldingen die verder niet beschreven zullen worden.
Het doel van deze paragraaf is dan ook niet het vaststellen van de precieze optimale waarden van de parameters, maar eerder het verkennen van de grenzen waarbinnen het ACR-algoritme functioneert.

Dimensionaliteit van de conceptuele ruimte
Het optimale aantal dimensies is afhankelijk van het kennisdomein dat moet worden gerepresenteerd. Voor de documenten uit de testset bleek dat 2 of minder dimensies duidelijk een slechtere ordening tot gevolg had. De meeste experimenten zijn uitgevoerd met een conceptuele ruimte van vijf dimensies, wat goede resultaten tot gevolg had.

Aantal epochs
Na 50 leestappen is de ordening nog niet voltooid, zoals onder andere blijkt uit de test van paragraaf 7.3.3. Tussen de ordening na 100 stappen en na 250 stappen is echter weinig verschil gevonden, wat erop wijst dat de optimale waarde rond de 100 epochs ligt.

Learingrate
Zoals uit paragraaf 7.3.3 blijkt leiden hogere startwaarden tot een snellere ordening. Uit experimenten blijkt tevens dat een minimum waarde van 2/15 tot een betere ordening leidt dan een waarde van 2/10 of 2/5, alhoewel de verschillen minimaal waren en niet altijd eenduidig.

Minimale woordfrequentie
Een groot gedeelte van de woorden uit de testset komen slechts één keer voor en het negeren van deze woorden leidt tot een grote reductie in de tijdscomplexiteit en heeft weinig gevolgen voor de ordening. Ook bij een minimale woordfrequentie van 3 tot 5 is nog sprake van een duidelijk ordening, alhoewel bij een waarde van 5 een duidelijke beginnende aftakeling van de ordening te onderkennen is. Bij een minimale woordfrequentie van 8 zijn er al 3 van de 40 documenten waarvan geen enkele term meer voorkomt in de tekstketen, zodat deze documenten niet meer geordend kunnen worden.



Hoofdstuk 8: Evaluatie, suggesties, conclusies

“It is not the facts but the relation of things that results in the universal harmony that is the sole objective reality.” (Pirsig, p.241)

Dit hoofdstuk vormt de afsluiting van deze scriptie. In deze scriptie is een nieuw algoritme geïntroduceerd waarvan enkele aspecten zijn getest. Dit algoritme zal worden geëvalueerd (8.1), waarna enkele suggesties voor toekomstig onderzoek zullen worden gegeven (8.2). Als laatste volgen enkele definitieve conclusies welke voortvloeien uit deze scriptie (8.3).

8.1 Evaluatie

Uit de experimenten van het vorige hoofdstuk blijkt dat het ACR-WEBSOM-algoritme correct functioneert in die zin dat het in staat is documenten semantisch te ordenen. Hiervoor zijn enkele hypothesen geverifieerd welke noodzakelijk zijn voor het goed functioneren van het algoritme. Zo is aangetoond dat het ACR-algoritme in staat is associatieve woordrelaties af te leiden uit de indexen van boeken en vervolgens de woorden

op basis van deze relaties te ordenen. Deze ordening is verder consistent gebleken vanuit verschillende willekeurige beginposities.
Ook is aangetoond dat hogere beginwaarden voor de learning-rate sneller leiden tot een goede ordening vanuit deze beginposities.
Hiernaast is gebleken dat het actief vergeten een onmisbaar deel is van het ACR-algoritme. Het probleem is echter dat het uitvoeren van de vergeetregel een groot aantal berekeningen vereist, waardoor het ACR-WEBSOM-algoritme bijzonder veel processortijd nodig heeft om tot een goede clustering te komen. Hierbij kan echter wel worden opgemerkt dat de vergeetregel zich bijzonder goed leent om parallel te worden uitgevoerd en implementatie van het ACR-algoritme op een parallelle computer zal dan ook bijzonder effectief zijn. Indien bijvoorbeeld het algoritme uit zou worden gevoerd op een massief parallelle CNAPS-computer met 25.000 eenvoudige parallelle processors zou het theoretisch mogelijk zijn het algoritme bijna 25.000 maal sneller te laten verlopen.
Voor enkele parameters is gezocht naar een benadering van een optimale waarde. Hierbij is echter voorbijgegaan aan het feit dat veel parameters afhankelijk zijn van de samenstelling van de documentencollectie, zoals de diversiteit van de documenten qua onderwerpen en de variatie in gemiddelde woordfrequentie. Zolang echter een eenduidige en precieze methode ter evaluatie van de ordening van woord-cluster-afbeelding en document-cluster-afbeelding ontbreekt is het heel lastig om deze parameters verder te verfijnen.

De experimenten in deze scriptie behelzen allemaal interne testen: zij controleren de werking van de componenten van het systeem op juistheid. Wat echter noodzakelijk is voor zowel het ACR-WEBSOM-algoritme als het WEBSOM-algoritme en de Aqua-browser zijn externe testen: de systemen dienen te worden geïmplementeerd zodat (toekomstige) gebruikers de werking van het systeem kunnen controleren bij het uitvoeren van de hoofdtaak van de systemen: het vinden van relevante documenten. Pas dan is het mogelijk zinnige uitspraken te doen omtrent de precisie en vindkracht van deze systemen.
Met behulp van de zes criteria die in paragraaf 4.1 zijn behandeld is het desalniettemin enigszins mogelijk de prestaties van het ACR-WEBSOM-algoritme te evalueren:

1. Het **bereik van de collectie**: Het ACR-WEBSOM-algoritme is in staat documenten te ordenen aan de hand van hun indexen. Hierbij wordt het bereik logischerwijs beperkt tot boeken met een index. Dit systeem kan worden gezien als complementair aan andere systemen zoals het WEBSOM-algoritme.
2. De **vertraging**: Aangezien het vormen van de document-cluster-afbeelding al van te voren gebeurt is de vertraging tussen de initiatie van zoekactie door de gebruiker en het genereren van het resultaat door de computer zeer klein.
3. De **vorm van de presentatie** van de output: Het ACR-WEBSOM-algoritme heeft dezelfde grafische gebruiksinterface als de WEBSOM waarmee de gebruiker gemakkelijk kan zoeken.
4. De **moeite** die de gebruiker moet doen om de antwoorden te krijgen van het systeem: Hiervoor geldt hetzelfde als is gesteld voor het WEBSOM-algoritme in paragraaf 5.1.4: Door de unieke gebruikersinterface is het mogelijk voor de gebruiker om slechts met enkele klikken van de muis de voor haar/hem interessante documenten te vinden. Aangezien de gebruiker kan kiezen uit een aantal van te voren zichtbare clusters is het voor de gebruiker ook mogelijk te zoeken zonder een duidelijke informatiebehoefte. De gebruiker kan dan die cluster uitkiezen die het meest relevant lijkt en later een meer gedetailleerde ‘query’ geven, gebaseerd op de clusters die dan verschijnen.
5. De **vindkracht** van het systeem: Zoals gezegd is het noodzakelijk dat het algoritme zorgvuldig extern getest wordt voordat een onderbouwde uitspraak kan worden gedaan

over de vindkracht of precisie. Natuurlijk is het de bedoeling dat, door gebruik te maken van kennis over de associaties tussen woorden, het algoritme in staat is een betere vindkracht te leveren dan traditionele informatie-zoeksystemen. Documenten die niet dezelfde termen bevatten maar wel sterk geassocieerde termen zullen door het systeem als gelijk worden beschouwd, zodat minder relevante documenten worden gemist.

6. De **precisie** van het systeem: Ook de precisie moet nog door een praktijktest worden bepaald. Aangezien alle documenten in een cluster op de document-cluster-afbeelding ongeveer dezelfde onderwerpen bevatten is het aannemelijk dat, zodra één relevant document is gevonden, de overige documenten in de cluster ook relevant zijn en het algoritme dus een hoge precisie behaalt.

8.2 Suggesties

Ten aanzien van de bevindingen van deze scriptie kunnen de volgende suggesties worden gegeven voor verder onderzoek:

- **Reduceren van de tijdscomplexiteit**
  Zoals vermeld is het grootste probleem van het ACR-WEBSOM-algoritme momenteel de tijdscomplexiteit, en dan met name van de vergeetregel. Er is echter nog geen poging gedaan deze regel te optimaliseren. Indien toekomstig onderzoek een methode levert waardoor deze tijdscomplexiteit kan worden gereduceerd is de haalbaarheid van een praktische toepassing van het ACR-WEBSOM-algoritme sterk verbeterd.

- **Verificatie van de overige hypothesen**
  Bij het ontwerpen van het ACR-WEBSOM-algoritme zijn een groot aantal keuzen gemaakt die niet direct onderbouwd zijn. Enkele van deze keuzen zijn vermeldt in paragraaf 7.3. Misschien dat andere keuzen een betere werking van het algoritme tot gevolg hebben.

- **Sterkere associatie indien de termen op één pagina voorkomen**
  Eén van de keuzen die verder niet getoetst is, is het feit dat bij het afleiden van de associatie van woorden uit de tekstketen T_* uitsluitend wordt gekeken naar de volgorde waarin de woorden in het document voorkomen. Hierbij wordt echter voorbijgegaan aan de afstand tussen de termen, gemeten in pagina's. Indien ook deze informatie wordt gebruikt zou een betere ordening van woorden kunnen ontstaan.

- **Onderzoek naar het gebruik van de Aqua-browser bij indexen**
  In het begin van paragraaf 6.2 is gesteld dat, met gebruik van de bewerkte indexen zoals beschreven in paragraaf 6.1, het mogelijk is een Connectionistisch Semantisch Netwerk op te stellen. Indien deze gekoppeld zou worden aan de Aqua-browser ontstaat dan ook een informatie-zoeksysteem dat kennis kan extraheren uit indexen. Toekomstig onderzoek zal uit moeten wijzen of dit een haalbare oplossing is en hoe de prestaties van een dergelijk systeem zich verhouden tot de prestaties van het ACR-WEBSOM-algoritme .
- **Gebruik van het ACR-WEBSOM bij volledige teksten**
  Het ACR-WEBSOM-algoritme maakt gebruik van een index en inverteert deze tot een vorm waarin de tekst zoveel mogelijk is gereconstrueerd aan de hand van de

index. Het zou echter ook mogelijk moeten zijn het ACR-WEBSOM-algoritme toe te passen op volledige tekstdocumenten, mits deze natuurlijk niet van grote omvang zijn. Hierbij zal onder andere een manier moeten worden gevonden om de kernwoorden te selecteren uit de tekst, iets wat bij een index al is gedaan door de auteur.
Het voordeel van de mogelijkheid om ook kennis te kunnen extraheren uit volledige tekst-documenten is dat het bereik van het ACR-WEBSOM-algoritme zodanig wordt uitgebreid dat zowel kleine documenten als grote documenten met index in één document-cluster-afbeelding kunnen worden verwerkt. Op deze manier kan de gebruiker beide delen van de collectie tegelijk doorzoeken en kan tijd en moeite worden bespaard.

- **Betere evaluatiemethode voor semantische ordening**
  In hoofdstuk 7 kwam naar voren dat de kwaliteit van de ordening van documenten en woorden moeilijk te kwantificeren is. Dit vraagt om een oplossing, zodat niet alleen het ACR-WEBSOM-algoritme maar ook bijvoorbeeld het WEBSOM-algoritme beter kan worden geëvalueerd en geoptimaliseerd.

- **Externe tests**
  Zoals gezegd is het noodzakelijk dat externe tests worden uitgevoerd op het ACR-WEBSOM-algoritme voordat een duidelijk beeld ontstaat van de precisie en de vindkracht.

- **Andere toepassingen voor het ACR-algoritme**
  In deze scriptie is het ACR-algoritme ontwikkeld ten behoeve van een informatie-zoeksysteem. Dit algoritme is echter uniek in de ordening die het geeft en vormt een nieuwe manier van kennisrepresentatie. Het is niet ondenkbaar dat Associatieve Conceptuele Ruimte op andere manieren binnen informatie-zoeksystemen kan worden gebruikt of zelfs in een ander vakgebied kan worden toegepast, zoals bijvoorbeeld semantic priming. Semantic priming is dat deelgebied van natural language processing dat zich bezig houdt met de woorden die mensen verwachten te horen/lezen binnen een bepaalde context en kan onder andere worden gebruikt voor de verbetering van volledige stemherkenning. Onderzoek hiernaar is gewenst.

8.3 Conclusies

In deze scriptie is een nieuwe generatie informatie-zoeksystemen aan het licht gekomen. Deze systemen zijn in staat vanuit woorden te abstraheren naar hoger liggende concepten, om op deze wijze een document en een query op een hoger conceptueel niveau te vergelijken, daarbij niet meer gehinderd door individuele details zoals woordkeuze.
Het ACR-WEBSOM-algoritme construeert dit hogere abstractieniveau op een wijze die analoog is aan de manier waarop volgens Hebb de menselijke hersenen een hoger abstractieniveau afleiden uit gegevens van de zintuigen zoals beschreven in paragraaf 2.2.3: “When A, B and C are looked at successively, in any order, but in a short period of time, activity may continue by reverberation in two of the structures while the third is sensorily aroused.... According to the assumptions made earlier, simultaneous activity in a, b, and c would establish facilitation between them....The resulting superordinate system must be essentially a new one, by no means a sum or hooking together of a, b and c.” (Hebb, p.96,97)

In termen van het ACR-WEBSOM-algoritme houdt dit in dat voor woorden die elkaar vaak opvolgen in de bewerkte index een geheel nieuwe structuur ontstaat. Dit is te herkennen in de clusters in de woord-cluster-afbeelding, waar geassocieerde woorden zijn gegroepeerd onder één cluster; één overkoepelend ‘concept’. Wat dit algoritme doet is dus te zien als het omhoog klimmen in de conceptuele hiërarchie van Aristoteles, van individuen in de vorm van woorden naar bovenliggende concepten.
Afgezien van deze filosofische reflecties blijkt dat het ACR-WEBSOM-algoritme gewoon werkt. Het is mogelijk een conceptuele ruimte op te bouwen aan de hand van associaties tussen woorden en deze associaties kunnen afgeleid worden uit indexen van boeken. Er zal echter nog veel onderzoek moeten gebeuren voordat het in de praktijk is toe te passen.

Bijlage 1: Programmacode van het ACR-WEBSOM-algoritme

```
unit ACRWEBSOM10; {Version 1.0}

interface

uses SysUtils;

const
  MaxLexicon = 13000;        {number of different words in the documents}
  MaxStringSize = 33000;     {number of words in the documents}
  Dimensions = 5;            {dimensions of the 'conceptual space'}
  HoodSize   = 9;            {size of the neighborhood}
  HoodFocus  = 5;            {node in the neighborhood that is the focus. usually the middle
                              one}
  NumberOfEpochs = 250;      {Number of trainingsteps}
  LRParameter = 10;          {1/LRParameter is the minimum learning-rate}
  LowerThreshold = 2;        {words must occur more often than this in the documents}

  InputFile  = 'C:\scriptie\PreProces\DataDestination\pdata';
  NumberOfFiles = 40;        {Number of index-files}

  OutputFile = 'C:\scriptie\semtex\t051.txt';

type
  TTextWord = string[25];

  TLexicalItem = record
    TextWord  : TTextword;
    Frequency : integer;
  end;

  TLexicon = array[0..MaxLexicon] of TLexicalItem;

  TConnection = record
    Weight : real;
  end;

  TNode = array [1..Dimensions] of TConnection;

  TNet = array [1..MaxLexicon] of TNode;

  TWordString = array[1..MaxStringSize] of word;
```

```
 TMain = object
   public

   procedure Run;                  {Starts the entire process}

   private

   Epoch               : integer;
   LearningRate        : real;
   Lexicon             : TLexicon;    {vocabulary}
   Net                 : TNet;
   WordString          : TWordString; {array containing references to words in lexicon}
   WordsInLexicon,
   WordsInWordString : integer;

   procedure Initialize;
   procedure LexicalAnalysis;
   procedure PreProcessing;
   procedure SemanticAnalysis;
   procedure SaveResults;

   procedure Connect(Node1,Node2 : integer; Strength:real);
   procedure Forget;
   function RepulseFunction(d:real):real;
   function Min(a,b:integer):integer;                {Returns minimum of a and b}
   function ReadWord(var TF: text) : TTextWord; {Read next word in textfile}
   function Find(S:TTextword):integer;               {Find word in lexicon. Returns}
  end;                                               {-1 if not found          }

var
  main : TMain;

implementation

function TMain.Min(a,b:integer):integer;
begin
  if a<b then
    Min := a
  else
    Min := b;
end;

Function TMain.ReadWord(var TF: text) : TTextWord;
var
  EndOfWord  : Boolean;
  Ch         : Char;
  S          : TTextWord;
begin
  Repeat
```

```
    S := '';
    EndOfWord := False;
    Repeat
      Read(TF, Ch);
      Ch:= upcase(CH);                    {All text in upper case.     }
      if (ord(Ch)>64) and (ord(Ch)<91) then {It is a letter of the      }
        S := S + Ch                       {alphabet.                   }
      else
      EndOfWord := True;                  {else it is the end of the word}
    until EndOfWord;
  until (S<>'') or EOF(TF);               {Until a word is found       }
  ReadWord := S;
end;

function TMain.Find(S:TTextword):integer;{returns -1 if not found}
var
  I,F :integer;
begin
  F := -1;
  i:=0;
  while i<=WordsInLexicon do
  begin
    if Lexicon[i].TextWord = s then
    begin
      F:=i;
      i:=WordsInLexicon;
    end;
    i:=i+1;
  end;
  Find:=F;
end;

function TMain.RepulseFunction(d:real):real;
const
  Threshold = 1;
begin
  if d < Threshold then  RepulseFunction := 1
  else  RepulseFunction := 1/d;
end;

procedure TMain.Initialize; {Initialize values}
var
  i,j : integer;
begin
  Epoch := 0;
  WordsInLexicon := 0;
  WordsInWordString := 0;
  randomize;
  for i := 1 to MaxLexicon do
```

```
    for j := 1 to Dimensions do Net[i,j].Weight :=random(1000)/100;
end;

procedure TMain.LexicalAnalysis; {Reading index-files and building lexicon}
var
  TextFile                    : text;
  Item                        : TTextWord;
  WordPtr, FileNumber         : Integer;
begin
  for FileNumber := 1 to NumberOfFiles do
  begin
    Assign(TextFile,InputFile + IntToStr(FileNumber) + '.txt');
    Reset(TextFile);
    FileMode := 0;          {ReadOnly}
    while not EOF(Textfile) do
    begin
      Item := ReadWord(TextFile);
      WordPtr:=Find(Item);
      if WordPtr>-1 then     {Word is found in lexicon}
        Lexicon[WordPtr].Frequency:= Lexicon[WordPtr].Frequency +1;
      if (WordPtr=-1) and ((WordsInLexicon+1) < MaxLexicon) then
      with Lexicon[WordsInLexicon+1] do {Add word to lexicon}
      begin
        TextWord := Item;
        Frequency := 1;
        WordsInLexicon := WordsInLexicon + 1;
        WordPtr := WordsInLexicon;
      end;
      if WordsInWordString < MaxStringSize then
      begin
        WordsInWordString := WordsInWordString + 1;
        WordString[WordsInWordString] := WordPtr;
      end;
    end;
    Close(TextFile);
  end; {of for loop}
end;

procedure TMain.PreProcessing; {words that do not occur often enough are deleted}
var
  i,LCount,SCount : integer;
  TLex        : array [0..MaxLexicon] of integer;       {Temporary lexicon reference to new}
                                                         lexicon          }
begin
  LCount := 0;
  for i := 0 to WordsInLexicon do
  if Lexicon[i].Frequency > LowerThreshold then
  begin
    TLex[i] := LCount;   {This lexical item will return on position LCount}
```

```
    LCount := LCount + 1;
  end
  else
  begin
    TLex[i] := -1;     {This lexical item will not return in the new lexicon}
  end;
  SCount := 1;
  for i := 1 to WordsInWordString do
    if TLex[WordString[i]] >= 0 then
    begin
      WordString[SCount] := TLex[WordString[i]];     {This word will return in string and also
                                                      gets new reference in lexicon}
      SCount := SCount + 1;
    end;
  for i := 0 to WordsInLexicon do
    if TLex[i] >= 0 then
    begin
      Lexicon[TLex[i]].TextWord := Lexicon[i].TextWord;
      Lexicon[TLex[i]].Frequency := Lexicon[i].Frequency;
    end;
  WordsInLexicon := LCount;
  WordsInWordString := SCount;
end;

procedure TMain.Connect(Node1,Node2 : integer);
var
  vector          : array [1..Dimensions] of real;
  distance,e      : real;
  i               : integer;
begin
  if Node1<>Node2 then
  begin
    distance := 0;
    for i := 1 to Dimensions do
    begin
      vector[i] := Net[Node1,i].Weight - Net[Node2,i].Weight;
      distance := distance + sqr(vector[i]);
    end;
    distance := sqrt( distance);  {'distance' now contains length of vector}
    for i := 1 to Dimensions do   {normalization}
    begin
      if distance = 0 then
        vector[i] := random
      else
        vector[i]:= vector[i]/(distance);  {vector is normalized}
      vector[i] := vector[i]* 2 *(WordsInWordString/WordsInLexicon)/
        (lexicon[node1].Frequency + lexicon[node1].Frequency);

      {get nodes closer:}
```

```
      net[node1,i].Weight := net[node1,i].Weight - LearningRate * vector[i];
      net[node2,i].Weight := net[node2,i].Weight + LearningRate * vector[i];
    end; {end for-loop}
  end;  {endif}
end;

procedure TMain.Forget; {Increase distance between all nodes}
var
  i,j,k    : integer;
  vector   : array[1..dimensions] of real;
  distance : real;
begin
  for i := 1 to WordsInLexicon do
    for j := i+1 to WordsInLexicon do
    begin
      distance := 0;
      for k := 1 to dimensions do
      begin
        vector[k] := net[i,k].weight - net[j,k].weight;
        distance := distance + sqr(vector[k]);
      end;
      distance := sqrt(distance);  {distance now contains euclidian distance}
      for k := 1 to dimensions do
      begin
        if distance = 0 then
          vector[k] := random
        else
          vector[k]:= vector[k]/(distance);  {vector is normalized}
        net[i,k].weight := net[i,k].weight + vector[k] * RepulseFunction(distance);
        net[j,k].weight := net[j,k].weight - vector[k] * RepulseFunction(distance);
      end;
    end;
end;
```

```
procedure TMain.SemanticAnalysis; {Adjusting of reference vectors}
var
  Hood          : Array[1..HoodSize] of integer;
  i,j           : Integer;
  WordPtr       : Integer;
  Start         : TDateTime;

begin
  for i:= 1 to HoodSize do Hood[i]:=0;
  for j:= 1 to WordsInWordString do
  begin
    WordPtr:=WordString[j];
    for i := 1 to HoodSize-1 do Hood[i]:=Hood[i+1];
    Hood[HoodSize] := WordPtr;        {Hood now contains <HoodSize> sequential words}

    for i := 1 to HoodSize do {Strengthen connection between nodes in hood}
      Connect(Hood[i],Hood[HoodFocus],HoodFunction(i,HoodFocus));
  end;
  Forget;       {Weaken connections between all nodes}
end;

procedure TMain.SaveResults;
var
  textfile : text;
  i,j : integer;
begin
  AssignFile(TextFile,OutputFile);
  Rewrite(TextFile);
  Writeln(TextFile,IntToStr(Dimensions));
  for i := 1 to WordsInLexicon do
  begin
    for j:= 1 to Dimensions do write(Textfile,Net[i,j].Weight:10:4);
    write(Textfile,' "'+Lexicon[i].textword+'"');
    writeln(Textfile,''); {EndOfLine}
  end;
  CloseFile(Textfile);
end;

procedure TMain.Run;
begin
  Initialize;
  LexicalAnalysis;
  PreProcessing;
  repeat
    Epoch := Epoch +1;
    LearningRate := (2/min(Epoch,LRParameter));
    SemanticAnalysis;
  until Epoch = NumberOfEpochs;
  SaveResults;
```

```
end;

end.
```

Referenties:

(Alexander 1997) M. Alexander, "Retrieving Digital Data With Fuzzy Matching", in *"Managing Information"*, Jan/Feb 1997, p.34

(Amerongen 1997) Wim Amerongen, "Prijsval geeft data warehousing enorme impuls", in *"Informatie Management"*, juni/juli 1997, p. 8

(Aristoteles) Aristoteles, "*The complete works of Aristotle* ", edited by Jonathan Barnes, Vol.2, Princeton University Press, 1984

(Bemelmans) Prof. Dr. T.M.A. Bemelmans*, "Bestuurlijke informatiesystemen en automatisering*", Kluwer 1994

(Bioch) Dr. J.C. Bioch, handout bij het *vak "Kennissystemen en Neurale Netwerken*", E.U.R., 1996

(Bloem 1997) J. Bloem, "Liquid Solutions, meerwaarde à la Origin Medialab", in *"Informatie"*, oktober 1997, p.12-15,19

(Charniak e.a.) E.Charniak, D.McDermott, "*Introduction to Artificial Intelligence*", Addison - Wesley Publishing Company, 1985

(Churchland) P.Churchland, "*the Engine of Reason, the Seat of the Soul*", MIT Press, 1996

(Coveney e.a.) P. Coveney, R. Highfield*, "Frontiers of Complexity, The Search for Order in a Chaotic World*", Faber and Faber, 1996

(Crestani) F. Crestani, "Application of Spreading Activation Techniques in Information Retrieval", in "*Artificial Intelligence Review"*, vol.11, Nr.6, December 1997, p.453

(Crick) F. Crick, "*the Astonishing Hypothesis: the Scientific Search for the Soul*", Macmillan Publishing Company, 1994

(Elman) J.L. Elman, "Grammatical Structure and Distributed Representation", in "*Connectionism: Theory and Practice",* vol. 3 in the series Vancouver Studies in Cognitive Science, Oxford University Press, 1992

(Ferber) R.Ferber, "*Information Retrieval*", GMD-IPSI, http://www.darmstadt.gmd.de/~ferber/ir-bb/frame.html

(Fu) L. Fu, "*Neural Networks in Computer Intelligence*", McGraw-Hill, 1994

(Gaarder) J.Gaarder, *"de Wereld van Sofie: roman over de geschiedenis van de filosofie*", Houtekiet/Fontein, 1994

(Gelder, van) T. van Gelder, "Making Conceptual Space", *in "Connectionism: Theory*

and Practice", vol. 3 in the series Vancouver Studies in Cognitive Science, Oxford University Press, 1992

(Hebb) D.O. Hebb*, "The Organization of Behavior: A Neuropsychological Theory*", John Wiley & Sons, Inc., 1966

(Hertz e.a.) J. Hertz, A. Kroch, R.G.Palmer, "*Introduction to the theory of neural computation*", Addison-Wesley Publishing Company, 1993

(Honkela e.a., 1995) T. Honkela, V. Pulkki, T. Kohonen, " Contextual Relations of Words in Grimm Tales, Analyzed by Self- Organizing Map", *in "Proceedings of International Conference on Artificial Neural Networks*", EC2 et Cie, Paris, 1995, p.3

(Honkela e.a., 1996) T. Honkela, S. Kaski, K. Lagus and T. Kohonen, "*Newsgroup Exploration with WEBSOM Method and Browing Interface*", TKK Offset 1996

(Imai e.a) M. Imai, D. Genter, "A cross-linguistic study of early word meaning: universal ontology and linguistic influence", in "*Cognition"*, nr.2, februari 1997, p.169

(Johnston) D. Johnston, "A Missing Link? LTP (long- term potentiation) and learning", in "*Science"*, vol. 278, 17 oktober 1997, p.401

(Kaski e.a. 1996) S. Kaski, T.Honkela, K. Lagus and T. Kohonen, "Creating an Order in Digital Libraries with Self-Organizing Maps*",* in "*Proc. World Congress on Neural Networks '96"*, p.814-817, Lawrence Erlbaum and INNS Press, NJ, 1996

(Kohonen) T. Kohonen, "*Self-Organizing Maps*", Springer 1995

(Kohonen e.a.1) T. Kohonen, S. Kaski, K. Lagus, T. Honkela*, "Self- Organizing Maps of document collections*" http://www.diemme.it/~luigi/websom.html

(Kohonen e.a.2) T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, "*SOM_PAK: The Self-Organizing Program Package*", Helsinki University of Technology, January 1996, http://www.cis.hut.fi/nnrc/nnrc-programs.html

(Lagus) K. Lagus, S. Kaski, T.Kohonen, T. Honkela*, "Exploration of Full-text Databases with Self-Organizing Maps*" Submitted to ICNN-96. Washington D.C.

(Palmer) F.R. Palmer, "*Semantics*", Cambridge University Press, Cambridge, 1981

(Pirsig) R.M. Pirsig, "*Zen and the Art of Motorcycle Maintenance*", Bantam Books, 1974

(Plato) Plato, "*Constitutie, Politeia*", vertaald door G. Koolschijn, Athenaeum Polak & Van Gennep, Amsterdam 1991

(Plaut) D.C. Plaut, "Semantic and Associative Priming in a Distributed Attractor Network", in "*Proceedings of the 17th Annual Conference of the Cognitive Science Society*", p.37-42, Hillsdale, NJ: Lawrence Erlbaum Associates

(Popper) Sir. K.R. Popper, "*Knowledge and the Body-Mind Problem, in defence of interaction*", Routledge, 1994

(Quillian) M.R. Quillian, "Semantic Memory", *in "Semantic Information Processing*", MIT press, 1968

(Radnitzky e.a.) G. Radnitzky, W.W. Barlley III,"*Evolutionary Epistemology, Rationality and the Sociology of Knowledge*", Open Court, 1987

(Rijsbergen) C. J. van Rijsbergen, "*Information Retrieval*", London: Butterworths, 1979, http://jabato.unizar.es/infordoc/rijs/Preface.html

(Schäuble) P. Schäuble, "Thesaurus Based Concept Spaces", in *"Proceedings of the Tenth Annual International ACMSIGIT Conference on Research & Development in Information Retrieval*", ACM Press, New Orleans, 1987

(Scholtes) J.C. Scholtes, "*Neural Networks in Natural Language Processing and Information Retrieval*" PhD Thesis, University of Amsterdam, 1993

(Shastri) L.Shastri, "*Semantic Networks: An Evidential Formalization and its Connectionist Realization*", Pitman Publishing, 1988

(Steels) L.Steels, "*Kennissystemen*", Addison- Wesley Publishing Company, 1992

(Veling) W.A. Veling, "The Aqua Browser: Visualisation of large information spaces in context", in "*AGSI Journal"*, November 1997, Volume 6, Issue 3, page 136-142

(Weustink) N. Weustink, "*INDEX-SOM: Een Informatie-Zoeksysteem gebaseerd op Self-Organizing Maps*", doctoraal scriptie E.U.R., 1997