NEURALE NETWERKE AS MOONTLIKE WOORDAFKAPPINGSTEGNIEK VIR AFRIKAANS

M. FICK

NEURALE NETWERKE AS MOONTLIKE WOORDAFKAPPINGSTEGNIEK VIR AFRIKAANS

deur

MACHTELD FICK

voorgelê ter gedeeltelike vervulling van die vereistes vir die graad

MAGISTER SCIENCIÆ

in die vak

OPERASIONELE NAVORSING

aan die

UNIVERSITEIT VAN SUID-AFRIKA

STUDIELEIER: PROF C J SWANEPOEL

SEPTEMBER 2002

****

Opsomming

In Afrikaans, soos in Nederlands en Duits, word saamgestelde woorde aanmekaar geskryf. Nuwe woorde word dus voortdurend geskep deur woorde aanmekaar te haak. Dit bemoeilik die proses van woordafkapping tydens teksprosessering, wat deesdae deur rekenaars gedoen word, aangesien die verwysingsbron gedurig verander. Daar bestaan verskeie afkappingsalgoritmes en tegnieke, maar die resultate is onbevredigend. Afrikaanse woorde met korrekte lettergreepverdeling is uit die elektroniese weergawe van die *Handwoordeboek van die Afrikaanse Taal* (HAT) onttrek. 'n Neurale netwerk (vorentoevoer-terugpropagering) is met sowat 5 000 van hierdie woorde afgerig. Die neurale netwerk is verfyn deur 'n geskikte afrigtingsalgoritme en oordragfunksie vir die probleem asook die optimale aantal verborge lae en aantal neurone in elke laag te bepaal. Die neurale netwerk is met 5 000 nuwe woorde getoets en dit het 97,56% van moontlike posisies korrek as òf geldige òf ongeldige afkappingsposisies geklassifiseer. Verder is 510 woorde uit tydskrifartikels met die neurale netwerk getoets en 98,75% van moontlike posisies is korrek geklassifiseer.

Summary

In Afrikaans, like in Dutch and German, compound words are written as one word. New words are therefore created by simply joining words. Word hyphenation during typesetting by computer is a problem, because the source of reference changes all the time. Several algorithms and techniques for hyphenation exist, but results are not satisfactory. Afrikaans words with correct syllabification were extracted from the electronic version of the *Handwoordeboek van die Afrikaanse Taal* (HAT). A neural network (feedforward backpropagation) was trained with about 5 000 of these words. The neural network was refined by heuristically finding a suitable training algorithm and transfer function for the problem as well as determining the optimal number of layers and number of neurons in each layer. The neural network was tested with 5 000 words not the training data. It classified 97,56% of possible points in these words correctly as either valid or invalid hyphenation points. Furthermore, 510 words from articles in a magazine were tested with the neural network and 98,75% of possible positions were classified correctly.

[Key words: Neural networks; Backpropagation; Feed-forward; Training algorithm; Transfer function; Resilient backpropagation; Early termination; Encoding; Hyphenation; Syllabification.]

Verklaring

Ek verklaar hiermee dat Neurale Netwerke as moontlike Woordafkappingstegniek vir Afrikaans my eie werk is en dat ek alle bronne wat ek gebruik of aangehaal het deur middel van volledige verwysings aangedui en erken het.

Inhoudsopgawe

1 Inleiding 1

1.1 Hoekom is afkapping nodig? . . . 2

1.2 Afkapping in verskillende tale . . . 5

1.2.1 Engels (Verenigde Koninkryk) . . . 5

1.2.2 Amerikaanse Engels . . . 6

1.2.3 Nederlands . . . 7

1.2.4 Afrikaans . . . 8

1.3 Rekenaarafkapping . . . 9

1.3.1 Hyphenologist . . . 10

1.3.2 Dashes . . . 11

1.3.3 SiSiSi . . . 13

1.3.4 Afkapping in TEX . . . 15

2 Neurale netwerke 23

2.1 Inleiding . . . 23

2.2 Argitektuur . . . 26

2.3 'n Neurale netwerk met 'n enkele neuron . . . 26

2.4 'n Neuronlaag . . . 28

2.5 Meervoudige neuronlae . . . 30

2.6 Neurale-netwerkmodelle . . . 31

2.6.1 McCulloch-Pitts . . . 31

2.6.2 Perseptron . . . 33

2.6.3 ADALINE . . . 36

2.6.4 Perseptrone met meervoudige lae . . . 39

2.7 Terugpropagering . . . 40

2.8 Ontwikkeling om terugpropagering te verbeter . . . 42

2.8.1 Lotafrigting (*batch training*) . . . 42

2.8.2 Momentum . . . 42
2.8.3 Veranderlike leertempo . . . 42
2.8.4 Veerkragtige terugpropagering (*Resilient backpropagation* – RP) . . . 43
2.8.5 Verwante-hellingalgoritmes (*Conjugate gradient*) . . . 43
2.8.6 Vroeë beëindiging . . . 44
2.9 Neurale netwerke wat afkapping doen . . . 45
2.9.1 Søren Brunak en Benny Lautrup . . . 45
2.9.2 Pavel Smerž en Petr Sojka . . . 48
2.9.3 Bernd Fritzke . . . 51
2.9.4 Walter Daelemans en Antal van den Bosch . . . 54

3 Ontwikkeling van neurale netwerk **59**
3.1 Lettergreepverdeling teenoor afkapping . . . 59
3.2 Idees uit vorige navorsing . . . 59
3.3 Verkryging van data . . . 60
3.4 Voorbereiding van invoer . . . 60
3.4.1 'n Verkorte alfabet . . . 60
3.4.2 Die volledige alfabet . . . 64
3.5 Die neurale netwerk . . . 68
3.5.1 Afrigting . . . 68
3.5.2 Evaluering van die neurale netwerk se prestasie . . . 70
3.5.3 Bepaal die beste neurale netwerk vir die probleem . . . 71
3.5.4 Om die neurale netwerk te gebruik . . . 78
3.5.5 Verdere verbetering van afrigtingsdata . . . 82
3.6 *Goeie* en *slegte* afkappingsfoute . . . 86
3.7 Argitektuur van die netwerk . . . 88
3.8 Gevolgtrekking . . . 89

A Data **93**
A.1 Die bron van data wat gebruik word . . . 93
A.2 'n Volledige datastel . . . 94
A.2.1 Woorde met lettergreepverdelings . . . 94
A.2.2 Kort woorde . . . 98
A.2.3 Deelversamelings . . . 99

Voorwoord

Programmatuur wat vir rekenaarteksprosessering gebruik word, bevat gewoonlik ingeboude afkappingsfunksies. Sulke funksies verdeel woorde wat nie aan die einde van 'n gedrukte reël inpas nie, en dui met 'n koppelteken aan dat die woord in die volgende reël voltooi word. Bestaande afkappingsfunksies is gewoonlik op 'n woordeboekbenadering of op 'n reëlbenadering of op 'n kombinasie van die twee gebaseer. Dit gebruik gewoonlik baie geheue (woordeboekbenadering) en/of is afhanklik van groot uitsonderingslêers (reëlbenadering). Die doel van afkapping is hoofsaaklik om visuele reëlmatigheid in gedrukte teks te bevorder.

In tale soos Nederlands, Afrikaans en Duits is afkapping 'n groot probleem, hoofsaaklik as gevolg van die feit dat nuwe woorde na willekeur geskep kan word deur woorde aanmekaar te haak. Waar 'n moedertaalspreker natuurlike afkappingsposisies in saamgestelde woorde outomaties herken, is dit vir 'n rekenaar onmoontlik om die konteks van saamgestelde woorde te ontleed en die afkapping altyd korrek te doen. By sulke tale lewer beide die woordeboekbenadering en reëlgebaseerde algoritmes onbevredigende resultate.

Hierdie navorsingsprojek is daarop gerig om 'n neurale netwerk te ontwikkel om die probleem van afkapping in Afrikaans aan te spreek. Neurale netwerke, wat deur biologiese senuweestelsels geïnspireer is, is by uitstek geskik vir patroonherkenningsprobleme. Aangesien woordafkapping inherent 'n patroonhetkenningsprobleem is, word die moontlikheid ondersoek dat 'n neurale netwerk wat met korrekte Afrikaanse afkappingsvoorbeelde afgerig word, in staat sal wees om afkappingspatrone in onbekende woorde te herken en as bruikbare afkappingsinstrument aangewend kan word.

Hoofstuk 1

Inleiding

> ... you cannot assemble words into effective communication unless you are prepared to break a few of them. But you must do it with skill. The automation of word assembly does not excuse editors and printers from their aesthetic obligation.
>
> Ronald McIntosh and David Fawthrop

Vir 'n moedertaalspreker is dit byna tweede natuur om woorde in lettergrepe te verdeel. Deur 'n woord lettergreep vir lettergreep uit te spreek, word natuurlike breekpunte in 'n woord geïdentifiseer. Hierdie breekpunte is dan ook natuurlike afkappingsposisies in gedrukte teks.

Voordat rekenaars in die drukkersbedryf gebruik is, is lettersetwerk per hand gedoen. Die toestel wat gebruik is, sou die operateur waarsku dat die einde van 'n reël nader kom. 'n Besluit moes dan geneem word oor of 'n woord aan die einde van die reël afgekap moet word of nie en waar die koppelteken moet kom, indien wel.

Die koms van rekenaars het die drukkersbedryf in vele opsigte verander. Daar is nou nie meer 'n operateur wat aan die einde van elke reël 'n besluit kan neem nie. Die rekenaar moet die besluit neem en daarvoor is 'n stel reëls of 'n algoritme nodig om die breekpunte te bepaal.

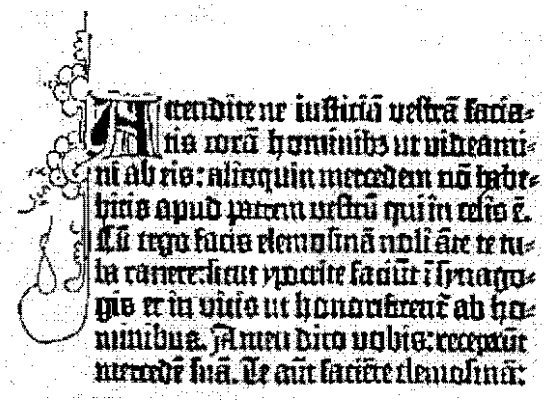
Daar is reeds baie werk gedoen om die moeilike probleem van afkapping aan te spreek. Daar bestaan programme wat afkapping in 'n verskeidenheid van tale suksesvol hanteer. Van hierdie produkte sal later bespreek word. Afkapping in tale soos Duits, Nederlands en Afrikaans, waar nuwe woorde na willekeur gevorm kan word deur twee (of meer) woorde aanmekaar te haak, is egter 'n baie moeilike probleem. Sulke saamgestelde woorde word aanmekaar geskryf en die rekenaar het geen aanduiding van waar die natuurlike breuke is nie.

Aangesien daar oneindig baie saamgestelde woorde in Afrikaans geskep kan word, is dit onmoontlik om 'n databasis met lettergreepverdelings as 'n opsoekbiblioteek te stoor en so die afkappingsprobleem op te los. So 'n databasis sal nooit volledig wees nie, en dit sal te lomp wees om op 'n gewone persoonlike rekenaar te gebruik.

Die doel van hierdie projek is om vas te stel of neurale netwerke 'n bruikbare tegniek is om Afrikaanse woorde in lettergrepe te verdeel. Indien dit moontlik is, kan die proses van afkapping op grond van lettergreepverdeling ontwikkel word.

1.1 Hoekom is afkapping nodig?

Sedert die middeleeue het tipograwe daarna gestrewe om teks in simmetriese blokke te skryf, met gelyke linker- en regterkante (dubbelgeskouerde teks). Hulle het van tegnieke soos afkortings en afkapping gebruik gemaak om woorde korter te maak ten einde visueel bevredigende bladsye te verkry. In Figuur 1.1 verskyn 'n gedeelte uit die drukwerk van Gutenberg [20]. Hy het vroeg in die vyftiende eeu beweegbare, herbruikbare letters uitgevind. Hy het ook die koppelteken as belangrike instrument teruggebring.



Figuur 1.1: Gutenberg se drukwerk in middeleeue

Mense het daaraan gewoond geraak om sulke geskouerde (*justified*) teks te lees en het hewig protes aangeteken toe moderne grafiese ontwerpers vroeg in die 20ste eeu met lettersetstyle begin eksperimenteer het en ruwe regterkante ingevoer het. Alhoewel geskouerde teks 'n rustigheid tot gevolg het, veroorsaak dit dikwels dat daar gebreke in die teks voorkom.

Veral in die smal kolomme van tydskrifte en koerante veroorsaak geskouerde teks die volgende:

- Spasiëringsprobleme soos
  - groot spasies tussen woorde;
  - variërende spasies tussen letters; en
  - riviere van wit spasie en stapels woorde van bo na onder in die teks.
- Verkeerde of lomp afkappings aan die einde van reëls.

Hierdie gebreke het tot gevolg dat die visuele ritme versteur word en dat leesgemak ingeboet word.

Vroeër, toe lettersetwerk nog per hand gedoen is, moes elke stukkie ekstra spasie in die teks per hand ingesit word en moes die lettersetter besluit waar elke spasie moes kom. Vandag word spasiëring deur rekenaarprogrammatuur gedoen, wat ongelukkig nie altyd goeie resultate lewer nie. Die verstekuitleg van programme lewer dikwels groot variasie in die spasiëring van woorde en letters, soos in Figuur 1.2.

Om die rekenaarprogrammatuur effektief te gebruik, moet die stellings (*settings*) aangepas word en selfs dan moet finale besluite op grond van visuele voorkeur deur die gebruiker self geneem word.

Spasiëring tussen woorde moet verkieslik klein wees – groot genoeg dat die woorde onderskei kan word, maar nie meer nie. Baie lettertipes laat 'n arbitrêre wydte, gebaseer op die grootte van 'n punt,

Vooruitskattings deur die bank se ekonomiese afdeling toon dat die mees realistiese siening oor die ekonomie een is waar ekonomiese groei sowat 2,6% per jaar sal wees.

Figuur 1.2: Rapport, 16 September 2001

eerder as die vorm van die letters, toe. Veral by kompakte lettertipes lei dit daartoe dat woordspasies by verstek te groot is. Meeste bladuitlegprogramme laat toe dat die woordspasie gekies kan word deur 'n persentasie te spesifiseer.

Oneweredige woordspasiëring is ongewens, maar oneweredige letterspasiëring is nog erger. Die spasie tussen letters word soms aangepas om 'n reël te vul. Dit is steurend aangesien 'n woord anders mag lyk as waaraan die leser gewoond is of dieselfde woord mag in twee opeenvolgende reëls verskillend lyk. Voorbeelde van wisselende letterspasiëring verskyn in Figuur 1.3.

te word. Sy het goeie onderskeidings in Afrikaans, Engels, wiskunde, biologie, aardrykskunde en n a t u u r - e n - skeikunde in die matriekeksamen behaal. Sy hoop om

dings vir ekonomie, wiskunde, natuur- e n - s k e i k u n d e, rekeningkunde en Engels ingepalm. Hy gaan 'n BCom r e k e n i n g k u n d e graad takel aan Tukkies. Hy gaan dié

Figuur 1.3: Rekord, 11 Januarie 2002

Wanneer groot spasies tussen woorde in opeenvolgende reëls bymekaar val, word wit eilandjies geskep wat die visuele egaligheid versteur. Dit gebeur selfs dat 'n rivier van wit spasie van bo na onder deur die teks loop. Net so steurend is woorde wat as gevolg van te groot spasies opmekaar gestapel word – swart bergreekse! Oormatige wit of swart gedeeltes versteur die egaligheid van drukwerk.

Die enigste manier om spasiëringsprobleme by teksuitleg te oorkom, is om sinvolle afkapping te gebruik. Dit is onnodig om afkapping te probeer vermy aangesien dit tot afgeronde, professionele drukwerk kan bydra.

Dit is makliker om teks met eweredig gespasieerde teks met baie afkappings te lees as met wisselvallige spasiëring en omtrent geen afkappings nie. In Figuur 1.4(a) verskyn 'n teksgedeelte uit Rapport, 19 Oktober 2001 waarin geen afkapping gebruik is nie. Dieselfde gedeelte waarin afkapping toegelaat is, verskyn in Figuur 1.4(b).

Indien 'n woord aan die einde van 'n reël afgekap word, moet die afkapping volgens erkende lettergreepverdeling gedoen word en die verstaanbaarheid van die teks moenie onderbreek word nie. As die eerste deel van die woord 'n bekende vorm het, of 'n korter woord in eie reg vorm, is dit gewoonlik 'n suksesvolle afkapping. Beide helftes moet verkieslik herkenbaar wees. Verkeerde afkapping het

In sy toespraak waarin hy grootliks gefokus het op die rol van plaaslike regering en geïntegreerde landelike ontwikkeling en die jongste inisiatiewe van die regering om deur *imbizo*'s die sienings van gewone mense aan te hoor, het Mbeki gesê Suid-Afrika word dikwels die "versinnebeelding van misdaad in die wêreld" genoem.

(a) Sonder afkapping

In sy toespraak waarin hy grootliks gefokus het op die rol van plaaslike regering en geïntegreerde landelike ontwikkeling en die jongste inisiatiewe van die regering om deur *imbizo*'s die sienings van gewone mense aan te hoor, het Mbeki gesê Suid-Afrika word dikwels die "versinnebeelding van misdaad in die wêreld" genoem.

(b) Met afkapping

Figuur 1.4: Dubbelgeskouerde teks

dikwels 'n onbekende opeenvolging van letters tot gevolg en die leser word tot stilstand geskok deur hierdie vreemde „woord". In Figuur 1.5 is voorbeelde van sulke verkeerde afkappings.

Maar nou, skaars 'n week later, is sy kaatjie van die baan. Wanneer sy honger is, maak sy fyn geluidjies en druk met haar neus teen die houtafskorting om te beduie dis tyd vir haar bottel.

Nog 'n fantastiese Huisgenoot-selfoonaanbod

Figuur 1.5: Steurende verkeerde afkapping (Huisgenoot, 13 September 2001)

Korrekte afkapping onderhou die gehalte van tipografie en maksimeer leesbaarheid. Selfs wanneer vaste woordspasies by linksgeskouerde teks gebruik word, is afkapping nodig. 'n Regterkant wat te veel varieer is steurend. In Figuur 1.6 verskyn linksgeskouerde teks met en sonder afkapping.

Almiskie. Ek voel nog sterker as my geleerde kollega oor aanspreeklikheid. Trustees van skemas met kapitaaltekorte sal vroeër of later uitvind herversekering is tog nie so 'n slegte manier om kapitaaltekorte aan te vul nie. Of dalk is 'n ondergeskikte lening ook nie so slegte 'n idee as hulle gehoor wil gee aan hul aanspreeklikheidsroeping jeens lede nie.

(a) Sonder afkapping

"Dit is geen toeval dat Mpumalanga, waar die provinsiale regering vigsberaders uit die Rob Ferreira-hospitaal gesmyt het omdat hulle AZT verskaf het, só min bestee het nie," was Kalyan se kommentaar.

Mnr. Dumisani Mlangeni, woordvoerder van Mpumalanga se gesondheidsowerheid, het egter gister gesê hul geld was eers in September vanjaar beskikbaar. Hulle het reeds 10% bestee.

(b) Met afkapping

Figuur 1.6: Linksgeskouerde teks (Beeld, 15 November 2001)

1.2 Afkapping in verskillende tale

1.2.1 Engels (Verenigde Koninkryk)

In die verlede het redakteurs en uitgewers met 'n klassieke agtergrond geglo dat lettergreepverdeling sover moontlik op grond van etimologie (dit is op grond van hul Griekse of Latynse stamme) gedoen moet word [20]. Dit het egter dikwels woordbreuke gelewer wat, alhoewel dit kosmeties goed gelyk het, nie maklik gelees het nie.

Professor Walter W. Skeat, die redakteur van die *CONCISE ETYMOLOGICAL DICTIONARY* het in 1905 gesê: "Nothing is gained by pretending to keep the root intact, when spoken utterance does nothing of the kind." Hy het voorgestel dat afkapping op grond van klank en uitspraak gedoen word, maar het voorspel dat streng, onveranderlike reëls nie daargestel sou kon word nie. Dit is duidelik uit die volgende voorbeelde [29]:

- In woorde soos *bi-o-log-ic-al*, *bi-ol-o-gist*, *bi-o-nom-ics* en *bi-on-o-my* verander lettergreepverdeling na aanleiding van uitspraak. Netso kry ons *ab-sorp-ti-om-e-ter* en *chro-nom-e-ter* teenoor *cen-ti-me-ter* en *hec-to-me-ter*.

- Uitspraak veroorsaak ook dat dit onmoontlik is om die afkapping van voor- en agtervoegsels te standardiseer. Beskou byvoorbeeld *pre-se-lect*, *pre-sen-sion*, *pres-en-ta-tion*, *pre-serv-a-tive* en *pres-er-va-tion*, *a-maz-ed-ly*, *a-mazed*.

Party Engelse woordeboeke raai gebruikers aan om nie afkapping te gebruik nie, aangesien dit so onreëlmatig is. Daar bestaan egter afkappingsprogramme waarin die taalkundige patrone geprogrammeer is om die fyn verskille in uitspraak en lettergreepverdeling op te spoor en wat uitstekende resultate lewer.

Ten spyte van die onreëlmatigheid van afkapping in Engels, bestaan die volgende riglyne [29]:

- Woorde wat die kombinasie vokaal-konsonant-vokaal bevat, maar wat soos een lettergreep klink, word nie afgekap nie soos *base*, *were*, *wife*, ens.

- As 'n kort vokaal deur een konsonant gevolg word, word na die konsonant afgekap (*ch*, *ck*, *sh*, *ph*, *th* word as een konsonant beskou) soos *per-il*, *jeal-ous*, *priv-iledge*, *pick-erel*, *gov-ernor*, ens. Dit is egter *divi-sion*.

- As 'n kort vokaal deur een of meer konsonante gevolg word, word gewoonlik na die eerste konsonant afgekap soos *Feb-ruary*, *his-tory*, *terres-trial*, *dis-cipline*, *coun-try*, *trick-ling*, *strug-gling*, ens.

- Die agtervoegsel *-ing* word gewoonlik van werkwoorde geskei in woorde soos *pull-ing*, *will-ing*, ens. Dit geld egter nie by *travel-ling* en *rob-bing* nie.

- Woorde met lang vokale of dubbel vokale aan die einde van 'n lettergreep word na die vokaal afgekap, soos in *na-tion*, *rea-son*, *deci-pher*, *fi-nite*, ens.

- Voor- en agtervoegsels word dikwels gebruik om lettergreepverdeling te doen, soos in *mis-behave*, *de-scend*, *sing-er*, *long-est*, *clear-ance*, *national-ist* ens. Dit geld egter nie by *admi-rable*, *favo-rable*, *starva-tion*, *criti-cism*, ens. nie.

Sekere woorde wat dieselfde gespel word, word na aanleiding van hul betekenis verskillend afgekap. Voorbeelde hiervan is

record	*rec-ord* of *re-cord*
crater	*cra-ter* of *crat-er*
vice	*vice* of *vi-ce versa*
nestling	*nest-ling* of *nes-tling*

Ter wille van leesbaarheid geld die volgende:

- In sekere woorde bestaan 'n lettergreep uit 'n enkele letter aan die begin van 'n woord, soos *a-ble*, *a-board*, *a-ny* ens. Sulke afkappings verbeter nie leesbaarheid nie en moet vermy word.
- Afkappings soos *passo-ver* en *une-ven* moet vermy word. Gebruik liewer *pass-over* en *un-even*.

1.2.2 Amerikaanse Engels

Britse koerante gebruik hoofsaaklik Amerikaanse taalprogrammatuur, met die gevolg dat Amerikaanse afkappingspraktyk ook daar die norm raak. *Hyphenologist* [20] se Engelse en Amerikaanse afkapingsmodules is byna identies en gebruik 'n gemeenskaplike algoritme.

Alhoewel die Amerikaanse drukkers ook afkapping op grond van fonetiese beginsels doen, verskil hul tradisies van hul Britse kollegas. Hier volg 'n paar voorbeelde waar afkapping in Amerikaanse Engels van die Britse weergawe verskil [20]:

- Die klinkende *-ed* aan die einde van woorde na byvoorbeeld *t* of *d* word toegelaat. (Moderne Britse gebruik laat ook *wick-ed-ly* en *un-doubt-ed-ly* toe.)
- Netso word *-es* aan die einde van woorde in koerante toegelaat.
- Twee-letter eindes kom algemeen voor, soos in *arous-al*, *nas-al*, *ana-lyt-ic*, *in-her-it*.
- Dit is gebruiklik om die agent-aanduiding *-er* af te kap, soos in *strang-er*, *rac-ers*, *ro-manc-er* (maar ook *gro-cer*). (Dit word al meer in Brittanje gebruik.)
- Die sagte *c* word gewoonlik in Amerikaanse Engels aan die einde van 'n lettergreep gelaat, terwyl dit in Britse Engels na die volgende lettergreep oorgedra word. In Amerikaans word ook altyd na die sagte *g* afgekap terwyl die Britte nie rondom die sagte *g* afkap nie. Verder veroorsaak die verskil in uitspraak dat afkapping in Amerikaanse en Britse Engels verskil. Voorbeelde hiervan verskyn in die volgende tabel:

Amerikaans	Brits
doc-ile	*do-cile*
de-duc-ible	*de-du-ci-ble*
crit-ic-ized	*crit-i-cised*
leg-end	*legend*
log-i-cal	*logi-cal*
de-bris	*deb-ris*
ze-nith	*zen-ith*
si-mul-ta-neous	*sim-ul-ta-neous*

1.2.3 Nederlands

Die aanbevole reëls vir afkapping in Nederlands verskyn in die *Woordenlijst van de Nederlandse taal* [15]. Afkapping word in die volgende gevalle aanbeveel:

1. Tussen twee vokale (of groepe vokale) wat direk op mekaar volg, maar nie diftonge (soos *oe*, *aai*, ens.) is nie, soos *be-amen*, *kri-oelen*, *zwaai-en*, ens.
2. Voor 'n woord of 'n stamwoord wat deel vorm van 'n saamgestelde woord, soos *doorn-struik*, *kwaad-agtig*, *weer-spannig*, ens.
3. Na 'n voorvoegsel, soos *be-horen*, *er-kennen*, *ge-lag*, *ont-zien*, ens.
4. Voor die agtervoegsels *-aard* en *-achtig*, soos *blood-aard*, *blau-achtig*, ens. en voor agtervoegsels wat met 'n konsonant begin soos *boom-pje*, *dek-sel*, *naai-ster*, ens.
5. Indien afkapping nie deur die bogenoemde reëls bepaal kan word nie, geld die volgende reëls:
   (a) Een konsonant tussen vokale word na die volgende lettergreep oorgedra, soos *ve-len*, *na-men*, *la-chen*, ens.
   (b) Waar twee konsonante tussen vokale voorkom, word die koppelteken tussen die konsonante geplaas, soos *konin-gin*, *bes-te*, *sys-teem*, ens.
   (c) Wanneer 'n woord meer as twee opeenvolgende konsonante bevat, word soveel as moontlik konsonante na die volgende reël oorgedra, soos *ek-ster*, *ad-mi-ni-stra-tie*, ens.
   (d) 'n Enkele x wat tussen vokale staan word nie afgekap nie (*exa-men*, *exo-dus*), behalwe as die x in die eerste deel van 'n saamgestelde woord voorkom (*telex-apparaat*).

Die volgende spesiale afkappingsreëls geld:

- ▷ Die deelteken verdwyn met afkapping, bv *beëindiging* word *be-eindiging* en *definiëring* word *defini-ering*.
- ▷ In verkleinwoorde waarvan die stamwoord dubbelvokale bevat, word 'n enkele vokaal gebruik, bv *papaatje* word *papa-tje*, *pianootje* word *piano-tje*, *skietje* word *ski-tje*, ens. 'n Uitsondering op hierdie reël is *dineetje* wat *diner-tje* word.
- ▷ Verkleinwoorde kan meer as een moontlike afkapping hê soos *diplomaatje* wat *diploma-tje* of *diplomaat-je* word of *vlootje* wat *vlo-tje* of *vloot-je* word, na gelang van die betekenis van die woord.

Afkapping in 'n taal waarin woorde willekeurig gevorm kan word deur woorde aanmekaar te haak, is 'n baie moeilike probleem. Waar 'n Nederlandse persoon woorde soos *verf-laag* en *beurs-prognoses* outomaties reg sal afkap (waarskynlik deur middel 'n proses van taalkundige patroonherkenning) kan 'n rekenaar nie die betekenis van hierdie woorde verstaan nie en sal dit waarskynlik as *ver-flaag* en *beur-sprognoses* afkap [29].

Die rekenaar sal selfs meer verward wees as 'n *s*, *e* of *en* tussen twee dele van 'n saamgestelde woord verskyn soos in *afdeling(s)chef*, *paard(en)bloem*, *eend(e)kroos*, ens. Die ekstra karakter word soms in die eerste lettergreep geplaas of 'n ekstra lettergreep word gevorm.

Die probleem met afkappings in die Nederlandse taal word deur die volgende woorde beklemtoon:

staats-wege	*staat-siekleed*
rechts-wege	*recht-streeks*
onthoof-ding	*hoofd-ingeland*
volks-telling	*inverzekering-stelling*
haring-spleetje	*ontharings-middel*
pols-te	*dol-ste*

1.2.4 Afrikaans

Afkapping in Afrikaans word hoofsaaklik op grond van uitspraak gedoen. In die 1991 *Afrikaanse Woordelys en Spelreëls* [28], word die volgende reëls vir afkapping gegee:

- Uitspraak geniet gewoonlik voorrang, soos *aar-tappel* in plaas van *aart-appel*, *huige-laar* in plaas van *huigel-aar*, ens.
- Leesbaarheid, herkenbaarheid en interpreteerbaarheid moet in aanmerking geneem word. Onoordeelkundige afkapping kan lastige woorddele lewer, soos *here-nig* (*her-enig*), *vere-wig* (*ver-ewig*), ens.
- In die volgende gevalle word afkapping nie op grond van uitspraak gedoen nie:
  - As 'n woord op 'n *t* of 'n *d* eindig, word voor die verkleinings-*jie* afgekap, soos in *baad-jie*, *biet-jie*, *koerant-jie*, *oond-jie*, ens.
  - Kap altyd tussen dubbele konsonante af, soos in *ap-plikant*, *bal-lerina*, *Boed-dhis*, *chauf-feur*, ens.
  - Kap tussen *n* en *g* af, soos in *En-gels*, *konin-gin*, *wan-ge*, ens.
  - As 'n *x* tussen vokale staan, word na die *x* afgekap, soos in *Alex-is*, *pirox-een*, *Tex-as*, *tax-i*.
- In die volgende gevalle is die lettergreepgrense nie duidelik nie, en is verskillende afkappings moontlik:
  - Waar *sp*, *st* en *sk* tussen vokale staan, soos in *korres-pondeer* of *korre-spondeer*, *res-toureer* of *re-stoureer*, ens.
  - Waar drie konsonante tussen vokale staan met die eerste konsonant *m*, *n*, *ng* of *r* gevolg deur *st*, soos in *aambors-tig* of *aambor-stig*, *angs-tig* of *ang-stig*, *inkoms-te* of *inkom-ste*, ens.
  - Waar *str* tussen vokale staan, soos in *Aus-tralië* of *Au-stralië*, *dis-trik* of *di-strik*, ens.
  - Waar *eks* deur een of twee konsonante gevolg word, soos in *eks-kuus* of *ek-skuus*, *eks-porteer* of *ek-sporteer*, *eks-trovert* of *ek-strovert*, ens.
  - Waar *trans* deur een of twee konsonante gevolg word, soos in *trans-pirasie* of *tran-spirasie*, *trans-kribeer* of *tran-skribeer*, ens.

Daar bestaan 'n reël dat woorde wat reeds 'n koppelteken bevat, nie weer afgekap word nie. Hierdie reël is blykbaar verouderd, aangesien sulke afkappings deesdae gereeld in gedrukte materiaal voorkom. 'n Voorbeeld hiervan verskyn in Figuur 1.7.

wat reeds van R900 miljoen tot R1,5 miljard gestyg het, na ander Audi- en Volkswagen-aanlegte in Noord- en Suid-Amerika, Europa en Asië verskuif. Maergner meen die maatskappy se uitvoer-strategie is in pas is met die doelwitte van die Mo-

Figuur 1.7: Dubbel koppeltekens (Sake-Beeld, 15 November 2001)

Dikwels word 'n woord volgens die konteks waarin dit voorkom verskillend afgekap, bv. *ge-klik* en *gek-lik*, *hou-tjie* en *hout-jie*. Geen algoritme wat woorde slegs buite konteks beskou, sal sulke uitsonderings kan identifiseer nie.

Soos by Nederlands word nuwe Afrikaanse woorde willekeurig geskep deur woorde aanmekaar te las. Hierdie saamgestelde woorde lewer groot probleme met afkapping en bemoeilik die uitleg van gedrukte materiaal.

1.3 Rekenaarafkapping

'n Rekenaar het nie die voordeel van uitspraak nie. 'n Woord is vir 'n rekenaar bloot 'n string karakters sonder enige aanduiding van natuurlike breekpunte. Deur die jare is pogings aangewend om algoritmes en programme te ontwikkel om die afkappingsprobleem mee aan te spreek.

Verskeie afkappingsalgoritmes is reeds vir Engels ontwikkel. Hierdie algoritmes is gewoonlik op 'n woordeboek- of 'n reëlbenadering of op 'n kombinasie van die twee gebaseer.

▷ **Woordeboekbenadering**

Hierdie benadering berus daarop dat 'n volledige woordelys met toegelate breek punte gestoor word. 'n Woord wat nie aan die einde van 'n reël inpas nie, word dan opgesoek om die moontlike breekpunte te kry. Hierdie benadering het die volgende nadele:

- Dit benodig baie geheue en is nie geskik vir persoonlike rekenaars nie.
- Die spoed van woordverwerking word ingeperk aangesien die afkappingsroetine telkens die gestoorde woordeboek moet oopmaak om die afkappingsposisies te bepaal.
- In sommige natuurlike tale soos Afrikaans, Duits en Nederlands word nuwe woorde na willekeur geskep deur twee (of meer) woorde saam te voeg. Indien 'n poging aangewend sou word om 'n woordeboek saam te stel, word dit gou onmoontlik groot, verouderd en is gewoonlik ontoereikend.

▷ **Reëlbenadering**

Met 'n reëlbenadering word afkapping deur 'n stel reëls, soos herkenning van algemene voor- en agtervoegsels, skeiding tussen dubbele konsonante, en ander meer gespesialiseerde reëls gedoen. Daar is egter reëls wat nie op 'n rekenaar geïmplimenteer kan word nie, soos *verdeel tussen die elemente van 'n saamgestelde woord.* Om 'n volledige stel reëls vir sekere tale saam te stel, is dus 'n moeilike en tydrowende taak.

Vir die romaanse tale soos Italiaans en Spaans is afkappingreëls eenvoudig – dit kan op 'n enkele vel papier beskryf word. Die programme wat hierdie afkappingsalgoritmes implementeer, is gevolglik ook klein, vinnig en baie akkuraat.

Vier reëlgebaseerde algoritmes word in hierdie hoofstuk bespreek, naamlik

- ▷ *Hyphenologist*,
- ▷ *Dashes*,
- ▷ *SiSiSi*, en
- ▷ Afkapping in TEX.

1.3.1 Hyphenologist

Hyphenologist is 'n program wat deur Dr Dave Fawthrop [10] en medewerkers ontwikkel is om afkapping in sowat 50 tale, insluitend tale soos Nederlands, Duits en Afrikaans, te doen. Dit is 'n aanpassing van 'n gevestigde familie van afkappingsalgoritmes en sluit 'n mate van kunsmatige intelligensie in om die gebruiker 'n sinvolle keuse tussen afkappingsposisies te gee.

Die *Hyphenologist*-algoritme gebruik voorvoegsels (*prefixes*), agtervoegsels (*suffixes*), tussenvoegsels (*infixes*) en binnewoorde (*inwords*) om afkapping in die verskillende tale te doen.

- ▷ Voorvoegsels is eenvoudige stringe karakters (met of sonder moontlike afkappingsposisies) waarmee woorde begin.
- ▷ Agtervoegsels is soortgelyk aan voorvoegsels maar kom aan die einde van woorde voor.
- ▷ Tussenvoegsels is stringe konsonante wat in woorde mag voorkom en wat gebruik word waar voor- en agtervoegsels nie die woord volledig afkap nie.
- ▷ Binnewoorde is woorde, of soms dele van woorde, wat in woorde voorkom. Dit kom hoofsaaklik in tale waar woorde na willekeur aanmekaargelas kan word, soos Duits en Afrikaans, voor.

Die algoritmes waarop *Hyphenologist* berus, is spesifiek vir verskillende tale ontwikkel. Vir tale met unieke probleme word spesiale kode ingesluit. Dit is 'n versameling van om en by 100 instrumente en reëlbasisse (uniek vir elke taal) wat op 'n meng-en-pasmetode deur die C-voorverwerker vir die verskillende tale gekies word.

Hyphenologist stel gewoonlik 'n afkapping vir elke drie of vier letters in 'n woord voor. Vir lang woorde word daar dus 'n keuse van afkappingsposisies gegee. Aan elke posisie word 'n gewig tussen 1 en 9 toegeken, en wel soos volg:

- ▷ Die begin of einde van 'n saamgestelde woord word met 'n gewig van 9 aangedui.
- ▷ Die begin of einde van morfeme[1] soos *-ion*, *-ing*, *sub-*, of 'n string karakters wat nie 'n volle woord uitmaak nie, maar wat betekenis het, soos *admin-*, *quad-* word met 'n gewig van 6 aangedui.
- ▷ Die begin of einde van 'n lettergreep word met 'n gewig van 3 aangedui.

[1] Die kleinste taaleenheid van vorm-met-betekenis [22].

Die reëls wat gebruik word, gee 'n empiriese syfer wat 'n gemiddelde van die werklike prestasie oor 'n groot hoeveelheid woorde aandui met 'n sterk neiging na woorde wat algemeen voorkom. Die volgende voorbeeld toon *Hyphenologist* se uitvoer vir die Franse woord *autosuggestion*:

- inword autosuggestion
- hyarr .3.9..3..6....
- lcword autosuggestion
- 3 9 6 3
- hyphenated au-to-sug-ges-tion

Sommige tale het unieke afkappingskonvensies wat uit die dae van handsetwerk spruit toe dit maklik was om letters by te voeg, weg te neem of te verander. Sulke reëls is baie moeilik om op 'n rekenaar te implementeer aangesien die woord weer na sy oorspronklike vorm moet teruggekeer as die koppelteken verwyder word wanneer herrangskikking plaasvind. Voorbeelde hiervan verskyn in die volgende tabel:

Taal	Woord	Afkapping
Nederlands	strootje	stro-tje
Ou Duits	Schiffahrt	Schiff-fahrt
	drucken	druk-ken
Hongaars	vissza	visz-sza
Sweeds	tillata	till-la-ta

1.3.2 Dashes

Sasha en Margaret Nizhnikov het beide by *CompuGraphics*, 'n leier op die gebied van rekenaargebaseerde tipografiese stelsels, gewerk [21]. Die afkappingsroetine wat *CompuGraphics* gebruik het, kon slegs 65% van moontlike afkappingsposisies uitken en dit was afhanklik van 'n groot uitsonderingswoordeboek wat baie geheue gebruik het. Aangesien die roetines deur probeer-en-tref ontwikkel is, was daar nie werklik 'n moontlikheid om dit te verbeter nie. Die maatskappy wou nie van voor af begin nie en het meer as sewe jaar spandeer om die bestaande roetines te probeer verbeter. Die Nizhnikovs het geglo hulle kan 'n beter produk ontwikkel en het in 1984 hul eie maatskappy, *Circle Noetic Services*, gestig.

Die eerste weergawe van *Dashes* (natuurlik vir afkapping in Engels) het teen die einde van 1984 verskyn. Terwyl Sasha die programme in C geskryf het, het Margaret die nodige taalkundige insette gelewer. Taalkundige patrone wat in woorde voorkom, vorm die basis vir die algoritmes.

- **Spoed en effektiwiteit**

  Die doelwit van 'n suksesvolle afkappingsprogram is om so min as moontlik kode te skryf om soveel as moontlik woorde te kan afkap, met ander woorde, daar is 'n stryd tussen grootte en spoed teenoor akkuraatheid. Die *Dashes*-stelsel [6] beslaan 170 KB en hanteer benewens Engels (VK en VSA) ook afkapping in die volgende tale:

Romaanse tale	Frans, Italiaans en Portugees;
Germaanse tale	Nederlands, Duits, Deens, Yslands, Noorweegs en Sweeds;
Ander	Arabies, Kroaties, Fins, Grieks, Russies en Turks.

  Selfs op gewone persoonlike rekenaars soos die 180MHz Power Macintosh of die 8500/180 of 200MHz Pentium doen *Dashes* afkapping teen 'n tempo van meer as 30 000 woorde per sekonde.

▷ **Woordlyste**

Die woordlyste van die verskillende tale sluit die volgende in:

- Basiswoorde wat soos volg in voorvoegsels, stamme en agtervoegsels opgebreek is:

```
(com/mun)ic/cate\\
(in/det)at/ion\\
(rad)ic/al\\
(re/cep\t)ion\\
un((reason)able)
```

- Die wisselvorme van 'n woord wat deur sekere reëls bepaal word. So word al die uitbreidings van die woord *play* verkry deur die reël v2 na die woord te spesifiseer:

```
play v2\\
---\\
Rule v2: -#,-ed,-ing--s\\
-->play, played, playing, plays
```

- 'n Gekombineerde formaat van bogenoemde twee gevalle:

```
(com/mun)ic/cate v1\\
(in/det)at/ion N1\\
(rad)ic/al a2\\
```

- Lyste van uitgebreide vorme waar die verskillende woorde elkeen op 'n nuwe reël gegee word:

```
play\\
played\\
playing\\
plays
```

- 'n Lys van 28 000 woorde in standaardspelling en -uitspraak vir Amerikaanse Engels met lettergreepverdeling en aksentaanduidings.
- 'n Lys van 10 000 Engelse idiome en *cliches* met geassosieerde betekenisse, soos

```
kick the bucket = die\\
out of this world = wonderful\\
over the hill = old
```

▷ **Uitvoer**

Nadat die *Dashes*-program geloop het, word alle diskresionêre afkappingsposisies van woorde gegee terwyl 'n rangorde ('n waarde tussen 0 en 4) aan elkeen toegeken word. Voorbeelde hiervan is:

- Die Engelse woord *antidisestablishmentarianism* word gegee as

  an-3ti-1dis-1es-2tab-2lish-1men-2tar-1i-3an-1ism.

  Die beter afkappingsposisies word dus deur die laer syfer aangedui. Die woord sal dus eerder by rangorde 1 as by 2 of 3 afgekap word.

– Die Duitse woord *Muttersprache* word gegee as

Mut-2ter-0spra-2che.

Die woord sal dus eerder tussen die twee saamgevoegde woorde (rangorde 0) afgekap word as by gewone lettergreepverdelings (rangorde 2).

1.3.3 SiSiSi

SiSiSi is 'n afkappingspakket wat deur Barth, Kodydek en Schönhacker ontwikkel is om betroubare en sinvolle afkapping vir Duits doen [2]. Die naam SiSiSi is afgelei van die Duitse woorde *Sichere Sinnentsprechende Silbentrennung.*

In 'n bespreking van die pakket word die volgende punte uitgelig:

▷ **Motivering:**

– Outomatiese afkapping gedurende die proses van teksuitleg is essensieel om goeie gehalte dokumente te lewer en vir die uitleg van smal kolomme soos in koerante; sonder afkapping mag groot woordspasies ontstaan.

– Die algemene afkappingsmetodes vir Engels, wat òf reëlgebaseerd òf woordeboekgebaseerd is, is nie vir die Duitse taal toepaslik nie as gevolg van die gereelde gebruik van saamgestelde woorde.

▷ **Ontwikkeling:**

Die afkappingsalgoritme is op woordontleding gebaseer. Dit is betroubaar en sinvol in die sin dat dit verkeerde afkappings of afkappings wat die betekenis van 'n woord mag verander, herken en vermy. Die belangrikste reëls vir die vorming van Duitse woorde word gedurende woordanalise toegepas.

Die stelsel word aangepas om die verandering in die Duitse ortografie te hanteer. Dit sal verder verbeter word deur genaturaliseerde woorde uit ander tale sistematies te integreer.

▷ **Betroubare afkapping:**

Die algoritme beskou alle moontlike opbreking van woorde. Afkappings wat nie in alle variante voorkom nie, is *onveilig* en moet nie gebruik word tensy die gebruiker, byvoorbeeld, geraadpleeg is nie. Voorbeelde hiervan is

Bau=mast	of	Baum=ast
Wach=stu-be	of	Wachs=tu-be

(Hier dui = *belangrike* en - *minder belangrike* afkappingspunte aan.)

▷ **Sinvolle afkapping:**

Sinvolle afkapping word bevorder deur voorkeur aan *belangrike* afkappingspunte op die grense tussen onderskeie saamgestelde enkelwoorde te gee. Alle ander afkappingspunte word as *minder belangrik* gemerk. Beskou byvoorbeeld die volgende:

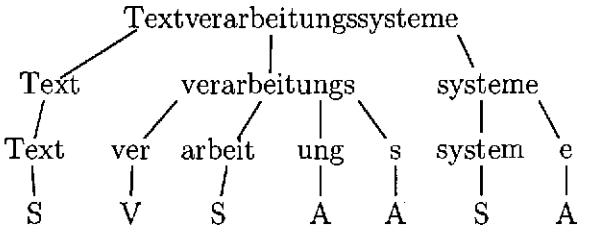
Woord	Verkies	Vermy
Getränkeautomaten	Getränke=automaten	Getränkeauto-maten
Kamerataschen	Kamera=taschen	Kamerata-schen

▷ **Algoritme:**

Die algoritme bestaan uit die volgende twee stappe:

1. Opbreking:

   Die woord onder beskouing word in die saamgestelde enkelwoorde verdeel. Hierdie enkelwoorde word dan in hul atomiese komponente (S = stam, V = voorvoegsel, A = agtervoegsel) opgebreek. Beskou byvoorbeeld die woord *Textverarbeitungssysteme*:



   Hierdie stap lewer reeds afkappingspunte tussen enkelwoorde sowel as na voorvoegsels soos *ver-* en voor spesiale agtervoegsels soos *-heit*, *-chen*, ens. Op hierdie stadium het ons die volgende afkappingspunte:

$$Text = ver \text{ - } arbeit \mid ung \mid s = system \mid e$$

   Die | dui aan dat uitsluitsel oor hierdie punte nog nie bepaal is nie – afkappingsreëls moet hier toegepas word.

2. Reëltoepassing om afkappingspunte in enkelwoorde te bepaal:

   Afkappingspunte binne enkelwoorde (*stam* + *agtervoegsels*) word bepaal deur reëls wat op die opeenvolging van konsonante en vokale gebaseer is, toe te pas. Die belangrikste reël is om voor die laaste konsonant in 'n groep konsonante af te kap. Party spesiale kombinasies van letters word as een konsonant beskou soos *ch* en *sch*, *st* (ou ortografiese reëls) en *ck* (nuwe ortografiese reëls). Die volledige afkapping is

$$Text = ver \text{ - } ar \text{ - } bei \text{ - } tungs = sy \text{ - } ste \text{ - } me.$$

▷ **Atomiese tabel:**

In die eerste stap gebruik die algoritme alle atomiese komponente van die woord. Aangesien elke stam net een keer sonder enige agtervoegsels voorkom en die program self saamgestelde woorde hanteer, is die tabel baie klein. Minder as 8 000 items is genoeg om (byna) alle Duitse woorde en meeste belangrike genaturaliseerde woorde uit ander tale te hanteer.

▷ **Gebruikerinteraksie:**

Interaksie met die gebruiker geskied deur die volgende twee boodskappe:

- *Onbekende woord*: Die woord kan as gevolg van spelfoute, eiename, plekname, ens. nie opgebreek word nie. Die gebruiker kan die woord regmaak of per hand afkap.
- *Dubbelsinnige woord*: Die woord kan op meer as een manier opgebreek word. Die moontlikhede word aan die gebruiker getoon wat dan die korrekte afkappingsvariant kies.

1.3.4 Afkapping in TEX

TEX (*Tau Epsilon Chi*) is 'n rekenaartaal wat veral vir die druk van wiskundige en ander tegniese dokumente ontwerp is [30]. In die laat 1970s, met die druk van Volume 2 van sy reeks boeke *The Art of Computer Programming* in Amerikaanse Engels, was Donald Knuth so ontevrede met die gehalte van die rekenaardrukwerk, dat hy begin het om TEX te ontwikkel.

Oorspronklike algoritme

In 1977 het Donald Knuth en Frank Liang die oorspronklike afkappingsalgoritme vir TEX ontwerp [17]. Dit was 'n reëlgebaseerde algoritme met drie hoofreëls, naamlik

- ▷ kap voor 'n agtervoegsels af,
- ▷ kap na 'n voorvoegsel af,
- ▷ kap tussen konsonante af wanneer die patroon *klinker-konsonant-konsonant-klinker* voorkom.

Spesiale reëls soos *kap tussen 'n vokaal en q af*; *kap na ck af*, ens. het die reëlbasis voltooi. 'n Uitsonderingswoordeboek van ongeveer 300 woorde het uit woorde wat deur bogenoemde reëls verkeerd afgekap is, of nie afgekap kon word nie, bestaan en is bykomend gebruik.

In toetse wat op 'n woordelys uit 'n sakwoordeboek gedoen is, is omtrent 40% van toelaatbare afkappingspunte bepaal, met 1% foute (relatief tot die totale aantal afkappingspunte).

Patrone

Die posisie waar afkapping in 'n woord moet plaasvind, word deur sekere letterpatrone bepaal. Deur sulke patrone te identifiseer, kan afkappingsposisies dus bepaal word. As die patroon *-tion* byvoorbeeld in 'n woord voorkom, moet afkapping meestal voor die *t* plaasvind.

In die volgende tabel verskyn voorbeelde van goeie *afkappingspatrone* in Engels wat deur 'n uitgebreide studie bepaal is. Hier dui 'n punt die begin of einde van 'n woord aan.

.in-d	.in-s	.in-t	.un-d	b-s	-cia	con-s	con-t
e-ly	er-l	er-m	ex-	-ful	it-t	i-ty	-less
l-ly	-ment	n-co	-ness	n-f	n-l	n-si	n-v
om-m	-sion	c-ly	s-nes	ti-ca	x-p		

Vir elke goeie afkappingspatroon is daar gewoonlik uitsonderings. So, byvoorbeeld, geld die „veilige" patroon *-tion* nie vir *cat-ion* nie en waar *n-t* in 'n woord voorkom is afkapping in 80% van die gevalle korrek, die res is uitsonderings. Nadat 'n stel afkappingspatrone gekies is, mag daar dus duisende uitsonderings wees wat in 'n uitsonderingswoordeboek gehou kan word.

Ook in die uitsonderings bestaan daar baie ooreenkomste. Die *vokaal-konsonant-konsonant-vokaal*-reël lewer byvoorbeeld dikwels foute van die vorm *X-Yer* of *X-Yers* vir sekere konsonantpare *XY*. Groot groepe uitsonderings kan nou weer deur reëls hanteer word. In die volgende tabel verskyn daar *inhiberingspatrone* wat aandui waar afkapping nié moet plaasvind nie.

b=ly	bs=	=cing	io=n	i=tin	=ls	nn=
ns=	n=ted	=pt	ti=al	=tly	=ts	tt=

Nadat een stel afkappings- en inhiberingspatrone aangewend is, mag 'n ander stel afkappingspatrone geïdentifiseer word, dan weer 'n ander stel inhiberingspatrone, ensovoorts. Elke vlak van patrone kan as uitsonderings op die uitsonderings van die vorige vlak gesien word. (Die TEX82 algoritme gebruik vyf sulke afwisselende vlakke van afkappings- en inhiberingspatrone.)

Die voordele van 'n patroongebaseerde benadering is die volgende:

- ▷ Dit is 'n algemene vorm van die afkappingsreëls. Dit kan voor- en agtervoegsels en ander reëls as spesiale gevalle insluit.
- ▷ Selfs 'n uitsonderingswoordeboek kan in terme van patrone beskryf word deurdat volledige woorde as patrone weergegee word. (Patrone is meer kompak as 'n uitsonderingswoordeboek aangesien 'n enkele patroon verskeie vorms van 'n woord kan hanteer, soos *pro-gram*, *pro-grams* en *pro-grammed*.)
- ▷ Dit is baie effektief. 'n Geskikte stel patrone bevat die inligting wat vir afkapping nodig is in kompakte formaat.
- ▷ Die formaat van patroonreëls is eenvoudig genoeg om outomaties uit 'n woordeboek gegenereer te kan word.

TEX82

Na 'n deeglike studie van afkapping in Engels, het Frank Liang 'n nuwe algoritme vir TEX82, wat op die idee van afkappingspatrone gebaseer is, ontwikkel [17].

Om 'n geskikte stel patrone vir afkapping te kies, moet die verlangde eienskappe van die afkappingsalgoritme gespesifiseer word. Die doelwit van die TEX82 algoritme is om slegs 'n matige hoeveelheid spasie (20–30Kb) te gebruik (insluitend 'n uitsonderingswoordeboek) en om soveel as moontlik afkappingspunte te bepaal terwyl min (of geen) foute gemaak word. Met ander woorde, maksimeer die aantal afkappingspunte terwyl foute en die spasie wat deur die algoritme benodig word, geminimeer word.

Die resultate van die afkappingsalgoritme is natuurlik afhanklik van die woordelys wat gebruik word. Aangesien die *Webster's Pocket Dictionary* meeste afgeleides van woorde bevat, is dit as basis vir die algoritme gebruik. Meer gewig is aan 'n paar duisend algemene woorde gegee, sodat die waarskynlikheid dat dit afgekap sal word, groter is.

Uitsonderingswoorde is bepaal deur die algoritme (gebaseer op die aanvanklike woordelys) teen 'n groter woordeboek te toets. Een duisend van hierdie woorde is bygevoeg om seker te maak dat hulle nie verkeerd afgekap sal word nie. Dit het saamgestelde woorde soos *camp-fire*, name soos *Af-ghan-i-stan* en nuwe woorde soos *bio-rhythm* ingesluit.

Om 'n woord af te kap, word alle patrone wat met die patrone in die woord ooreenstem geroep en 'n afkappingswaarde word vir elke posisie in die woord bepaal. As meer as een waarde vir 'n sekere posisie voorkom, word die hoogste waarde gebruik. (Hoër waardes geniet dus voorrang.) Indien die afkappingswaarde onewe is, is afkapping in die posisie moontlik. Indien die waarde ewe is, of geen waarde is gespesifiseer nie, word die woord nie in daardie posisie afgekap nie.

Die PATGEN-program wat op bladsy 21 bespreek word, word gebruik om afkappingspatrone vir TEX82 te genereer. Die ontwikkeling van die datastrukture wat in die PATGEN-program gebruik word, asook die verskillende metodes om dit te minimeer, word in die volgende afdeling kortliks bespreek.

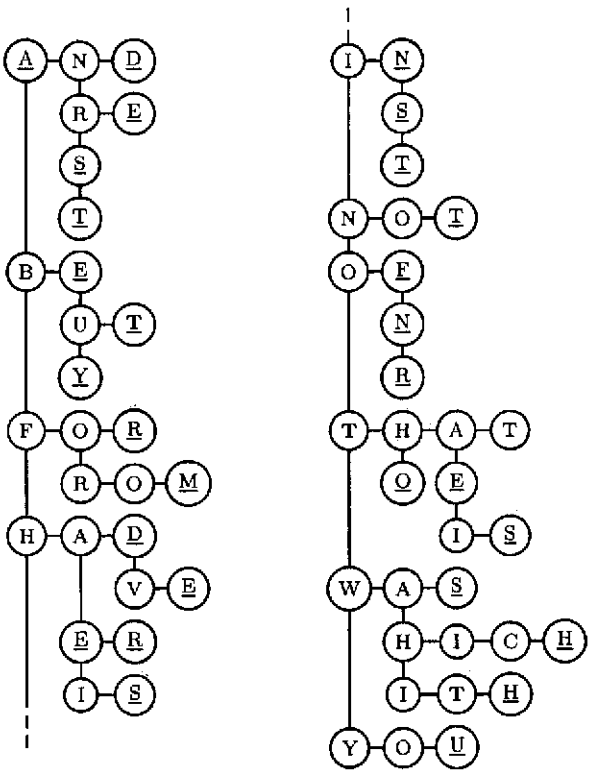
Datastrukture

Patrone word in datastrukture waaruit gegewens maklik onttrek kan word, gestoor. Liang het 'n datastruktuur wat hy 'n gepakte *trie* noem uit gekoppelde *trie*-strukture en indeks-*tries* ontwikkel. Die naam *trie* kom uit die woord „re*trie*val" en word as „traai" uitgespreek. In hierdie *trie*-strukture is elke sleutel 'n reeks waardes oor 'n eindige alfabet (met m elemente) en is dus ideaal om sleutels van variërende lengte te hanteer.

▷ 'n Gekoppelde *trie*-struktuur

In Figuur 1.8 verskyn 'n gekoppelde *trie* vir die 31 mees algemene woorde in Engels, naamlik

A	AT	FOR	HE	IN	OF	THE	WHICH
AND	BE	FROM	HER	IS	ON	THIS	WITH
ARE	BUT	HAD	HIS	IT	OR	TO	YOU
AS	BY	HAVE	I	NOT	THAT	WAS	



Figuur 1.8: 'n Gekoppelde *trie*-struktuur vir die 31 mees algemene Engelse woorde

In so 'n struktuur word die karakters van 'n ingevoerde woord sekwensieel met die karakters in die nodusse vergelyk.

- As die letters ooreenstem, word na die volgende letter in die sleutel beweeg.
- As die letters nie ooreenstem nie, word die volgende vertakking ondersoek.
- 'n Onderstreepte letter dui die einde van 'n woord aan ('n eindpuntnodus). As die einde van 'n woord met 'n eindpuntnodus ooreenstem, is die woord deel van die stel woorde

onder beskouing en die uitvoer is *ja* (1). Indien dit nie die geval is nie, is die uitvoer *nee* (0).

'n Soektog deur so 'n gekoppelde *trie* is taamlik stadig aangesien elke woordkarakter tot soveel as m (die grootte van die alfabet) keer met die noduskarakters vergelyk kan word.

▷ Indeks-*trie*

In 'n indeks-*trie* word dieselfde beginsel as hierbo gebruik, maar die vertakkings word in 'n skikking met m kolomme en $n + 1$ rye uitgevoer, waar n die aantal woorde in die *trie* is. Die elemente van die skikking dui aan watter „familie" van die *trie* volgende ondersoek moet word. 'n Groep nodusse in 'n gekoppelde *trie* word as 'n familie beskou.

'n Indeks-*trie* vir die 31 mees algemene woorde in Engels verskyn in Figuur 1.9. Die soektog deur so 'n indeks-*trie* is vinnig aangesien die aanvangsposisie in die *trie* maklik bepaal kan word. Dit gebruik egter baie onnodige spasie aangesien slegs 'n paar posisies inskrywings bevat, terwyl die res leeg is. In die indeks-*trie* hierbo is $26 \times 32 = 832$ posisies waarvan slegs 59 gebruik word.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	5				7		11	16					17	19					20			24		31	
2														3				4	0	0						
3				0																						
4					0																					
5					0																6				0	
6																				0						
7															8			9								
8																		0								
9															10											
10													0													
11	12				14			15																		
12				0																						
13					0																					
14																		0								
15																			0							
16														0					0	0						
17															18											
18																				0						
19						0								0				0								
20								21							0											
21	22				0				23						0											
22																				0						
23																			0							
24	25							26	29																	
25																				0						
26									27																	
27			28																							
28								0																		
29																				30						
30								0																		
31															32											
32																					0					

Figuur 1.9: 'n Indeks-*trie* vir die 31 mees algemene Engelse woorde

▷ Gepakte *tries*

Liang het 'n tegniek ontwikkel wat die beste eienskappe (spasie en spoed) van bogenoemde twee tegnieke kombineer. Dit word *gepakte tries* genoem aangesien die koppelings van een familie in die leë spasies in 'n indeks-*trie*, wat vir nul-koppelings van ander families gereserveer is, ingepak word. In die volgende figuur word so 'n situasie uitgebeeld:

(Voortaan word na families as *toestande* en na posisie-aanwysers as *oorgange* verwys om met die terminologie van eindige-toestandmasjiene ooreen te stem.)

In 'n gepakte *trie* is dit nodig om aan te toon of 'n oorgang op die huidige toestand betrekking het of aan 'n ander toestand, wat in dieselfde area gepak is, behoort. Dit word gedoen deur 'n karakterindeks saam met die oorgangsindeks te stoor. Slegs wanneer die karakter onder beskouing dus met die karakterindeks ooreenstem, behoort 'n oorgang aan die toestand onder beskouing. Die bykomende vereiste dat verskillende toestande nie in dieselfde basisposisie gepak mag word nie, moet dus nagekom word.

Die *trie* word met die *eerste-pas*metode gepak. Dit beteken dat toestande een-een ingepak word met elke toestand in die plek met die laagste indeks waar dit sal inpas sonder om met enige vorige gepakte oorgange of reeds gevulde basisposisies te oorvleuel. Dikwels word byna alle oop spasies in die *trie* met oorgange van ander toestande gevul.

In Figuur 1.10 is die indeks-*trie* in 'n enkelry skikking gepak – die rye in die figuur vorm eintlik een lang ry. (In hierdie geval is agtervoegselkompressie, wat in die volgende afdeling bespreek word, toegepas.)

	0	1	2	3	4	5	6	7	8	9
00		A 8	B11			D0	F 3	E 0	H30	I23
10	C 5			H 0	N25	O32	E 0		O12	M0
20	T33	R14	N 1	W46	T 0	Y37	R 2	S 0	T 0	O 6
30	R 0	A29	U 4	D 0	S 0	E12	Y 0	N 0	F 0	I15
40	O 4	H44	S 0	T 0	1 7	A 4	N 0	A15	Q 0	E 0
50	R 0	V 2	O38	I15	H35	I36	T 5			U 0

Figuur 1.10: Gepakte *trie* vir die 31 mees algemene Engelse woorde

Met 'n gepakte *trie* word die 31 mees algemene Engelse woorde dus in slegs 60 posisies ingepak in vergelyking met die 832 wat nodig was met die indeks-*trie*.

Om die woord FROM in die gepakte *trie* te soek, gaan ons soos volg te werk:

- Die waardes 1 tot 26 word met die letters A tot Z geassosieer. Die letter **F** stem dus met posisie 6 ooreen. Soek die letter **F** in posisie 6. Die inskrywing is F 3. Aangesien die karakterindeks F is, is dit 'n geldige oorgang. Gaan dus na posisie 3.
- In posisie 3 is geen inskrywing nie en ons soek die letter **R**. Soek nou van hierdie posisie af die eerste **R**-karakterindeks – R14 in posisie 21. Gaan na posisie 14.

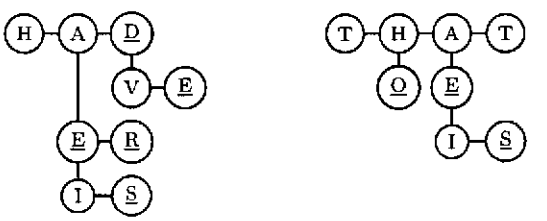
- Die inskrywing in posisie 14 is N25. Ons soek 'n **O**. Dit is dus nie 'n geldige oorgang nie. Soek van posisie 14 af die eerste O-karakterindeks – O32 in posisie 15. Gaan na posisie 32.
- Die inskrywing in posisie 32 is A29. Ons soek 'n **M**, dus is dit nie 'n geldige oorgang nie. Soek van hier af die eerste M-karakterindeks – M 0 in posisie 19. Die feit dat dit onderstreep is, dui daarop dat die woord voltooi is en wel in die datastel voorkom. Die uitvoer van die soektog is dus 1.

Minimeringsmetodes

▷ **Agtervoegselkompressie**

'n Groot voordeel van *trie*-datastrukture is dat agtervoegsels wat in verskeie woorde dieselfde is, outomaties in verteenwoordigende paaie in die struktuur gekombineer kan word. Die woordeboek word dus tot 'n mate saamgepers. Om spasie te spaar, word gemeenskaplike gedeeltes van *tries* saamgevoeg. (Ooreenkomste word van die einde van die vertakkings gesoek.)

Beskou byvoorbeeld die woorde THIS en HIS in die gekoppelde *trie* in Figuur 1.8. Die toestande wat hierdie woorde bevat, word in Figuur 1.11 herhaal.



Figuur 1.11: Toestande wat die woorde HIS en THIS bevat

Die eindnodusse van beide HIS en THIS toets vir die letter S. Hierdie twee nodusse kan gekombineer word deur hul voorgangers na dieselfde nodus te laat verwys. Die stel woorde waarna die *trie* verwys, verander nie. Die voorgangernodusse kan op dieselfde manier gekombineer word, aangesien beide vir I toets en na die S-node gaan indien suksesvol. Alhoewel die voorgangers van die I-nodusse beide E is, kan hulle nie gekombineer word nie, aangesien die een aan die linkerkant ook na R vertak.

Alle eindnodusse wat vir dieselfde karakters toets kan so saamgevoeg word. Dit alleen spaar soveel nodusse as die aantal woorde in die woordeboek plus hoogstens 26. Verder kan langer agtervoegsels soos *-ly*, *-ing*, *-tion*, ens. ook dikwels gekombineer word.

Die *trie*-gebaseerde woordeboekvoorstelling kan aangepas word om afkapping te doen deur die uitvoer te verander. In die uitvoer word dan aangedui waar die woord afgekap kan word. In plaas van 'n enkel-bisuitvoer wat sê of 'n woord in die woordeboek is of nie, word moontlike afkappingsposisies in die uitvoer aangedui.

▷ **Voorvoegselkompressie**

Soos met agtervoegsels, kan voorvoegselkompressie ook gedoen word. Afkapping word byvoorbeeld aangedui sodra die letters *hyph* aan die begin van 'n woord voorkom. Die uitvoer sal sê

kap na die tweede letter af aangesien dit vir alle woorde wat met hierdie letters begin, geld. Dit geld egter nie vir die letters *hyp* nie aangesien woorde soos *hyp-not-ic* dit weerspreek. Daar is heelwat woorde wat voorvoegels van ander woorde is, maar wat verskillend afgekap word, soos *ca-ret* en *care-ta-ker* of *as-pi-rin* en *as-pir-ing*. Hierdie probleem word aangespreek deur 'n karakter aan die einde van die woord te haak voordat dit in die gepakte *trie* geplaas word.

▷ ***Gee-nie-om-nie*-kompressie**

Soos voorheen opgemerk is, word die meeste posisies in 'n indeks-*trie* nie gebruik nie. Aangesien hierdie leë posisies nooit deur woorde in die woordeboek gevul word nie, kan hulle toegelaat word om arbitrêre oorgange te bevat. Dis nie dieselfde as oorvleuelende toestande wat in gepakte *tries* voorkom nie – die oorgange wat bygevoeg word, word uiteindelik deel van die toestand.

Hierdie *gee-nie-om-nie*-benadering bevorder die minimeringsproses op die volgende twee maniere:

- Toestande hoef nie meer identies te wees om saamgevoeg te word nie. Net die karakters met eksplisiete oorgange moet ooreenstem.
- Deur die samevoeging van nie-ekwivalente toestande word verdere samevoeging dikwels moontlik aangesien oorgange ekwivalent word.

Beskou byvoorbeeld weer die toestande van die *trie* in Figuur 1.11 wat die woorde HIS en THIS bevat. Ons het vroeër gesien dat die I en die S in HIS en THIS saamgevoeg kan word, maar nie die E nie. In *gee-nie-om-nie*-kompressie kan dit egter saamgevoeg word. So 'n samevoeging vereis dat die DV toestand na die A aan die linkerkant en die T aan die regterkant saamgevoeg word. Dit is moontlik aangesien daar geen oorvleuelende oorgange teenwoordig is nie.

Patroongenerasie

Patroongenerasie word deur die PATGEN-program gedoen. Dit gebruik 'n lys van woorde met korrekte afkappings en genereer 'n stel patrone wat in die TEX82 afkappingsalgoritme gebruik kan word.

Om ingewikkelde interaksies tussen die verskillende patrone wat op 'n bepaalde afkappingspunt betrekking mag hê uit te skakel, word patrone in 'n recks lopies deur die woordlys gekies. In elke lopie word slegs die invloed van patrone wat in die voorafgaande lopies gekies is, in berekening gebring. Verder word 'n gulsige benadering gebruik waarin die mees effektiewe patrone in elke lopie gekies word.

Die effektiwiteit van 'n patroon word gedefinieer as

$$eff = goed/(1 + sleg)$$

waar *goed* die aantal korrekte afkappings is, en *sleg* die aantal foute wat gemaak is. As inhiberingspatrone gebruik word, word die hoeveelheid *sleg* 'n bietjie afgeskaal na gelang van die patroon se effektiwiteit op die volgende vlak. 'n Beter formule is

$$eff = \frac{goed}{1 + sleg/sleg\_eff}$$

waar *sleg_eff* die beraamde effektiwiteit op die volgende vlak is. 'n Redelike beraming gebaseer op vorige lopies van die patroongenerasieprogram word as beraming van die effektiwiteit op die volgende vlak gebruik.

Die gegenereerde patrone word in 'n gepakte *trie* gestoor. Aangesien die stel patrone met elke lopie aangevul word, is dit nodig om die nuwe patrone dinamies in die gepakte *trie* te kan skryf. 'n Dinamies gepakte *trie* word deur die volgende gekenmerk:

- ▷ Dis kompak en lewer baie vinnige resultate.
- ▷ Agtervoegselkompressie word nie gebruik nie aangesien dit terugverwysings benodig wat die proses bemoeilik. Verder het dit nie 'n groot effek op die grootte van die *trie* nie aangesien die patrone onder beskouing relatief kort is.
- ▷ Veranderde toestande word telkens herpak en skakels opgedateer om pakkingsprobleme op te los. Vir goeie spasiebenutting word 'n dinamiese lys van onbenutte selle in die *trie* gehou. Tensy die *trie* baie yl is, is dit duur om groot toestande te probeer herpak. Dit word oorkom deur 'n drumpel op eerste-paspogings te plaas. Indien die drumpel oorskry word, word sulke toestande in die oop spasie direk regs van die besette posisies gepak.

Om statistiek oor die patrone in elke woord in te samel, moet die woord eers volgens die patrone wat in voorafgaande lopies gekies is, afgekap word. Die patroongenerasieprogram gebruik dus 'n tweede gepakte *trie* om die stel patrone wat sover gekies is, te stoor.

Om al die patrone in 'n sekere woord te bepaal, word 'n soektog vanaf elke letter in die woord uitgevoer. Nadat 'n soektog vanaf 'n sekere letter voltooi is, mag dit nodig wees om agteruit in die woord te beweeg om die volgende soektog te begin. Die konteksvrye aard van patrone maak dit moontlik.

Die patrone bestaan uit stringe letters en syfers waar elke syfer 'n afkappingswaarde vir 'n sekere posisie in 'n woord aandui. In Tabel 1.1 verskyn 'n paar voorbeelde van woorde met die patrone wat van toepassing is, die resulterende afkappingswaardes en die uiteindelike afkapping. Let daarop dat indien meer as een waarde vir 'n bepaalde posisie gespesifiseer word, geniet die hoër waarde voorrang. Indien die finale waarde onewe is, is afkapping in daardie posisie toelaatbaar, terwyl 'n ewe waarde aandui dat afkapping nie toelaatbaar is nie.

Woord	**Geassosieerde patrone**	**Afkappingswaardes**	**Afkappings**
algorithm	l1g4 lgo3 1go 2ith 4hm	al1g4o3r2it4hm	al-go-rithm
computer	4m1p pu2t 5pute put3er	co4m5pu2t3er	com-put-er
mathematics	math3 ath5em th2e 1ma at1ic 4cs	math5e1mat1i4cs	math-e-mat-ics
program	pr2 1gr	pr2o1gram	pro-gram
typesetting	type3 e1s2e 4t3t2 2t1in	type3s2e4t3t2ing	type-set-ting

Tabel 1.1: Voorbeelde van patrone en afkappings deur PATGEN gegenereer

Hoofstuk 2

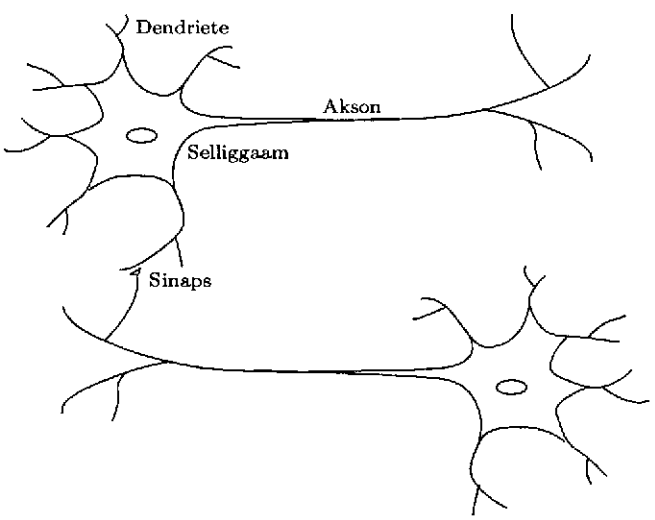
Neurale netwerke

Die afkappingsprobleem is inherent 'n patroonherkenningsprobleem. Aangesien kunsmatige neurale netwerke op vele gebiede suksesvol vir patroonherkenning aangewend word, maak dit sin om ook op hierdie gebied daarmee te eksperimenteer. In hierdie hoofstuk verskyn die basiese agtergrond oor kunsmatige neurale netwerke asook vier voorbeelde van neurale netwerke wat ontwikkel is om afkapping te doen.

2.1 Inleiding

Die studiegebied van neurale netwerke het 'n geskiedenis van om en by vyf dekades, maar het eers oor die afgelope vyftien jaar werklik toepaslik begin raak en is steeds besig om te ontwikkel [13]. Dit verskil dus van beheerstelsels of optimering waar terminologie, basiese wiskunde en ontwerpprosedures oor baie jare gevestig en toegepas is.

Kunsmatige neurale netwerke is inligtingverwerkers wat deur die werking van biologiese senuweestelsels geïnspireer is. In Figuur 2.1 verskyn 'n skematiese voorstelling van biologiese neurone.

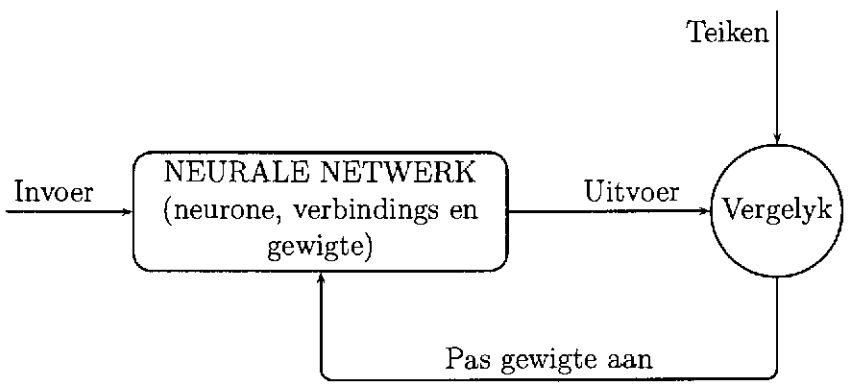


Figuur 2.1: Biologiese neurone

Kunsmatige neurale netwerke is 'n versameling van wiskundige modelle wat die biologiese eienskap van *leer deur middel van aanpassing* naboots. Dis saamgestel uit 'n aantal eenvoudige verwerkingseenhede (neurone) wat met mekaar kommunikeer deur seine oor 'n groot aantal geweegde verbindings (sinapse) te stuur.

'n Neuron ontvang invoer vanaf ander neurone (of eksterne bronne), bereken 'n uitvoersein wat na ander neurone gestuur word en dit pas gewigte op verbindings aan. Soos in die natuur word die funksie van 'n netwerk grootliks deur die verbindings tussen die neurone bepaal. 'n Neurale netwerk kan afgerig word om 'n sekere funksie uit te voer deur die waardes (gewigte) op die verbindings aan te pas. Afrigting vind plaas deur die gewigte so aan te pas dat 'n sekere invoer na 'n spesifieke teikenuitvoer lei.

In Figuur 2.2 verskyn 'n skematiese voorstelling van die afrigting van 'n neurale netwerk.



Figuur 2.2: Afrigting van 'n neurale netwerk

Kontrole van die afrigtingsproses vind plaas deurdat die uitvoer van die netwerk telkens met die teikendata wat verskaf word, vergelyk word. Gewigte word aangepas totdat die netwerk se uitvoer genoegsaam met die ingevoerde teiken ooreenstem. Die proses word in die volgende stappe saamgevat:

1. Ken aanvanklike gewigte aan verbindings toe (gewoonlik willekeurig).
2. Voer 'n afrigtingsdatapaar (invoer en teiken) aan die neurale netwerk.
    3. Bepaal die huidige uitvoer deur neurale-netwerkberekenings.
    4. Vergelyk uitvoer met die teiken. Indien die verskil kleiner as 'n voorafbepaalde waarde is (of 'n sekere aantal iterasies is uitgevoer) gaan na stap 6. Indien nie, gaan na stap 5.
    5. Pas gewigte aan.
6. Indien nog data beskikbaar is, gaan na stap 2. Indien nie, stop.

Daar bestaan twee metodes van afrigting wat op die manier waarop data aan die neurale netwerk gevoer word, gebaseer is, naamlik sekwensiële en lotafrigting.

- ▷ Sekwensiële afrigting:
  Datapare (invoer met verlangde uitvoer) word een vir een ingevoer, vorentoe deur die netwerk gevoer en gewigte word aangepas. Wanneer al die data ingevoer is, word 'n volgende epog begin waarin die data weereens een vir een ingevoer word. Die proses gaan voort totdat 'n voorafbepaalde vlak van ooreenstemming bereik word of totdat 'n sekere aantal iterasies uitgevoer is.

- ▷ Lotafrigting:
  Met lotafrigting word al die beskikbare data gelyktydig ingevoer. Stappe drie tot vyf word herhaal totdat die uitvoer van die neurale netwerk tot op 'n voorafbepaalde presiesheid met die teikens ooreenstem of totdat 'n sekere aantal iterasies uitgevoer is.

Die aanpassing van gewigte in 'n neurale netwerk word deur middel van 'n *afrigtingsreël* gedoen. Die doel daarvan is om die netwerk se gewigte so aan te pas dat die netwerk 'n sekere uitvoer sal lewer.

Afrigting soos hierbo, waar 'n stel voorbeelde van hoe die netwerk behoort te funksioneer (elke invoer het 'n ooreenstemmende korrekte teiken waarteen die neurale netwerk homself kan meet) verskaf word, word afrigting met kontrole (*supervised training*) genoem. Daar bestaan ook afrigtingsmetodes waar gewigte slegs in reaksie op invoere aangepas word. (Geen teikens is beskikbaar nie.) Sulke metodes word afrigting sonder kontrole (*unsupervised training*) genoem. Baie van hierdie netwerke klassifiseer invoerpatrone in 'n eindige aantal klasse. Dit is veral bruikbaar in toepassings soos vektorkwantifisering.

Neurale netwerke kan afgerig word om probleme wat vir konvensionele rekenaars en vir mense moeilik is, op te los. Dit word reeds gebruik om komplekse funksies in verskeie dissiplines uit te voer. Voorbeelde hiervan is die volgende [13]:

- ▷ Bankwese:
  Die lees van tjeks en ander dokumente; evaluasie van krediettoekenning; ens.

- ▷ Medies:
  Ontleding van borskankerselle; EEG- en EKG-analise; prosteseontwerp; optimering van oorplantingstye; ens.

- ▷ Ruimtevaart:
  Vlugroetesimulasie; ruimtetuigkontrolestelsels; opsporing van foute in ruimtetuie; ens.

- ▷ Spraak:
  Spraakherkenning; spraakkompressie; vokaalklassifikasie; teks-tot-spraaksintese; ens.

- ▷ Telekommunikasie:
  Beeld- en datakompressie; geoutomatiseerde inligtingsdienste; intydse vertaling van gesproke taal; stelsels om kliëntebetalings te prosesseer; ens.

- ▷ Verdediging:
  Missielstuurmeganismes; teikenopsporing; gesigherkenning; ens.

- ▷ Vermaak:
  Animasie; spesiale effekte; markvooruitskatting; ens.

- ▷ Versekering:
  Evaluasie van polisuitbetalings; produkoptimering; ens.

- ▷ Vervaardiging:
  Proseskontrole; produkontwerp en -analise; proses- en masjiendiagnose; ontleding van sweisgehalte; dinamiese modellering van chemiese prosesstelsels; ens.

- ▷ Vervoer:
  Voertuigskedulering; roetestelsels; diagnostiese stelsels vir swaarvoertuigremstelsels; ens.

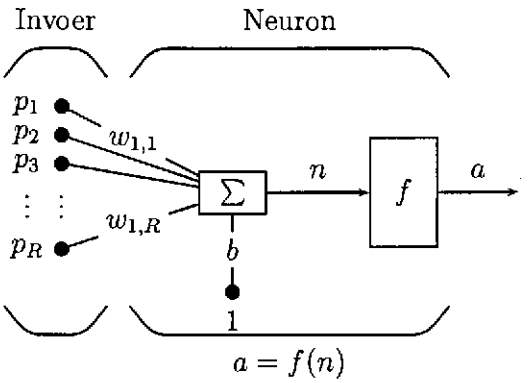
2.2 Argitektuur

'n Kunsmatige neurale netwerk is 'n netwerk van neurone wat met mekaar verbind is. Die neurone is eenvoudige eenhede wat inligting vanaf ander neurone (of van buite die netwerk) verwerk. Die resultaat word langs verbindings na ander neurone (of as uitvoer uit die netwerk) gestuur.

Elkeen van die verbindings dra 'n gewig wat die inligting wat by 'n neuron aankom in die verwerkingsproses weeg. Tydens die afrigtingsproses word gewigte op verbindings aangepas om uiteindelik 'n sekere resultaat te lewer.

Om die werking van 'n neurale netwerk te verstaan, is dit nodig om die netwerk visueel voor te stel. Hagan, Demuth en Beale [13] se notasie word gebruik.

2.3 'n Neurale netwerk met 'n enkele neuron



Figuur 2.3: Neurale netwerk met 'n enkele neuron

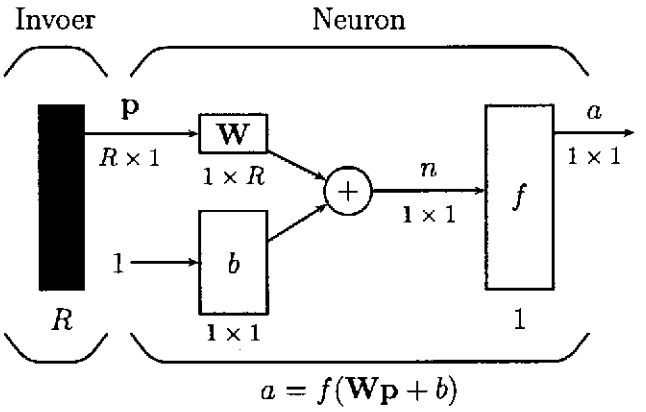
In Figuur 2.3 verskyn 'n neurale netwerk wat uit 'n enkele neuron bestaan. Die basiese komponente van hierdie neurale netwerk is die volgende:

- ▷ *Invoer*: Die invoer bestaan uit elemente wat die probleem onder beskouing verteenwoordig. In Figuur 2.3 bestaan die invoervektor $\mathbf{p}$ uit R elemente,

$$p_1,\ p_2,\ \ldots,\ p_R.$$

- *Belading*: Die neuron ontvang 'n invoer met waarde een. Die gewig b op die verbinding tussen die invoer en die neuron staan as 'n belading bekend.
- *Gewigte*: Elke verbinding tussen die invoer en die neuron dra 'n gewig. Die gewigte op die verbinding tussen die eerste, tweede, tot die R-de element van die invoervektor en neuron 1, word gegee deur
$$w_{1,1};\ w_{1,2};\ \ldots;\ w_{1,R}.$$
- *Netto invoer*: Elke invoerelement word met die gewig op die verbinding tussen die element en die neuron geweeg. Hierdie geweegde invoere en die belading word gesommeer om die netto invoer n te kry, met
$$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b.$$
- *Oordragfunksie*: Die oordragfunksie f word gebruik om die uitvoer van die neuron te bepaal. Dit is 'n lineêre of 'n nie-lineêre funksie van die netto invoer n. Die funksie word gekies om aan spesifikasies van die probleem onder beskouing te voldoen. Voorbeelde van oordragfunksies is
  - die lineêre oordragfunksie, met
  $$a = f(n) = n;$$
  - die streng beperkende trapfunksie, met
  $$a = f(n) = \begin{cases} 0 \text{ as } n < 0; \\ 1 \text{ as } n \geq 0; \end{cases}$$
  - die *log-sigmoidale* oordragfunksie, met
  $$a = f(n) = \frac{1}{1 + e^{-n}}.$$

Wanneer 'n neurale netwerk baie neurone bevat, wat gewoonlik die geval is, word dit moeilik, of selfs onmoontlik, om al die detail soos hierbo voor te stel. Ons gebruik dus 'n meer kompakte notasie. In Figuur 2.4 verskyn die enkelneuronnetwerk in kompakte notasie.



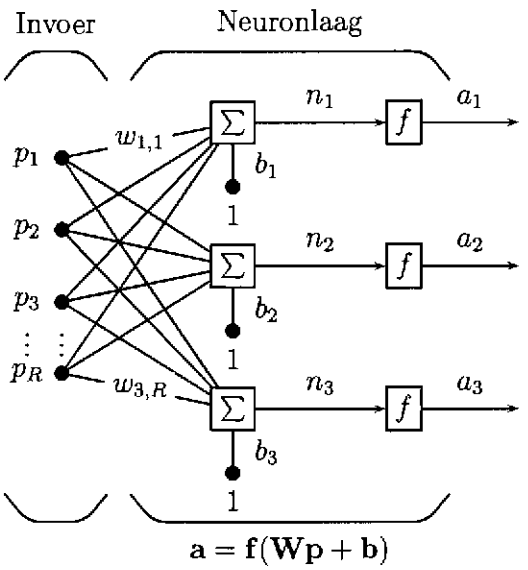
Figuur 2.4: Enkelneuronnetwerk in kompakte notasie

Let op die volgende besonderhede van die notasie:

- ▷ Die soliede vertikale balk aan die linkerkant stel die invoervektor $\mathbf{p}$ voor. Die dimensies van $\mathbf{p}$ verskyn onder die simbool $\mathbf{p}$ as $R \times 1$, met ander woorde 'n kolomvektor met R elemente. Die R onder die vertikale balk gee dus die grootte van die invoervektor. Let daarop dat die invoer nie as 'n neuron beskou word nie.
- ▷ $\mathbf{W}$ dui die matriks van gewigte op die verbindings aan. In hierdie enkelneuronnetwerk is $\mathbf{W}$ 'n $1 \times R$ matriks, met ander woorde 'n ryvektor met R elemente.
- ▷ 'n Beladingselement van waarde 1 word ingevoer. Die belading b is 'n enkele waarde.
- ▷ Die netto invoer van die neuron is $n = f(n) = \mathbf{Wp} + b$.
- ▷ Die oordragfunksie f bepaal die uitvoer van die neuron, naamlik $a = f(\mathbf{Wp} + b)$.
- ▷ Soos voorheen genoem, is die oordragfunksie 'n funksie van n wat gekies word om aan spesifikasies van die probleem onder beskouing te voldoen. In grafiese voorstellings van neurale netwerke word oordragfunksies gewoonlik deur ikone aangedui, soos byvoorbeeld  vir die lineêre oordragfunksie,  vir die streng beperkende trapfunksie en  vir die log-sigmoidale oordragfunksie.

2.4 'n Neuronlaag

'n Enkele neuron, selfs met baie invoerelemente, is gewoonlik nie in staat om probleme op te los nie. Ten minste 'n paar neurone wat parallel optree, is gewoonlik nodig. Ons noem so 'n groep parallelle neurone 'n neuronlaag. In Figuur 2.5 verskyn 'n neurale netwerk met een laag wat uit drie neurone bestaan.



Figuur 2.5: 'n Neuronlaag met 3 neurone

Hierdie neurale netwerk het die volgende eienskappe wat van die enkelneuronnetwerk verskil:

- ▷ Die invoer $\mathbf{p} = (p_1, p_2 \ldots, p_R)^T$ is, soos voorheen, 'n kolomvektor met R elemente.
- ▷ Die gewigmatriks bestaan uit 3 rye en kolomme R, met ander woorde die $3 \times R$ matriks

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,R} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,R} \\ w_{3,1} & w_{3,2} & \ldots & w_{3,R} \end{bmatrix}.$$

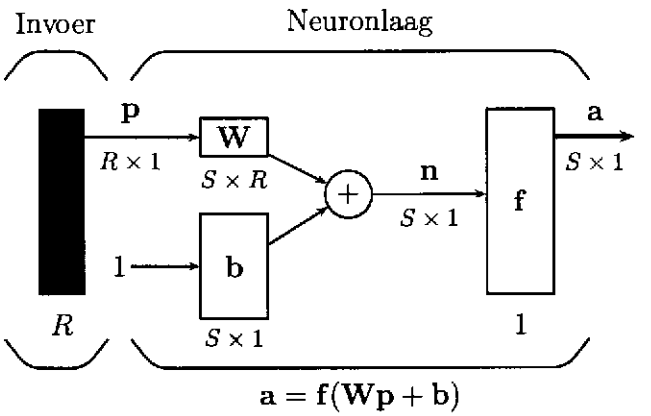
- ▷ Elke neuron het 'n belading wat in die beladingsvektor $\mathbf{b} = (b_1, b_2, b_3)^T$ verskyn.
- ▷ Elkeen van die neurone in die laag het sy eie netto invoer. Die netto-invoervektor word gegee deur

$$\mathbf{n} = (n_1, n_2, n_3)^T = \mathbf{Wp} + \mathbf{b}.$$

- ▷ Vir elkeen van die neurone is daar 'n oordragfunksie. Die uitvoer van die netwerk word dus gegee deur

$$\mathbf{a} = \mathbf{f}(\mathbf{Wp} + \mathbf{b}).$$

In Figuur 2.6 verskyn 'n neuronlaag met S neurone in kompakte notasie.



Figuur 2.6: 'n Neuronlaag in kompakte notasie

Let op die volgende besonderhede:

- ▷ Die invoer is weereens 'n vektor $\mathbf{p}$ met R elemente. Elkeen van hierdie elemente is met elke neuron verbind.
- ▷ Die gewigmatriks het S rye en R kolomme en word gegee deur

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,R} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,R} \\ \vdots & & & \vdots \\ w_{S,1} & w_{S,2} & \ldots & w_{S,R} \end{bmatrix}.$$

- ▷ Elke neuron i se geweegde invoere en beladings word bymekaargetel om sy netto invoer n_i te kry. Die verskillende n_i's vorm saam 'n netto invoervektor $\mathbf{n} = \mathbf{Wp} + \mathbf{b}$ met S elemente.

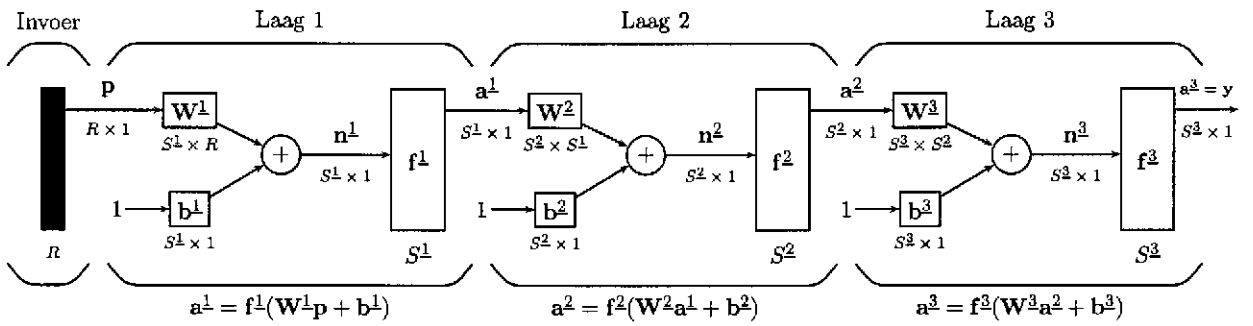
▷ Die uitvoer van die neuronlaag is 'n kolomvektor $\mathbf{a}$ met

$$\mathbf{a} = \mathbf{f}(\mathbf{n}) = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b}).$$

▷ Die neuronlaag bestaan uit S neurone en sluit die gewigmatriks, die vermenigvuldig- en sommasiebewerkings, die beladingsvektor en die oordragfunksies in.

2.5 Meervoudige neuronlae

'n Neurale netwerk kan uit verskeie lae bestaan. In Figuur 2.7 verskyn 'n netwerk met drie lae.



Figuur 2.7: 'n Neurale netwerk met 3 lae

Notasie: Om die komponente in die verskillende lae te onderskei, word die nommer van die laag waarin dit voorkom as boskrif gebruik. Die laagnommer word telkens onderstreep sodat dit nie met magte verwar word nie.

In neurale netwerke met meervoudige lae word tussen die volgende soorte lae onderskei:

▷ *Verborge lae*: 'n Neuronlaag waarvan die uitvoer na 'n ander laag in die netwerk gevoer word, word 'n verborge laag genoem.

▷ *Uitvoerlae*: 'n Neuronlaag waarvan die uitvoer ook die uitvoer van die neurale netwerk is, word 'n uitvoerlaag genoem.

In die bostaande drielaagnetwerk word die volgende komponente onderskei:

▷ Die invoer $\mathbf{p}$ is, soos voorheen, 'n kolomvektor met R elemente. (*Let wel:* Die invoer word nie as 'n laag beskou nie.)

▷ Laag 1:

- Die laag bestaan uit $S^{\underline{1}}$ neurone;
- Die gewigte verskyn in die $S^{\underline{1}} \times R$ matriks $\mathbf{W}^{\underline{1}}$;
- Die beladings verskyn in $\mathbf{b}^{\underline{1}}$;
- Die geweegde gewigte en beladings word gesommeer en die netto invoere verskyn in $\mathbf{n}^{\underline{1}}$;
- Die oordragfunksies verskyn in $\mathbf{f}^{\underline{1}}$;
- Die uitvoer verskyn in $\mathbf{a}^{\underline{1}}$.

- ▷ Die uitvoer van die eerste laag ($\mathbf{a}^{\underline{1}}$) word as invoer na die tweede laag gevoer. Laag 1 is dus 'n verborge laag.

- ▷ Laag 2:
  - Die laag bestaan uit $S^{\underline{2}}$ neurone;
  - Die gewigte verskyn in die $S^{\underline{2}} \times S^{\underline{1}}$ matriks $\mathbf{W}^{\underline{2}}$;
  - Die beladings verskyn in $\mathbf{b}^{\underline{2}}$;
  - Die netto invoere verskyn in $\mathbf{n}^{\underline{2}}$ met $\mathbf{n}^{\underline{2}} = \mathbf{W}^{\underline{2}}\mathbf{a}^{\underline{1}} + \mathbf{b}^{\underline{2}}$;
  - Die oordragfunksies verskyn in $\mathbf{f}^{\underline{2}}$;
  - Die uitvoer verskyn in $\mathbf{a}^{\underline{2}}$.

- ▷ Die uitvoer van die tweede laag ($\mathbf{a}^{\underline{2}}$) word as invoer na die derde laag gevoer. Laag 2 is dus ook 'n verborge laag.

- ▷ Laag 3 (Die uitvoerlaag):
  - Die laag bestaan uit $S^{\underline{3}}$ neurone;
  - Die gewigte verskyn in die $S^{\underline{3}} \times S^{\underline{2}}$ matriks $\mathbf{W}^{\underline{3}}$;
  - Die beladings verskyn in $\mathbf{b}^{\underline{3}}$;
  - Die netto invoere verskyn in $\mathbf{n}^{\underline{3}}$ met $\mathbf{n}^{\underline{3}} = \mathbf{W}^{\underline{3}}\mathbf{a}^{\underline{2}} + \mathbf{b}^{\underline{3}}$;
  - Die oordragfunksies verskyn in $\mathbf{f}^{\underline{3}}$;
  - Die uitvoer van hierdie laag is $\mathbf{a}^{\underline{3}}$. Dit is ook die uitvoer van die hele netwerk. Laag 3 is dus die uitvoerlaag.

Notasie: Daar word dikwels na 'n sekere neurale netwerk verwys deur sy samestelling te gee, naamlik die aantal invoerelemente, die aantal neurone in elk van die verborge lae en die aantal neurone in die uitvoerlaag (dit is die aantal uitvoerelemente). Bostaande drielaagnetwerk word as 'n $R{-}S^{\underline{1}}{-}S^{\underline{2}}{-}S^{\underline{3}}$ netwerk beskryf. 'n Tweelaagnetwerk met 301 invoerelemente, 80 neurone in die verborge laag en een neuron in die uitvoerlaag word, byvoorbeeld, as 'n $301 - 80 - 1$ netwerk beskryf.

2.6 Neurale-netwerkmodelle

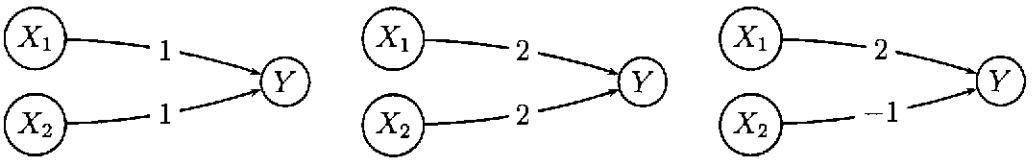
Oor die jare het neurale-netwerkmodelle ontwikkel van eenvoudige netwerke wat net lineêr skeibare probleme soos die logiese EN en OF funksies kan hanteer tot gesofistikeerde modelle wat byvoorbeeld die taak om 'n vragmotor met 'n wa agteruit tot teen 'n perron te stuur, kan beheer. In hierdie afdeling bespreek ons 'n paar van hierdie modelle met die klem op patroonherkenning.

2.6.1 McCulloch-Pitts

Warren McCulloch en Walter Pitts het in 1943 een van die eerste kunsmatige neurone beskryf [9]. Die belangrikste eienskap van hul neuronmodel is dat 'n geweegde som van die invoerseine met 'n drumpelwaarde vergelyk word om die uitvoer te bepaal. In Figuur 2.8 verskyn McCulloch-Pitts neurone wat die logiese funksies EN, OF en EN NIE uitvoer. (Die EN NIE-funksie is 'n kombinasie van die basiese funksies.)

Elkeen van hierdie neurone het 'n drumpelwaarde $\theta = 2$ en die gewigte (w_i met $i = 1, 2$) word op die verbindings aangedui. Indien die geweegde som van die invoer groter of gelyk aan die drumpel is,

is die uitvoer *ja* en dit word met 'n 1 aangedui. (In hierdie geval sê ons die neuron „vuur".) Indien die geweegde som egter kleiner as die drumpel is, is die uitvoer *nee* wat met 'n 0 aangedui word.



(a) EN (b) OF (c) EN NIE

Figuur 2.8: McCulloch-Pitts neurone vir logiese funksies EN, OF en EN NIE

Die logiese funksies EN, OF en EN NIE word deur die volgende invoerpare (p_1, p_2) en teikens (t) beskryf:

EN			OF			EN NIE		
p_1	p_2	t	p_1	p_2	t	p_1	p_2	t
1	1	1	1	1	1	1	1	0
1	0	0	1	0	1	1	0	1
0	1	0	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0

Beskou die EN neuron:

- Invoer (1; 1): $\sum_{i=1}^{2} p_i w_i = 1 \times 1 + 1 \times 1 = 2 = \theta$ dus is die uitvoer 1 (*ja*).
- Invoer (0; 1): $\sum_{i=1}^{2} p_i w_i = 0 \times 1 + 1 \times 1 = 1 < \theta$ dus is die uitvoer 0 (*nee*).
- Invoer (1; 0): $\sum_{i=1}^{2} p_i w_i = 1 \times 1 + 0 \times 1 = 1 < \theta$ dus is die uitvoer 0 (*nee*).
- Invoer (0; 0): $\sum_{i=1}^{2} p_i w_i = 0 \times 1 + 0 \times 1 = 0 < \theta$ dus is die uitvoer 0 (*nee*).

Beskou die OF neuron:

- Invoer (1; 1): $\sum_{i=1}^{2} p_i w_i = 1 \times 2 + 1 \times 2 = 4 > \theta$ dus is die uitvoer 1.
- Invoer (0; 1): $\sum_{i=1}^{2} p_i w_i = 0 \times 2 + 1 \times 2 = 2 = \theta$ dus is die uitvoer 1.
- Invoer (1; 0): $\sum_{i=1}^{2} p_i w_i = 1 \times 2 + 0 \times 2 = 2 = \theta$ dus is die uitvoer 1.
- Invoer (0; 0): $\sum_{i=1}^{2} p_i w_i = 0 \times 2 + 0 \times 2 = 0 < \theta$ dus is die uitvoer 0.

Beskou die EN NIE neuron:

- Invoer (1; 1): $\sum_{i=1}^{2} p_i w_i = 1 \times 2 + 1 \times -1 = 1 < \theta$ dus is die uitvoer 0.
- Invoer (0; 1): $\sum_{i=1}^{2} p_i w_i = 0 \times 2 + 1 \times -1 = -1 < \theta$ dus is die uitvoer 0.
- Invoer (1; 0): $\sum_{i=1}^{2} p_i w_i = 1 \times 2 + 0 \times -1 = 2 = \theta$ dus is die uitvoer 1.
- Invoer (0; 0): $\sum_{i=1}^{2} p_i w_i = 0 \times 2 + 0 \times -1 = 0 < \theta$ dus is die uitvoer 0.

McCulloch en Pitts het aangetoon dat netwerke wat uit hierdie neurone saamgestel word meeste berekenings- of logiese funksies kan hanteer. Selfs die logiese XOR (*exclusive or*) funksie wat kan nie lineêr skeibaar is nie (sien bladsy 36) kan opgelos word deur die drumpelwaardes reg te kies.

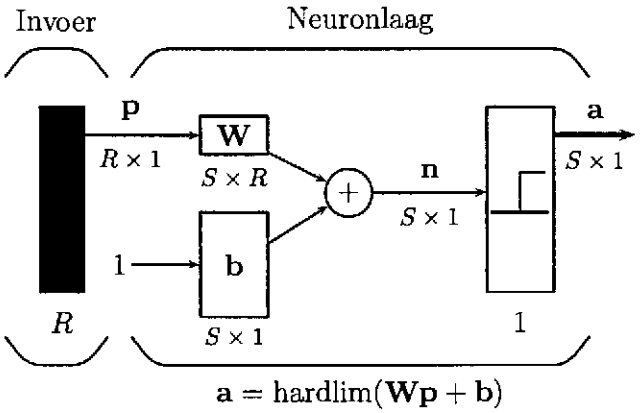
Die nadeel van hierdie netwerke is egter dat die parameters van die netwerke vir elke probleem spesifiek ontwerp moet word. Daar bestaan geen metode om dit te genereer nie.

2.6.2 Perseptron

Frank Rosenblatt en verskeie ander navorsers het in die laat 1950s 'n groep neurale netwerke wat perseptrone genoem word, ontwikkel [13]. Die neurone in hierdie netwerke was soortgelyk aan dié van McCulloch-Pitts maar van 'n afrigtingsreël waarmee perseptrone afgerig kon word om patroonherkenningsprobleme op te los, is bekendgestel.

Rosenblatt het bewys dat sy afrigtingsreël (die deltareël) altyd na die korrekte netwerkgewigte sal konvergeer indien daar 'n oplossing vir die probleem bestaan.

In Figuur 2.9 verskyn die algemene perseptronnetwerk.



Figuur 2.9: Algemene perseptron

Die gewigmatriks

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

kan geskryf word as

$$\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

waar

$${}_i\mathbf{w} = \begin{bmatrix} \mathbf{w}_{i,1} \\ \mathbf{w}_{i,2} \\ \vdots \\ \mathbf{w}_{i,R} \end{bmatrix}$$

die vektor is wat uit die elemente van die i-de ry van $\mathbf{W}$ bestaan.

Die i-de element van die netwerk se uitvoervektor kan nou geskryf word as

$$a_i = \text{hardlim}(n_i) = \text{hardlim}({}_i\mathbf{w}^T\mathbf{p} + b_i).$$

Die *hardlim*-oordragfunksie word gedefinieer as

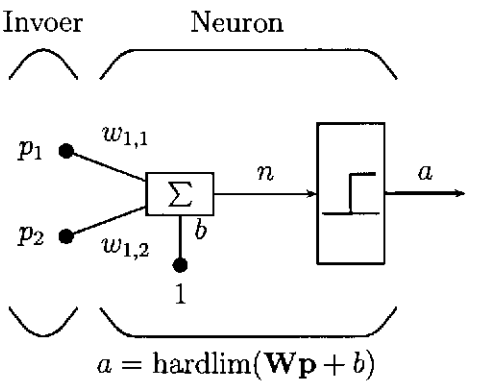
$$a = \text{hardlim}(n) = \begin{cases} 1 \text{ as } n \geq 0, \\ 0 \text{ andersins.} \end{cases}$$

Hieruit volg dat

- ▷ indien die produk van die i-de ry van die gewigmatriks en die invoervektor groter of gelyk aan $-b_i$ is, is die uitvoer 1; en
- ▷ indien die produk kleiner as $-b_i$ is, is die uitvoer 0.

Elke neuron in die netwerk verdeel die invoerruimte in twee areas. Daar bestaan dus 'n grens waar die uitvoer aan die een kant 1 sal wees (wat sê *ja* die invoer stem met die patroon ooreen) en aan die ander kant 0 (wat sê *nee* dit stem nie ooreen nie). Hierdie grens word die beslissingsgrens genoem.

Beskou die perseptronneuron met twee invoerelemente in Figuur 2.10.



Figuur 2.10: Perseptron – twee invoerelemente

Die uitvoer van hierdie netwerk word gegee deur

$$a = \text{hardlim}(\mathbf{Wp} + b) = \text{hardlim}({}_1\mathbf{w}^T\mathbf{p} + b) = \text{hardlim}(w_{1,1}p_1 + w_{1,2}p_2 + b).$$

Die beslissingsgrens word deur die invoervektore waarvoor die netto invoer n nul is, bepaal. Dit is

$$n = {}_1\mathbf{w}^T\mathbf{p} + b = w_{1,1}p_1 + w_{1,2}p_2 + b = 0.$$

Veronderstel dat

$$w_{1,1} = 1, \quad w_{1,2} = 1, \quad b = -1.$$

Die beslissingsgrens word nou gegee deur

$$p_1 + p_2 - 1 = 0,$$

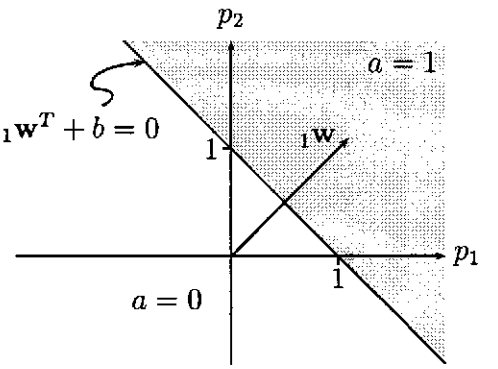
met ander woorde

$$p_2 = 1 \text{ as } p_1 = 0 \qquad \text{en} \qquad p_1 = 1 \text{ as } p_2 = 0.$$

Om te bepaal watter kant van die grens met 'n uitvoer van 1 ooreenstem, beskou ons enige invoerpunt, sê $\mathbf{p} = [2\ 1]^T$. Die netwerkuitvoer is

$$a = \text{hardlim}\left([1\ 1]\begin{bmatrix}2\\1\end{bmatrix} - 1\right) = \text{hardlim}(2) = 1.$$

Die uitvoer van die netwerk sal dus 1 wees vir al die invoerpunte regs van die beslissingsgrens in Figuur 2.11, en 0 vir alle invoerpunte links daarvan.



Figuur 2.11: Beslissingsgrens

Ons definieer die fout in 'n perseptron se uitvoer as

$$e = t - a.$$

Dit is die verskil tussen die teiken en die huidige uitvoer van die netwerk. Die afrigtingsreël vir die gewigte van perseptrone word gegee deur

$${}_1\mathbf{w}^{nuut} = {}_1\mathbf{w}^{oud} + e\mathbf{p} = {}_1\mathbf{w}^{oud} + (t-a)\mathbf{p}$$

en vir die belading deur

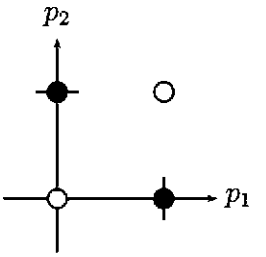
$$b^{nuut} = b^{oud} + e.$$

Hierdie afrigtingsreël kan uitgebrei word vir perseptrone met meervoudige neurone. In matriksnotasie word dit geskryf as

$$\mathbf{W}^{nuut} = \mathbf{W}^{oud} + \mathbf{e}\mathbf{p}^T \quad \text{en} \quad \mathbf{b}^{nuut} = \mathbf{b}^{oud} + \mathbf{e}.$$

Aangesien perseptrone lineêre beslissingsgrense (hipervlakke) lewer, kan slegs probleme wat lineêr skeibaar is daardeur opgelos word. Die logiese XOR funksie wat sê dat òf die eerste òf die tweede invoerelement 1 moet wees om aan die patroon te voldoen, word deur die volgende invoerpare en ooreenkomstige teikens verteenwoordig:

p_1	p_2	$\rightarrow$	t
1	1		0
1	0		1
0	1		1
0	0		0



Figuur 2.12: Die XOR funksie

In Figuur 2.12 verskyn die grafiese voorstelling van hierdie logiese funksie.

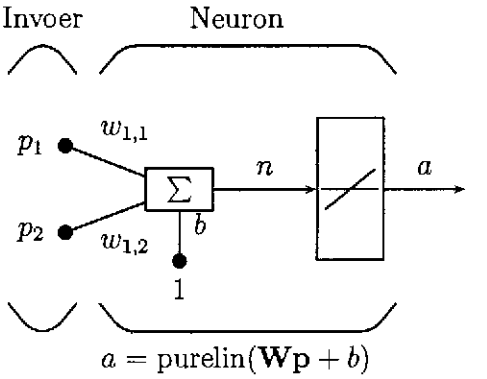
Dit is duidelik dat daar nie 'n lineêre grens bestaan wat hierdie probleem kan oplos nie. Hierdie onvermoë van die perseptron om probleme wat nie lineêr skeibaar is nie op te los, is in 1969 deur Minsky en Papert uitgelig. Dit het daartoe gelei dat belangstelling in neurale netwerke gedurende die 1970s sterk afgeneem het.

2.6.3 ADALINE

Bernard Widrow en een van sy studente Marcian Hoff het in 1960 die ADALINE (*ADAptive LInear NEuron*) netwerk en 'n afrigtingsreël wat hulle die LMS- (*Least Mean Square*) algoritme genoem het, bekendgestel [13].

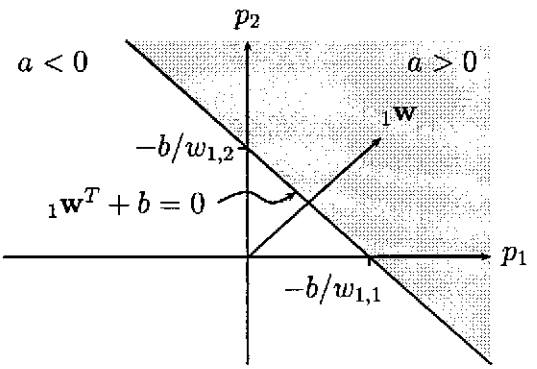
Die ADALINE-netwerk is dieselfde as die perseptron, behalwe dat die oordragfunksie lineêr is in plaas van streng beperkend. Die ADALINE kan ook net lineêr skeibare probleme oplos, maar die LMS-agrigtingsreël is kragtiger as die perseptron-afrigtingsreël. Alhoewel dit gewaarborg is dat die perseptronreël na 'n oplossing sal konvergeer, is dit sensitief vir geraas aangesien die patrone dikwels naby die beslissingsgrens lê. Die LMS-algoritme minimeer die gemiddelde kwadraatfout en probeer dus om die beslissingsgrens so ver as moontlik van die afrigtingspatrone te plaas.

In Figuur 2.13 verskyn 'n ADALINE-neuron met twee invoerelemente.



Figuur 2.13: ADALINE – invoervektor met twee elemente

Net soos by die perseptron word die beslissingsgrens deur ${}_1\mathbf{w}^T\mathbf{p} + b = 0$ gegee. Sien Figuur 2.14.



Figuur 2.14: Beslissingsgrens vir ADALINE (2 invoerelemente)

Die LMS-algoritme is ook, soos die perseptron-afrigtingsreël, 'n voorbeeld van afrigting met kontrole waar 'n stel voorbeelde van hoe die netwerk behoort op te tree, verskaf word. So 'n stel afrigtingsdata is

$$\{\mathbf{p}_1,\ \mathbf{t}_1\}, \{\mathbf{p}_2,\ \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q,\ \mathbf{t}_Q\},$$

waar $\mathbf{p}_q$ 'n invoer en $\mathbf{t}_q$ die ooreenstemmende teiken is. Soos elke invoer aan die netwerk gevoer word, word die netwerk se uitvoer met die teiken vergelyk. Die LMS-algoritme pas die gewigte en beladings van die ADALINE aan sodat die gemiddelde kwadraatfout geminimeer word.

Ter wille van eenvoud word al die parameters wat aangepas word (dit sluit die belading in) in een vektor geplaas, naamlik

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix}$$

en die beladingsinvoer (1) word as 'n komponent van die invoervektor ingesluit, sodat

$$\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}.$$

Die netto invoer kan nou geskryf word as

$$a = \mathbf{x}^T\mathbf{z}.$$

Die gemiddelde kwadraatfout is

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

waar die verwagte waarde oor alle afrigtingspare geneem word.

Widrow en Hoff het die gemiddelde kwadraatfout $F(\mathbf{x})$ deur

$$\hat{F}(\mathbf{x}) = (t(k) - a(k))^2 = e^2(k)$$

beraam. Die verwagte waarde van die kwadraatfout is deur die kwadraatfout in iterasie k vervang. In elke iterasie het ons dus 'n beraamde helling van die vorm

$$\hat{\nabla} F(\mathbf{x}) = \nabla e^2(k).$$

Die eerste R elemente van $\nabla e^2(k)$ is afgeleides met betrekking tot die netwerkgewigte, terwyl die $(R+1)$ste element die afgeleide met betrekking tot die belading is. Dus is

$$[\nabla e^2(k)]_j = \frac{\delta e^2(k)}{\delta w_{1,j}} = 2e(k)\frac{\delta e(k)}{\delta w_{1,j}} \quad \text{vir} \quad j = 1, 2, \ldots, R$$

en

$$[\nabla e^2(k)]_{R+1} = \frac{\delta e^2(k)}{\delta b} = 2e(k)\frac{\delta e(k)}{\delta b}.$$

Die parsiële afgeleide aan die einde van die eerste vergelyking is

$$\begin{aligned}\frac{\delta e(k)}{\delta w_{1,j}} &= \frac{\delta[t(k) - a(k)]}{\delta w_{1,j}} \\ &= \frac{\delta}{\delta w_{1,j}}[t(k) - ({}_1\mathbf{w}^T\mathbf{p}(k) + b)] \\ &= \frac{\delta}{\delta w_{1,j}}\left[t(k) - \left(\sum_{i=1}^{R} w_{1,i}p_i(k) + b\right)\right]\end{aligned}$$

waar $p_i(k)$ die i-de element van die invoervektor in die k-de iterasie is. Dit kan vereenvoudig word na

$$\frac{\delta e(k)}{\delta w_{1,j}} = -p_j(k).$$

Netso kan die finale term van die helling van die belading verkry word, naamlik

$$\frac{\delta e(k)}{\delta b} = -1.$$

Aangesien $p_j(k)$ en 1 elemente van die vektor $\mathbf{z}$ is, kan die helling van die kwadraatfout in iterasie k geskryf word as

$$\hat{\nabla} F(\mathbf{x}) = \nabla e^2(k) = -2e(k)\mathbf{z}(k).$$

Om dus die beraamde helling te bereken, is dit net nodig om die fout met die invoer te vermenigvuldig.

Deur die beraming van $\nabla F(\mathbf{x})$ in die steilste-dalingsalgoritme ([13], p. 9-2) te gebruik, kry ons

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\nabla F(\mathbf{x}).$$

(Hier is α die leertempo – 'n klein positiewe waarde.) As ons nou $\nabla F(\mathbf{x})$ met $\hat{\nabla} F(\mathbf{x})$ vervang, kry ons

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k)$$

of

$${}_1\mathbf{w}(k+1) = {}_1\mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k) \quad \text{en} \quad b(k+1) = b(k) + 2\alpha e(k).$$

Laasgenoemde twee vergelykings vorm die basis van die Widrow-Hoff- of LMS-algoritme.

Vir meervoudige neurone word die i-de ry van die gewigmatriks aangepas deur

$${}_i\mathbf{w}(k+1) = {}_i\mathbf{w}(k) + 2\alpha e_i\mathbf{p}(k) \quad \text{en} \quad b_i(k+1) = b_i(k) + 2\alpha e_i(k)$$

waar $e_i(k)$ die $i$de element van die fout in iterasie k is. In matriksnotasie is die LMS-algoritme

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\mathbf{e}(k)\mathbf{p}^T(k) \quad \text{en} \quad \mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k).$$

2.6.4 Perseptrone met meervoudige lae

Die eerste beskrywing van 'n algoritme om neurale netwerke met meervoudige lae af te rig, het blykbaar in Paul Werbos (1974) se tesis verskyn, soos verwys in [13] . Dit was egter nie voor die middel 1980s dat David Rumelhart, Geoffry Hinton en Ronald Williams (1986), David Parker (1985) en Yann Le Cun (1986) onafhanklik van mekaar die gebruik van meervoudige lae in neurale netwerke herontdek en gepubliseer het nie. Die terugpropagerings- (*backpropagation*) algoritme (wat op netwerke met meervoudige lae gegrond is) het bekend geword toe dit in die boek *Parallel Distributed Processing* opgeneem is waarin die werk van sielkundiges David Rumelhart en James McClelland bespreek is. Na die publikasie van hierdie boek het die belangstelling in neurale netwerke weer opgevlam.

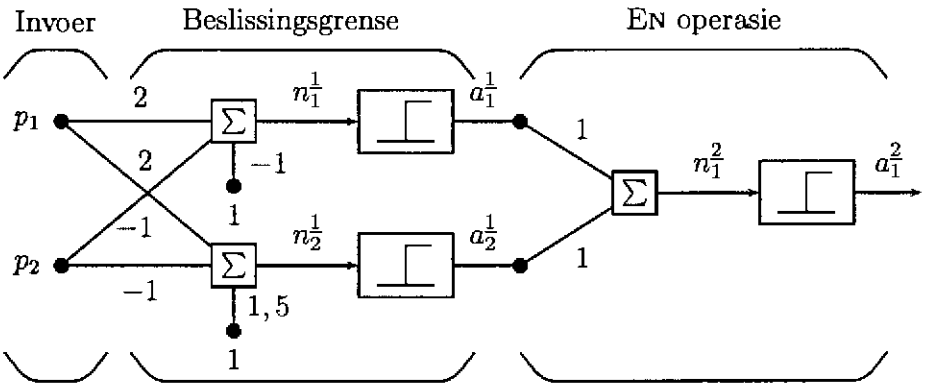
Perseptrone met meervoudige lae is bloot enkellaag-perseptrone wat aanmekaar gehaak is. Om die struktuur van 'n meerlaagnetwerk te identifiseer, word die notasie

$$R - S^{\underline{1}} - S^{\underline{2}} - S^{\underline{3}}$$

gebruik waar die aantal invoerelemente deur die aantal neurone in elke laag gevolg word.

Die bekende XOR probleem, wat op bladsy 36 verskyn, kan deur 'n tweelaagnetwerk opgelos word.

Die neurale netwerk wat hierdie probleem oplos verskyn in Figuur 2.15.

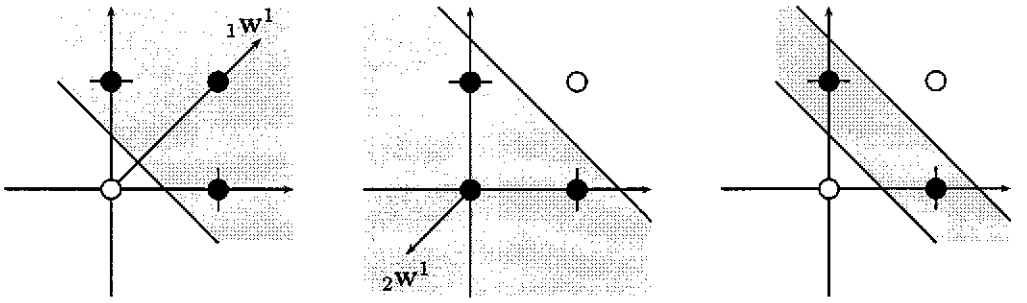


Figuur 2.15: Tweelaagnetwerk vir XOR

Die twee neurone in die eerste laag word gebruik om twee beslissingsgrense te skep. Die eerste grens skei $\mathbf{p}_1$ van die ander patrone en die tweede grens skei $\mathbf{p}_2$ van die ander patrone. Die tweede laag word dan gebruik om die twee grense deur middel van die EN-funksie te kombineer.

In Figuur 2.16 verskyn die afsonderlike beslissingsgrense en die uiteindelike gekombineerde beslissingsgebied wat die XOR probleem oplos.

(Daar bestaan ook ander oplossings vir hierdie probleem.)



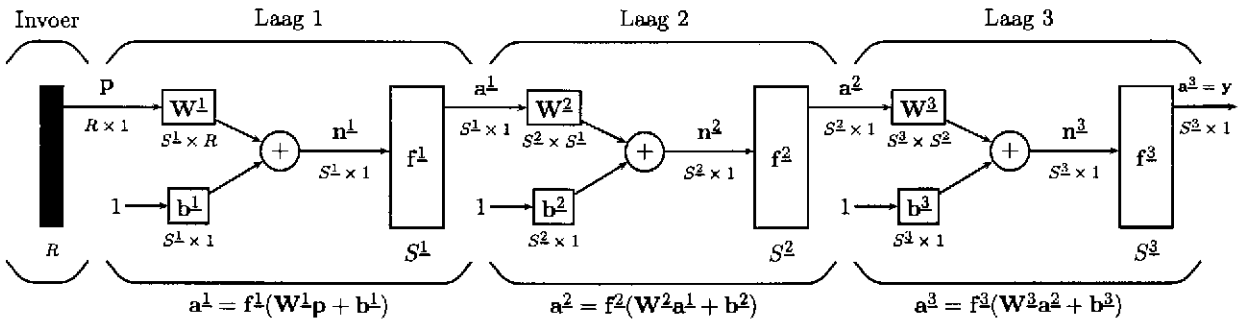
(a) Neuron 1 (Laag 1) (b) Neuron 2 (Laag 1) (c) Beslissingsgebied

Figuur 2.16: Bepaling van beslissingsgebied vir XOR

2.7 Terugpropagering

Terugpropagering (*backpropagation*) is 'n metode waar inligting oor die foute wat by die uitvoerneurone voorkom na die verborge neurone teruggevoer word. Hierdie afrigtingsmetode, wat as die terugpropagering van foute of die *algemene deltareël* bekend staan, is 'n metode wat die kwadraat van die totale fout van die uitvoer wat deur die netwerk bereken is, minimeer deur die steilste daling in die helling te gebruik.

Die drielaagnetwerk in Figuur 2.17 is 'n herhaling van Figuur 2.7. Die terugpropageringsalgoritme is 'n veralgemening van die LMS-algoritme wat in afdeling 2.6.3 bespreek is en is op neurale netwerke soos hieronder gebaseer.



Figuur 2.17: Drie-laag neurale netwerk

Die netwerkuitvoer word telkens met die teiken vergelyk. Die algoritme moet die netwerkparameters aanpas sodat die gemiddelde kwadraatfout ($F(\mathbf{x})$) geminimeer word, waar

$$F(\mathbf{x}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t} - \mathbf{a})^T(\mathbf{t} - \mathbf{a})].$$

Die gemiddelde kwadraatfout word beraam deur

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k) - \mathbf{a}(k))^T(\mathbf{t}(k) - \mathbf{a}(k)) = \mathbf{e}^T(k)\mathbf{e}(k)$$

waar die verwagte waarde van die kwadraatfout die kwadraatfout in die k-de iterasie is.

Die steilste-dalingsalgoritme vir die beraamde gemiddelde kwadraatfout is

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha \frac{\delta \hat{F}}{\delta w_{i,j}^{m}} \quad \text{en} \quad b_i^m(k+1) = b_i^m(k) - \alpha \frac{\delta \hat{F}}{\delta b_i^m}$$

waar α die leertempo is. Soos voorheen beskou ons die parsiële afgeleides aan die einde van hierdie uitdrukkings.

Die fout by netwerke met meervoudige lae is 'n indirekte funksie van die gewigte in die verborge lae. Daarom gebruik ons die kettingreël van calculus om die afgeleides te bepaal:

$$\frac{\delta \hat{F}}{\delta w_{i,j}^{m}} = \frac{\delta \hat{F}}{\delta n_i^m} \times \frac{\delta n_i^m}{\delta w_{i,j}^{m}} \quad \text{en} \quad \frac{\delta \hat{F}}{\delta b_i^m} = \frac{\delta \hat{F}}{\delta n_i^m} \times \frac{\delta n_i^m}{\delta b_i^m}.$$

Ons weet dat die netto invoer van laag m (n_i^m) 'n eksplisiete funksie van die gewigte en beladings in daardie laag is, naamlik

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m.$$

Die tweede term van elk van die parsiële afgeleides kan dus maklik bepaal word, naamlik

$$\frac{\delta n_i^m}{\delta w_{i,j}^m} = a_j^{m-1} \quad \text{en} \quad \frac{\delta n_i^m}{\delta b_i^m} = 1.$$

As ons nou $\hat{F}$ se sensitiwiteit ten opsigte van verandering in die i-de element van die netto invoer as

$$s_i^m \equiv \frac{\delta \hat{F}}{\delta n_i^m}$$

definieer, vereenvoudig ons vergelykings na

$$\frac{\delta \hat{F}}{\delta w_{i,j}^m} = s_i^m a_j^{m-1} \quad \text{en} \quad \frac{\delta \hat{F}}{\delta b_i^m} = s_i^m.$$

Die steilste-dalingsalgoritme kan nou geskryf word as

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1} \quad \text{en} \quad b_i^m(k+1) = b_i^m(k) - \alpha s_i^m.$$

In matriksnotasie is dit

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \quad \text{en} \quad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$$

waar

$$\mathbf{s}^m \equiv \frac{\delta \hat{F}}{\delta \mathbf{n}^m} = \left[\frac{\delta \hat{F}}{\delta n_1^m}, \frac{\delta \hat{F}}{\delta n_2^m}, \ldots, \frac{\delta \hat{F}}{\delta n_S^m} \right]^T.$$

Die basiese terugpropageringsalgoritme is 'n uitbreiding van die LMS-algoritme. Beide is benaderde steilste-dalingsalgoritmes wat die kwadraatfout minimeer. Die enigste verskil tussen hulle is die manier waarop die helling bereken word. Die terugpropageringsalgoritme gebruik die kettingreël om die afgeleides van die kwadraatfout met betrekking tot die gewigte en beladings in die verborge lae te bereken. Dit word terugpropagering genoem aangesien die afgeleides eers in die laaste laag van die netwerk bereken word en dan terugwaarts deur die netwerk gevoer word om die afgeleides in die verborge lae te bereken.

2.8 Ontwikkeling om terugpropagering te verbeter

Een van die groot probleme met terugpropagering is dat dit baie lank neem om af te rig. Sedert dit egter populêr begin raak het, is daar heelwat werk gedoen om die konvergensie van die algoritme te versnel. Hier volg 'n paar van die ontwikkelings [13]:

2.8.1 Lotafrigting (*batch training*)

Die parameters van die neurale netwerk word eers aangepas wanneer die hele stel afrigtingspare ingevoer is. Die gemiddelde van die hellings wat met elke afrigtingsvoorbeeld bereken is, word gebruik om 'n meer akkurate beraming van die helling te gee.

2.8.2 Momentum

Daar is waargeneem dat konvergensie verbeter indien die ossilasies in die trajek na die optimum uitgestryk kan word. Momentum laat toe dat 'n netwerk nie net op die huidige helling reageer nie, maar ook op onlangse neigings in die fout-oppervlakte. Deur soos 'n lae-deurgangsfilter op te tree, veroorsaak momentum dat klein veranderings in die fout-oppervlakte geïgnoreer word. Sonder momentum kan 'n netwerk in 'n vlak plaaslike minimum vashaak.

Wanneer die momentumfilter by die parameteraanpassings gevoeg word, kry ons die volgende vergelykings:

$$\triangle \mathbf{W}^m(k) = \gamma \triangle \mathbf{W}^m(k-1) - (1-\gamma)\alpha \mathbf{s}^m(\mathbf{a}^{m-1})^T \quad \text{en} \quad \triangle \mathbf{b}^m(k) = \gamma \triangle \mathbf{b}^m(k-1) - (1-\gamma)\alpha \mathbf{s}^m$$

waar γ die momentumkoëffisiënt is, met $0 \leq \gamma < 1$.

2.8.3 Veranderlike leertempo

Hier volg die reëls van 'n lotafrigtingsprosedure waar die leertempo volgens die prestasie van die algoritme verander:

1. Indien die kwadraatfout (oor die hele stel afrigtingsinvoere) na 'n gewigsaanpassing met meer as 'n voorafbepaalde persentasie ζ ($1\% < \zeta < 5\%$) toeneem, word die aanpassing geïgnoreer, die leertempo word met een of ander faktor ρ ($0 < \rho < 1$) vermenigvuldig en die momentumkoëffisiënt γ word gelyk aan nul gestel (indien dit gebruik word).

2. Indien die kwadraatfout met minder as ζ toeneem, word die gewigsaanpassing aanvaar, maar die leertempo bly onveranderd. As γ voorheen gelyk aan nul gestel is, kry dit weer die oorspronklike waarde.

3. Indien die kwadraatfout na 'n gewigsaanpassing afneem, word die aanpassing aanvaar en die leertempo word met 'n faktor $\eta > 1$ vermenigvuldig. As γ voorheen gelyk aan nul gestel is, kry dit weer die oorspronklike waarde.

2.8.4 Veerkragtige terugpropagering (*Resilient backpropagation* – RP)

Netwerke met meervoudige lae gebruik gewoonlik S-vormige (*sigmoidale*) funksies wat invoere met oneindige spanwydte in 'n eindige spanwydte „indruk". Sulke funksies het die eienskap dat hul helling na nul neig as die invoer groot word. Dit veroorsaak 'n probleem wanneer steilste daling gebruik word om die netwerk af te rig, aangesien die helling baie klein kan wees en dus klein veranderings in die gewigte en beladings veroorsaak al is dit nog ver van die optimale waardes.

Die doel van die veerkragtige-terugpropageringsalgoritme (RP) is om hierdie skadelike effek van die grootte van die parsiële afgeleides uit te skakel. Slegs die teken van die afgeleide word gebruik om die rigting van die opdatering van gewigte te bepaal – die grootte van die afgeleide het geen effek op die opdatering nie. Die volgende reëls geld vir die opdateringswaarde van gewigte:

1. Wanneer die afgeleide van die afrigtingsreël met betrekking tot 'n sekere gewig vir twee opeenvolgende iterasies dieselfde teken (positief of negatief) het, word die opdateringswaarde vir daardie gewig met 'n sekere faktor vermeerder.
2. Die opdateringswaarde van 'n sekere gewig word met 'n sekere (ander) faktor verminder indien die afgeleide met betrekking tot daardie gewig van een iterasie tot 'n volgende verander.
3. Indien die afgeleide nul is, bly die opdateringswaarde dieselfde.
4. Indien die afgeleide met betrekking tot 'n sekere gewig tussen positief en negatief ossilleer, word die opdateringswaarde verminder.

RP is in die algemeen vinniger as die standaard steilste-dalingsalgoritmes en vereis slegs matige toename in geheuevereistes. Die opdateringswaardes moet egter vir elke gewig en belading gestoor word, wat ekwivalent is aan die stoor van die hellings.

2.8.5 Verwante-hellingalgoritmes (*Conjugate gradient*)

Die gewone terugpropageringsalgoritme pas gewigte aan in die rigting waarin die afrigtingsfunksie die vinnigste afneem. Alhoewel die funksie die vinnigste langs die negatiewe helling afneem, lewer dit egter nie die vinnigste konvergensie nie. Met die verwante-hellingsalgoritmes word 'n soektog langs verwante rigtings uitgevoer, wat gewoonlik vinniger konvergensie as die steilste-dalingsrigtings tot gevolg het.

In die eerste iterasie van verwante-hellingsalgoritmes word in die steilste-dalingsrigting gesoek, naamlik

$$\mathbf{p}_0 = -\mathbf{g}_0.$$

Daarna word 'n lynsoektog uitgevoer om te bepaal wat die optimale afstand in die huidige soekrigting is, naamlik

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k.$$

Die volgende soekrigting word so bepaal dat dit verwant is aan die vorige soekrigtings – die nuwe steilste-dalingsrigting word met die vorige soekrigting gekombineer, naamlik

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}.$$

Verskillende variasies van die verwante-hellingalgoritme word onderskei deur die manier waarop die konstante β_k bereken word.

Hier volg vier sulke algoritmes:

Fletcher-Reeves Opdatering (CGF)

Opdatering geskied deurdat die vierkant van die norm van die huidige helling deur die vierkant van die norm van die vorige helling gedeel word:

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}.$$

Polak-Ribiére Opdatering (CGP)

Die binneproduk van die vorige verandering in helling en die huidige helling word deur die vierkant van die norm van die vorige helling gedeel:

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}.$$

Powell-Beale Herstelling (CGB)

In alle verwante-hellingsmetodes word die soekrigting van tyd tot tyd (gewoonlik as die aantal iterasies gelyk aan die aantal netwerkparameters is) na die negatief van die helling herstel. Daar bestaan herstelmetodes wat die effektiwiteit van afrigting kan verbeter. In Powell-Beale se metode word die helling herstel as daar baie min ortogonaliteit tussen die huidige en die vorige helling oorbly. Dit word soos volg getoets:

$$\left|\mathbf{g}_{k-1}^T \mathbf{g}_k\right| \leq 0,2 \parallel \mathbf{g}_k \parallel^2 .$$

As hierdie voorwaarde bevredig word, word die soekrigting na die negatief van die helling herstel.

Geskaalde verwante-hellingmetode (SCG)

Elk van die vorige algoritmes vereis dat 'n lynsoektog met elke iterasie uitgevoer word. Dit vereis baie berekenings, aangesien die uitvoer vir alle invoere 'n paar keer vir elke soektog bereken moet word. Die geskaalde-hellingsalgoritme is deur Moller ontwikkel om hierdie tydrowende lynsoektog uit te skakel. Dit is 'n komplekse algoritme wat daarop neerkom dat die vertrouensarea-benadering (wat in Levenberg-Marquardt algoritme gebruik word) met die verwante-hellingbenadering gekombineer word.

2.8.6 Vroeë beëindiging

Een van die groot probleme wat tydens afrigting van 'n neurale netwerk kan voorkom, is oorpassing. Die netwerk word afgerig totdat die fout baie klein is. Met nuwe, onbekende data lewer dit egter weer groot foutwaardes. Die netwerk het dus die afrigtingsvoorbeelde gememoriseer, maar het nie geleer om vir nuwe situasies te veralgemeen nie.

Een metode om veralgemening te verbeter, word vroeë beëindiging (*early stopping*) genoem. Hierdie tegniek behels dat die beskikbare data in twee dele verdeel word. Die eerste deel (gewoonlik die

grootste) word gebruik om die netwerk mee af te rig, terwyl die tweede gebruik word om die netwerk se werking op sekere tye te valideer. Validasie kom daarop neer dat die gesimuleerde uitvoer van die netwerk van tyd tot tyd met nuwe, onbekende data bepaal word. Hierdie uitvoer word dan met die teiken vergelyk om 'n validasiefout te bereken. Die validasiefout sal gewoonlik gedurende die eerste fase van afrigting afneem (soos die afrigtingsfout ook afneem). Wanneer die netwerk egter begin om die afrigtingsdata te memoriseer, sal die validasiefout begin toeneem. Indien die validasiefout vir 'n gespesifiseerde aantal iterasies toeneem, word afrigting gestaak en die gewigte en beladings waar die validasiefout 'n minimum was, word in die finale netwerk gebruik.

2.9 Neurale netwerke wat afkapping doen

Die toepaslikheid van neurale netwerke as tegniek vir die afkappingsprobleem is reeds deur die volgende groepe navorsers ondersoek:

- Søren Brunak en Benny Lautrup (Engels),
- Pavel Smerž en Petr Sojka (Tsjeggies),
- Bernd Fritzke (Duits), en
- Walter Daelemans en Antal van den Bosch (Nederlands).

Elkeen se metodiek en resultate word vervolgens kortliks bespreek.

2.9.1 Søren Brunak en Benny Lautrup

Søren Brunak en Benny Lautrup beskryf in hul boek *Neural networks: Computers with intuition* [5] hoe perseptrone gebruik is om die taak van afkapping in Engels aan te pak.

Doelwit

Die perseptron moet besluit of dit reg is (of nie) om 'n woord in 'n sekere punt tussen twee opeenvolgende letters af te kap – dus 'n klassifiseringstaak. Die besluit word op grond van die konteks van letters rondom die punt onder beskouing geneem. Die konteks word tot vier letters weerskante van die afkappingspunt beperk. Die perseptron beskou dus die woord deur 'n *venster* van agt letters wyd.

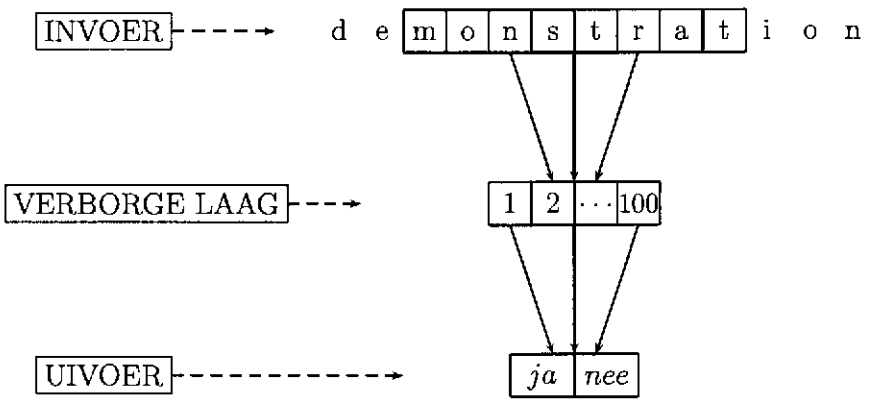
Voorstelling

Daar moet duidelik tussen die 26 letters van die alfabet wat vir Engels gebruik word, onderskei kan word. Elke letter word deur 'n blok van 26 neurone, waarvan net een sal vuur terwyl die ander onaktief bly, verteenwoordig. Elkeen van die 26 letters het dus sy eie unieke vuurpatroon.

As die invoervenster uit 8 letters bestaan, sal die invoerlaag van die perseptron uit 8 blokke van 26 neurone elk bestaan – dus 208 neurone.

Argitektuur

Die argitektuur van die netwerk wat gebruik is, verskyn in Figuur 2.18.



Figuur 2.18: Die afkappingsperseptron

Die perseptron het twee uitvoerneurone wat die volgende antwoorde verteenwoordig: *ja* as die woord in die betrokke posisie afgekap kan word en *nee* indien nie. Die antwoord word deur die uitvoerneuron met die hoogste aktiwiteit bepaal. Die neurone het kontinue aktiwiteite wat enige waardes tussen nul en een kan aanneem. As die *ja*-neuron se aktiwiteit 0,63 is en die *nee*-neuron s'n is 0,37, is die antwoord *ja*. (Wenner kry alles.)

Om te besluit hoeveel neurone in die verborge laag moet wees is baie moeilik, aangesien niemand weet hoe kompleks die afkappingsprobleem is nie. Dit moet eksperimenteel bepaal word. Brunak en Lautrup het besluit om 100 neurone in die verborge laag te gebruik. Dit impliseer dat die perseptron $100 \times 208 + 2 \times 100 = 21\,000$ sinapsgewigte en $100 + 2 = 102$ drumpelwaardes bevat. Die netwerk bevat dus 21 102 veranderlike parameters.

Afrigting

'n Steekproef van 15 699 woorde wat willekeurig uit die Engelse woordeboek *Collins Gem: Word Speller and Divider* (1973) getrek is, is gebruik om die netwerk af te rig. Elke woord is letter vir letter deur die venster getrek. Die woorde het 135 902 letters bevat en daaruit is 120 203 verskillende afrigtingsvoorbeelde gegenereer – elke woordfragment het uit hoogstens 8 letters bestaan.

Die aanvanklike gewigte en drumpelwaardes is ewekansig toegeken. Met elke invoer is die fout bepaal en gebruik om die gewigte so aan te pas dat die antwoord nader aan die verlangde teiken beweeg. Nadat al die afrigtingsvoorbeelde 300 keer aan die perseptron gevoer is, dus ongeveer 36 miljoen individuele invoere, kon die perseptron 99,8% van die antwoorde met die afrigtingsdata regkry. (Sonder 'n verborge laag kon die perseptron slegs 83% reg doen.)

Veralgemening

Dit is belangrik om te bepaal tot watter mate die perseptron afkappingspunte in woorde wat dit nog nooit teengekom het nie, reg sal kan bepaal. Die netwerk was in staat om 94,8% van 2 110 nuwe

woorde reg af te kap.

Kunsmatige woorde met dieselfde letterfrekwensies en byna dieselfde korrelasies tussen letters as regte Engelse woorde is gegenereer. Voorbeelde van sulke woorde met hul afkappings verskyn in die volgende tabel:

`prodarous`	`pro-dar-ous`
`urulonital`	`uru-lon-i-tal`
`romeraty`	`ro-mer-a-ty`
`ricoxtate`	`ri-cox-tate`

Hierdie pseudo-Engelse woorde het 'n vreemde Shakespearagtige klank en is beslis verwant aan Engels. Die perseptron kon hierdie woorde redelik goed in lettergrepe verdeel.

'n Interessante verduideliking

Brunak en Lautrup beskryf die neurale netwerk as 'n klein maatskappy.

- ▷ Die 208 invoerneurone sit by ontvangs, maak die pos oop en stuur dit na die volgende vlak sonder om enige verandering daaraan aan te bring.
- ▷ Die 100 verborge neurone tree as 'n groep klerke op. Elkeen ondersoek elke posstuk vir spesifieke eienskappe en som sy/haar bevindings oor die inkomende boodskap in 'n enkele syfer op. Die inligting oor die inkomende woordfragment word nou deur 100×208 syfers verteenwoordig. Die res van die inligting word geïgnoreer.
- ▷ Twee besluitnemers (die uitvoerneurone) gebruik uiteindelik die opsommende inligting om te besluit of die antwoord *ja* of *nee* moet wees.

Die verwerking van inligting bestaan daaruit om oorbodige inligting weg te gooi, eerder as om nuwe inligting te skep. In hierdie voorbeeld word inligting gelyktydig deur 100 verskillende agente „weggegooi". Elke verborge neuron lewer sy opsomming sonder om te weet wat die ander „klerke" in hul opsommings skryf. As gevolg van hierdie parallelle werking tussen die klerke is neurale netwerke in beginsel in staat om inligting baie vinnig te verwerk.

In die menslike brein stuur miljarde neurone deurgaans „opsommings" aan mekaar. In die neuronnetwerk van die brein word nie net vorentoevoer gebruik nie en die onderskeiding tussen „klerke" en „besluitnemers" is glad nie duidelik nie. Die brein bevat ontelbare terugvoerlusse wat *tyd* baie belangriker maak as in die naïewe model wat vir die afkappingsprobleem gebruik is. 'n Perseptron gebruik 'n eindige aantal berekeningstappe terwyl die brein 'n leeftyd mag neem.

2.9.2 Pavel Smerž en Petr Sojka

Pavel Smrž en Petr Sojka het 'n neurale netwerk ontwikkel om afkapping in Tsjeggies te doen [25]. Afkapping in Tsjeggies word volgens lettergrepe gedoen met etimologiese uitsonderings. Afkapping word tussen 'n voorvoegsel en die stam en op die grens tussen saamgestelde woorde verkies.

Invoer

Invoer tot die netwerk is telkens 'n reeks van sewe opeenvolgende letters uit 'n woord wat vir afrigting gebruik word. Die middelste letter in die reeks is die een waarvoor uitvoer gegee moet word. Die drie letters aan weerskante verskaf die konteks wat help om die afkappingspunt te bepaal. Individuele woorde word deur die *venster* beweeg sodat elke letter, behalwe die laaste twee, in die sentrale posisie beskou word. Spasies word voor en na die woord ingevoeg waar nodig.

Vir elkeen van die sewe letterposisies in die invoer het die netwerk 'n stel van 43 invoerkomponente: een vir elk van die 41 letters in Tsjeggies, een vir letters uit ander tale en een vir 'n spasie. Daar is dus $43 \times 7 = 301$ invoerkomponente.

Afrigting

Die volgende twee stelle woorde met afkappingspunte is gebruik om afrigtingspare te genereer:

1. 'n Stel van 169 888 woorde. Hierdie stel woorde het baie foute bevat.
2. 'n Stel van 78 809 woorde wat met die letter *m* begin. Hierdie stel woorde het minder foute as die eerste stel bevat.

Die eerste stel data is in 170 dele, elk met 1 000 woorde verdeel. In totaal is 1 581 183 afrigtingspatrone gegenereer.

Netwerke met verskillende aantal neurone in die verborge laag is afgerig, naamlik 301-30-1 en 301-100-1 netwerke.

Die netwerke is met elke deel van die afrigtingsdata afgerig totdat die netwerkfout onder 0,1 was of totdat 100 siklusse bereik is. Die leertempo is aanvanklik op 0,7 gestel en daarna met 0,1 verminder tot die finale waarde van 0,3 bereik is, wat verderaan gebruik is.

Die afrigtingsproses het baie lank geneem. Dit het omtrent 17 dae geneem om die 301-30-1 netwerk op 'n *Sun SparcStation 10* af te rig. (Dit sluit nie die tyd in om die afrigtingsdata te genereer nie.) Om die 301-100-1 netwerk af te rig, is die super rekenaar *Silikon Graphics POWER Challenge L* gebruik. Ten spyte van die kragtige rekenaar het dit omtrent 18 dae geneem om af te rig.

Die resultate verskyn in Tabel 2.1.

Die volgende twee soorte foute (wat 'n probleem by lettersetting in smal kolomme veroorsaak) is geïdentifiseer:

- ▷ Die afkapping kom in 'n posisie voor waar die woord nie afgekap kan word nie. Dit is die ergste van die twee soorte foute.
- ▷ Die toelaatbare afkappingspunt kan nie bepaal word nie.

STATISTIEK		(1581183 patrone)
verkeerd:	3,43%	(54282 patrone)
reg:	96,57%	(1526901 patrone)

(a) 301-30-1 netwerk

STATISTIEK		(1581183 patrone)
verkeerd:	3,26%	(51599 patrone)
reg:	96,74%	(1529584 patrone)

(b) 301-100-1 netwerk

Tabel 2.1: Resultate van afrigting met 1 581 183 afrigtingspatrone

Die grootste probleem met die afrigting van 'n neurale netwerk om afkapping te doen, is om goeie afrigtingsdata te kry. Dit is onrealisties om alle foute te vermy, maar dit is nodig om patrone waarin die aantal verkeerde afkappings 'n minimum is, te gebruik.

Die lêer met 169 888 woorde het baie foute bevat. Toe die netwerk dus met hierdie data getoets is, is woorde wat reg afgekap is as verkeerd beoordeel. Ander foute het hoofsaaklik voorgekom in woorde wat uitsonderings op die afkappingsreëls is – veral woorde wat uit ander tale oorgeneem is.

Om te bepaal of 'n netwerk al die uitsonderings kan leer, is die 54 282 afrigtingspatrone wat deur die 301-30-1 netwerk verkeerd afgekap is as een groot afrigtingsdatastel gebruik. 'n Ander 301-30-1 netwerk is hiermee afgerig. Die leertempo is stapgewys van 0,8 tot 0,2 verminder. Na 100 siklusse het die netwerk op vier na al die patrone geleer.

Invloed van kodering

Die letters is as syfers uit die versameling $\{0,02;\ 0,04;\ \ldots;\ 0,98\}$ gekodeer. Aanvanklik is kodes volgens die alfabet toegeken. Sien Tabel 2.2.

␣	a	á	b	c	č	d
0,02	0,04	0,06	0,08	0,10	0,12	0,14
d'	e	é	ě	f	g	h
0,16	0,18	0,20	0,22	0,24	0,26	0,28
i	í	j	k	l	m	n
0,30	0,32	0,34	0,36	0,38	0,40	0,42
ň	o	ó	p	q	r	ř
0,44	0,46	0,48	0,50	0,52	0,54	0,56
s	š	t	t'	u	ů	ú
0,58	0,60	0,62	0,64	0,66	0,68	0,70
v	w	x	y	ý	z	ž
0,72	0,74	0,76	0,78	0,80	0,82	0,84

Tabel 2.2: Kodering volgens alfabet

'n Ander koderingsmetode is later gebruik om die resultate te probeer verbeter. Die vokale is as klein getalle tussen 0,02 en 0,28 gekodeer en die konsonante (r en l uitgesluit) as getalle tussen 0,50 en 0,89. Die letters r en l is konsonante maar kan lettergrepe in Tsjeggies vorm. Dit is as 0,40 en 0,42 gekodeer om dit van ander konsonante te skei. Die spasiekarakter is as 0,34 gekodeer – tussen vokale en konsonante. Hierdie kodering verskyn in Tabel 2.3.

a	á	e	é	ě	i	í
0,02	0,04	0,06	0,08	0,10	0,12	0,14
o	ó	u	ú	ů	y	ý
0,16	0,18	0,20	0,22	0,24	0,26	0,28
		␣			r	l
		0,34			0,40	0,42
			b	c	č	d
			0,50	0,52	0,54	0,56
d'	f	g	h	j	k	m
0,58	0,60	0,62	0,64	0,66	0,68	0,70
n	ň	p	q	ř	s	š
0,72	0,74	0,76	0,78	0,80	0,82	0,84
t	t'	v	w	x	z	ž
0,86	0,88	0,90	0,92	0,94	0,96	0,98

Tabel 2.3: Kodering met vokale en konsonante geskei

Die resultate van netwerke waar hierdie verskillende koderings gebruik is, verskyn in Tabel 2.4. Netwerke met twee verborge lae is gebruik, naamlik 7-30-9-2 netwerke.

STATISTIEK		(8411 patrone)
verkeerd:	14,12%	(1188 patrone)
reg:	85,88%	(7223 patrone)

(a) Kodering volgens Tabel 2.2

STATISTIEK		(8411 patrone)
verkeerd:	3,41%	(287 patrone)
reg:	96,59%	(8124 patrone)

(b) Kodering volgens Tabel 2.3

Tabel 2.4: Resultate van afrigting met 1 581,183 afrigtingspatrone

Inligting oor die soort letter het dus die netwerk gehelp om te leer. Die alternatiewe kodering van r en l het ook gehelp. Die resultate kan waarskynlik verder verbeter word deur byvoorbeeld c en h, wat in Tsjeggies as een klank uitgespreek word en nooit tydens afkapping geskei word nie, saam deur 'n spesiale syfer wat anders as die kode vir beide letters afsonderlik is, te kodeer.

Verskillende netwerkkonfigurasies

Netwerke van die vorm 301-30-1, 301-60-1, 7-30-1, 7-30-9-2 en 7-60-2 is vergelyk. Die kodering soos in Tabel 2.3 is gebruik in die geval van netwerke met 7 invoerelemente. Die volgende resultate is verkry:

- ▷ Geen betekenisvolle verskil in prestasie is tussen die 301-30-1 en 301-60-1 netwerke waargeneem nie. Dieselfde resultaat is vroeër met 301-30-1 en 301-100-1 netwerke met ander afrigtingsdata verkry.

- ▷ Die 7-30-9-2 netwerk met twee verborge lae het beter resultate as die 7-60-2 netwerk gelewer alhoewel dit minder neurone en verbindings as laasgenoemde het.

▷ Die veralgemeningsvermoë van die 7-30-9-2 en 301-30-1 netwerke is vergelyk. Die netwerke is met 'n deelversameling van 1 000 woorde met korrekte lettergreepverdeling afgerig en toe met die stel van 78 809 woorde getoets. Die resultate verskyn in Tabel 2.5.

STATISTIEK		(648928 patrone)
verkeerd:	4,91%	(31886 patrone)
reg:	95,09%	(647042 patrone)

(a) Die 7-30-9-2 netwerk

STATISTIEK		(648928 patrone)
verkeerd:	2,78%	(18057 patrone)
reg:	97,22%	(630871 patrone)

(b) Die 301-30-1 netwerk

Tabel 2.5: Resultate van afrigting met 1 581,183 afrigtingspatrone

As die persentasie foute in die toetsdata in aanmerking geneem word, is die persentasie woorde wat deur die 301-30-1 netwerk verkeerd afgekap is baie laag.

Afkapping met slegs lettergreepaanduiding

In die laaste plek is getoets hoe goed 'n netwerk presteer indien slegs die soort letters (vokale of konsonante) verskaf word. Konsonante is as 0 gekodeer, vokale as 1 en spasies as 0,5. Die netwerk 7-30-1 is gebruik. Die resultate het getoon dat lettergreepafkapping 'n dominante rol in Tsjeggies speel, maar aangesien dit 'n fout van ongeveer 6% gelewer het, is dit duidelik dat afkapping wat net op lettergreepaanduiding gebaseer is onbevredigende resultate sal lewer.

2.9.3 Bernd Fritzke

Bernd Fritzke het 'n neurale netwerk afgerig om afkapping in Duits te doen [11]. Afkapping in Duits is 'n moeilike probleem, hoofsaaklik as gevolg van die voorkoms van saamgestelde woorde. Bestaande oplossings is gewoonlik reëlgebaseerd of maak van groot woordeboeke gebruik. Die resultate is, in teenstelling met Engels, dikwels nie bevredigend nie.

Die netwerkargitektuur

Die argitektuur van die netwerk het ooreenkomste met die een wat deur Sejnowski en Rosenberg gebruik is vir NETtalk. Dit het die volgende eienskappe:

▷ Dit ontvang invoer wat uit inligting van 8 aanliggende letters ('n *venster*) in 'n woord bestaan. Elke woord in die afrigtingsdatalêer word na verskeie pare invoer-uitvoerpatrone omgeskakel. So word die Duitse woord *ROTATION* wat as *RO-TA-TI-ON* afgekap word, na die volgende vyf patrone omgeskakel:

	invoerinligting								uitvoerinligting
patroon 1:	-	-	R	O	T	A	T	I	ja
patroon 2:	-	R	O	T	A	T	I	O	nee
patroon 3:	R	O	T	A	T	I	O	N	ja
patroon 4:	O	T	A	T	I	O	N	-	nee
patroon 5:	T	A	T	I	O	N	-	-	ja

afkappingsposisie

Leë posisies in die invoerpatrone word deur spasies gevul. Letters buite die venster word nie oorweeg nie. In Duits word enkele letters nooit afgekap nie. Die afkappingsposisie na die eerste en voor die laaste letter word dus nie oorweeg nie.

Elke karakterposisie in die woord word deur n neurone (met n die grootte van die alfabet) gekodeer. In Duits word die gewone alfabet van 26 letters plus 4 ekstra karakters gebruik. Saam met die spasiekarakter is daar 31 verskillende letters. 'n Letter word gekodeer deur die ooreenstemmende neuron gelyk aan 1 te stel en die ander 30 gelyk aan 0 (1-uit-n-kodering). Dit word vir elk van die agt invoerposisies gedoen. Ons het dus $8 \times 31 = 248$ neurone nodig om die invoer te kodeer.

- Die verborge laag bevat 40 neurone. Elke invoerneuron het 'n verbinding na elke neuron in die verborge laag wat elkeen weer met die uitvoerneuron verbind is.

- Die uitvoerlaag bestaan uit 'n enkele neuron wat aandui of afkapping by die afkappingsposisie tussen die vierde en vyfde letter van die invoervenster moontlik is. *Ja* is as 1 gekodeer en *nee* as 0.

- Daar is 'n enkele terugvoerverbinding van die uitvoerneuron na 'n ekstra invoerneuron. Hierdie verbinding is ingevoer om die netwerk in staat te stel om die konteksinligting te gebruik as afkapping in die vorige posisie moontlik was. (In Duits bestaan elke lettergreep uit ten minste twee letters. Aanliggende afkappingsposisies is dus nie moontlik nie.) Die terugvoerverbinding dra die uitvoerwaarde aan die ekstra invoerneuron oor voor elke nuwe patroon ingevoer word. Hierdie soort netwerk word 'n „sekwensiële netwerk" genoem. Eksperimente met en sonder terugvoer het getoon dat netwerke met terugvoer betekenisvol beter presteer as daarsonder. [*Opmerking*: Die sekwensiële aard van die netwerk verleen voorrang aan die eerste breekpunt wat bepaal is. Indien die tweede breekpunt die korrekte een sou wees, gaan die netwerk dit nie identifiseer nie!]

- Standaardterugpropagering met momentum is as afrigtingsreël gebruik. Die *log*-funksie $f(x) = \frac{1}{1+e^{-x}}$ is as oordragfunksie is gebruik.

Die volledige netwerkargitektuur verskyn in Figuur 2.19.

Data

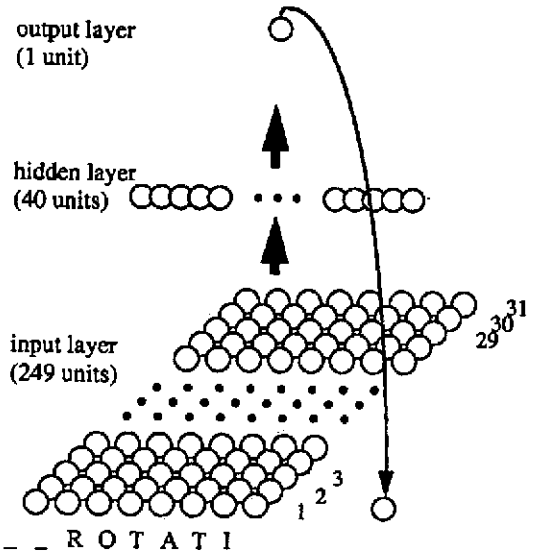
'n Databasis van 1 200 woorde is uit die volgende drie groepe opgebou:

- die 325 woorde wat meeste in Duits gebruik word;

- 850 woorde wat uit artikels in 'n bekende Duitse koerant geneem is; en

- 25 woorde met ongewone letters of letterkombinasies.

Verskillende afrigtingslêers W_n, wat telkens uit n woorde bestaan, is uit hierdie databasis getrek met $n \in \{5,\ 10,\ 20,\ 40,\ 80,\ 160,\ 320,\ 640,\ 1\,000\}$. Kleiner lêers is dus deelversamelings van die groteres. As toetsdata word W_t gebruik wat uit 200 woorde wat nie in $W_{1\,000}$ voorkom nie, bestaan.



Figuur 2.19: Die netwerkargitektuur

Afrigting

Die netwerk is met standaardterugpropagering afgerig om die woorde in die afrigtingsdatabasisse W_n af te kap. Een siklus het uit die aanbieding van al die afrigtingsdata aan die netwerk bestaan. Die woorde is willekeurig ingevoer, maar die verskillende invoer-uitvoerpatrone vir elke woord ($m-3$ vir 'n woord van lengte m) is sekwensieel ingevoer. Dit was nodig om die terugvoerverbinding te kon benut. Tot 10 lopies per databasis W_n is uitgevoer.

In Tabel 2.6 verskyn die konvergensiegedrag van die netwerk. Vir elke woordbasis verskyn die aantal invoer-uitvoerpatrone (N), die totale kwadraatfout na konvergensie (TKF) en die relatiewe fout (RF) as 'n persentasie. Laasgenoemde twee groothede word gedefinieer as

$$TKF = \sum_{i=1}^{N}(\text{verlangde uitvoer vir patroon } i - \text{werklike netto uitvoer})^2$$

en

$$RF = \frac{TKF}{\text{maksimum moontlike fout}} = \frac{TKF}{N}.$$

Die maksimum moontlike fout is gelyk aan die aantal invoer-uitvoerpatrone aangesien die netwerk in die slegste geval elke afkappingspunt verkeerd sal evalueer, met ander woorde 'n 1 in plaas van 'n 0 en anders om. Die fout sal in elke geval 1 of -1 wees met die kwadraatfout altyd 1.

Dit is duidelik dat die netwerk nie afgerig kan word om afkapping perfek te doen as die afrigtingsbasis groter as 20 is nie. In alle gevalle is die oorblywende relatiewe fout onder een persent.

Veralgemening

Die toetsdata W_t is gebruik om die netwerk mee te toets. Dit was duidelik dat daar 'n korrelasie bestaan tussen die grootte van die afrigtingsbasis en die vermoë om nuwe woorde af te kap. Met 'n

W_n met $n =$	5	10	20	40	80	160	320	640	1 000
N	34	60	101	215	468	908	1 917	3 744	5 793
TKF	0	0	0	0,1	1,6	4,7	9	32	39
RF as %	0	0	0	0,046	0,342	0,518	0,470	0,855	0,673

Tabel 2.6: Afrigting met woordbasisse W_n

afrigtingsbasis van 10 woorde kon net 77,8% van die toetspatrone reg afgekap word. Dit verbeter tot 96% met 'n afrigtingsbasis van 1 000 woorde.

Effek van die grootte van die verborge laag

Netwerke met 10, 40 en 80 neurone in die verborge laag is met mekaar vergelyk. Dit het geblyk dat konvergensie beter is as die aantal verborge neurone toeneem. Die veralgemeningsvermoë het ook toegeneem soos die aantal neurone toegeneem het. Die netwerk met 80 verborge neurone kon 96,6% van die afkappings in W_t reg doen in vergelyking met die 96,04% wat die een met 40 verborge neurone kon doen.

Te veel neurone in die verborge laag lei egter tot 'n afname in veralgemeningsvermoë, aangesien die netwerk die afrigtingsdata memoriseer. Daar bestaan dus 'n kritieke aantal verborge neurone.

Gevolgtrekking

Fritzke kom tot die gevolgtrekking dat afkapping met neurale netwerke moontlik blyk te wees. Om resultate te verbeter moet die woordbasis wat vir afrigting gebruik word, groter wees. Afkapping kan vir ander tale gedoen word deur bloot die afrigtingsdatabasis te vervang.

2.9.4 Walter Daelemans en Antal van den Bosch

In die artikel *Veralgemeningsprestasie van terugpropagering op 'n lettergreepverdelingstaak* [7] beskryf Daelemans en Van den Bosch hul navorsing oor hierdie onderwerp. Die proses van lettergreepverdeling in Nederlands word deur teenstrydigheid in fonologiese[1] en morfologiese[2] beperkings bemoeilik. Dit is dus 'n interessante probleem om die veralgemeningsvermoë van neurale netwerke mee te toets.

Die proses van lettergreepverdeling word deur die volgende fonologiese beginsels gelei:

- Die *maksimale-aanvangsbeginsel* wat sê dat tussen twee vokale soveel konsonante as wat saam uitgespreek kan word, na die tweede lettergreep oorgedra word.
- Die *sonoriteitsbeginsel* wat sê dat die segmente in 'n lettergreep na gelang van sonoriteit[3] gerangskik is – van lae sonoriteit met die aanvang van die lettergreep na hoë sonoriteit in die kern en weer lae sonoriteit aan die einde.

[1]Fonologie is die studie van spraakklanke as funksionele elemente in 'n taal.
[2]Morfologie is die studie van die verbuigings- en vervoegingsvorme van 'n taal (vormleer).
[3]Sonoriteit is klankrykheid of welluidendheid.

Hierdie beginsels lei tot lettergreepverdelings soos *groe-nig*, *I-na* en *bad-stof*. Dit word egter soms deur sekere morfologiese beginsels omvergegooi, soos in *groen-achtig* waar *groe-nachtig* verwag sou word, en *stads-tuin* waar *stad-stuin* verwag sou word. As gevolg van woordsamestelling, misluk die normale fonologiese beginsels dikwels (gemiddeld 6% van woordvorme in Nederlandse koerante.)

Die doelwit van die neurale netwerk wat ontwerp word, is om

- ▷ die maksimale-aanvangs- en sonoriteitsbeginsels te onttrek en dit op invoer wat nie in afrigtingsmateriaal voorkom nie, toe te pas;
- ▷ 'n begrip van morfologiese grense en taalspesifieke afkappingskonvensies wat die fonologiese beginsels weerspreek, te onttrek; en
- ▷ om leenwoorde wat die voorafgaande beginsels omverwerp, te herken.

'n Drielaag (invoer, verborge laag en uitvoer) vorentoevoer terugpropageringsnetwerk word gebruik. Die aspekte wat die prestasie van die netwerk beïnvloed, word bespreek.

Enkodering

Die patroonherkenningstaak behels die volgende: besluit vir 'n sekere karakterposisie in 'n woord met 'n linker- en regterkonteks of dit die eerste karakter van 'n nuwe lettergreep is, of nie. Die invoer is dus 'n karakterstring ('n patroon) wat 'n gedeelte van 'n woord verteenwoordig, met een karakterposisie as die beslissingsposisie waarop gefokus word. Die uitvoer is 'n eenvoudige *ja/nee* eenheid wat besluit of die fokusposisie die begin van 'n nuwe lettergreep is, of nie. Die enkodering kan gesien word as 'n „venster" wat oor die woord beweeg word. Beskou byvoorbeeld die enkoderingspatrone vir die woord *ziekenhuis* in Tabel 2.7.

Patroon	Konteks			Konteks		
(Venster)	links		Fokus	regs		Teiken
1	-	-	z	i	e	0
2	-	z	i	e	k	0
3	z	i	e	k	e	0
4	i	e	k	e	n	1
5	e	k	e	n	h	0
6	k	e	n	h	u	0
7	e	n	h	u	i	1
8	n	h	u	i	s	0
9	h	u	i	s	-	0
10	u	i	s	-	-	0

Tabel 2.7: Vensterenkodering vir die woord *ziekenhuis* (hospitaal)

Data

Die afrigtingsdata bestaan uit 19 451 woorde met lettergreepaanduidings en die toetsdata uit 1 945 woorde (wat nie in die afrigtingsdata voorkom nie) uit dieselfde databasis.

Venstergrootte

Afhangende van die venstergrootte wat gekies word, word die volgende waargeneem:

- Die *aantal* soorte afrigtingspatrone wat voorkom, verskil.
- Die gemiddelde verhouding tussen soorte patrone en die aantal keer wat dit in die afrigtingsdata voorkom (T/t), gee 'n aanduiding van die mate waarin die afrigtingsdata die probleemruimte[4] verteenwoordig.
- Verskillende hoeveelhede *dubbelsinnigheid* kom in die afrigtingsdata voor. (Dit is die persentasie van patroonsoorte waarvoor teenstrydige besluite in die afrigtingsdata bepaal kan word.)
- Daar kom *oorvleueling* tussen afrigtings- en toetsdata voor as gevolg van gedeeltelike ooreenkomste tussen woorde. Die woorde *draadje* en *paadje* lewer byvoorbeeld identiese patrone [aadje], [adje-] en [dje--] in die geval van 'n 5-karaktervenster.

Hierdie eienskappe is vir venstergroottes 3, 5 en 7 waargeneem en word in Tabel 2.8 weergegee.

Venstergrootte	Aantal patroonsoorte	T/t	Dubbel-sinnigheid	Oor-vleueling
3 (1-1-1)	6 266	0,03	16,2	97,8
5 (2-1-2)	66 231	0,32	1,0	75,9
7 (3-1-3)	124 309	0,61	0,1	48,7

Tabel 2.8: Vergelyking van eienskappe van verskillende venstergroottes

Uit hierdie resultate blyk die volgende:

- Die afrigtingsdata word minder verteenwoordigend van die probleemruimte soos die venstergrootte toeneem. Dit blyk ook uit die toename in aantal patroonsoorte en die afname in oorvleueling met die toetsdata.
- Vir venstergrootte 7 is dubbelsinnigheid reeds baie laag – so te sê optimaal. Dit blyk dat verdere toename in venstergrootte nie noodwendig tot 'n toename in veralgemeningsvermoë sal lei nie. (Geraas is afwesig by venstergrootte 7.)

Ontleding van uitvoer

Die uitvoer van die neurale netwerk is 'n besluit of 'n koppelteken voor die fokusposisie van die invoerpatroon geplaas moet word – *JA* (aktivering groter of gelyk aan 0,5) of *NEE* (aktivering kleiner as 0,5).

Die netwerkfout op die toetsdata meet die aantal verkeerde besluite oor patrone. Die volgende soorte foute is geïdentifiseer:

1. *Weglating van koppelteken*: Die uitvoer is *NEE* in plaas van *JA* en tel as 'n verkeerde afkapping (koppelteken gemis), byvoorbeeld *pia-no* in plaas van *pi-a-no*.
2. *Plasing van 'n verkeerde koppelteken*: Die uitvoer is *JA* in plaas van *NEE* en tel ook as 'n verkeerde afkapping (koppelteken te veel), byvoorbeeld *pi-a-n-o*.

[4]Die ruimte van alle moontlike patrone, vir hierdie probleem 26^k vir venstergrootte k.

3. *Transposisie*: 'n Koppelteken kom een posisie links of regs van die teiken voor. Dit is 'n kombinasie van bogenoemde twee soorte foute. Sulke foute kom gewoonlik by die koppelposisie van saamgestelde woorde voor waar bykomende morfologiese inligting nodig is om die korrekte afkapping te doen. Twee aangrensende verkeerde netwerkuitvoere, soos in *daa-rom* in plaas van *daar-om*, tel as een verkeerde afkapping.

4. *Afkapping weerskante van 'n konsonant*: Twee aangrensende patrone mag beide 'n *JA*-uitvoer lewer. Indien die koppeltekens weerskante van 'n vokaal voorkom, is dit in Nederlands aanvaarbaar, soos in *pi-a-no*. Indien dit egter weerskante van 'n konsonant voorkom, moet een van die twee koppeltekens verkeerd wees, soos in *daa-r-om*. Dit is dus belangrik om by sulke foute te bepaal of 'n vokaal of konsonant tussen die koppeltekens voorkom.

Deur die hoogste uitvoerwaarde vir die laaste soort fout as die korrekte een te aanvaar, en die ander een as *NEE* te stel, kan die fout reggestel word. Indien die besluit nie die regte een is nie, verander dit na 'n *transposisiefout*. Deur met hierdie regstellingsprosedure te eksperimenteer, is bevind dat dit in omtrent 60 tot 70% van alle gevalle die korrekte oplossing gegee het.

Optimering van prestasie

Na aanvanklike eksperimentering is besluit om die volgende eienskappe as statiese netwerkparameters te beskou:

- *Grootte van verborge laag*: Daar is bevind dat die verborge laag omtrent 1,5 tot 2 keer die aantal invoereenhede moet bevat.

- *Aktiveringswaardes*: Invoer en teikenaktiveringswaardes van 0,9 en 0,1 is gebruik in plaas van 1,0 en 0,0. Dit het tot minder verkeerde afkappings aanleiding gegee.

- *Netwerkparameters*: Vaste waardes is vir leertempo (0,55) en vir momentum (0,5) gebruik.

- *Afrigtingslengte*: Dit het omtrent 300 tot 400 epogge geneem om netwerke tot 'n fout van nagenoeg nul af te rig. As gevolg van oorpassing en oorveralgemening presteer die netwerk dikwels beter op toetsdata na 50 - 100 epogge.

In 'n poging om veralgemeningsvermoë te optimeer, is die effek van sistematiese verandering in die volgende netwerkeienskappe waargeneem:

- *Venstergrootte*: Daar is vasgestel dat die gemiddelde lettergreep uit 4,3 letters bestaan. Om die optimale venstergrootte te bepaal, is linker en regterkonteks apart beskou en die gevolgtrekking is gemaak dat, alhoewel regterkonteks meer bruikbare inligting vir afkapping bevat, dit nie voldoende is vir die taak nie. Venstergrootte 7 (3-1-3) het optimale resultate gelewer.

- *Netwerkargitektuur*: Die regstellingsprosedure vir soort (d) foute is nie deel van die netwerk self nie. Met standaardterugpropagering is dit onmoontlik vir die netwerk om sulke foute te identifiseer, aangesien dit geen geheue het om te onthou dat die vorige patroon reeds 'n *JA*-uitvoer gehad het nie. Simulasie met verskillende terugvoernetwerke het getoon dat terugvoer afkapping nie verbeter nie.

Kombinering van netwerkoplossings

Deur die uitvoere van netwerke wat verskillende resultate lewer, soos een wat byvoorbeeld sekere afkappings uitlaat en een wat neig om te veel afkappings in te plaas, te kombineer, mag tekortkominge gedeeltelik reggestel word. Die volgende twee benaderings is ondersoek:

- ▷ *Modulêre kombinasie*: Die uitvoere van verskeie netwerke vir dieselfde probleem word gekombineer en as invoer aan 'n ander neurale netwerk wat afgerig is om op grond van die invoere te besluit waar die afkapping moet kom, gevoer. Met eksperimentering is getoon dat die beste resultate nie verkry word deur die beste netwerk vir elke enkodering te gebruik nie, maar deur goeie netwerke met netwerke wat wel foute lewer, te kombineer.
- ▷ *Interne kombinasie*: Verskillende enkoderings word in enkele patrone gekombineer en 'n enkele netwerk word afgerig.

Die gevolgtrekking is gemaak dat die kombinasie van verskillende enkoderings op 'n modulêre of interne manier wel tot verbetering in afkappingsprestasie lei. Dit blyk egter dat die hierdie soort netwerke 'n akkuraatheidsdrumpel van 96% het.

Gevolgtrekking

Daelemans en Van den Bosch kom tot die gevolgtrekking dat afkappingsoplossings wat op kennis oor die onderwerp gebaseer is steeds beter as, of vergelykbaar met, neurale-netwerkoplossing is, selfs al word laasgenoemde met taalkundige inligting belaai. Hulle wys egter daarop dat 'n neurale-netwerkbenadering baie effektief gebruik kan word, aangesien die gewigmatriks van 'n afgerigte netwerk, gekombineer met eenvoudige kode vir enkodering, aktivering, dekodering en naprosessering, in 'n eenvoudige en effektiewe afkappingsmodule saamgevat kan word vir teksprosesseerders. In terme van akkuraatheid is dit vergelykbaar met bestaande benaderings, maar sonder dat dit nodig is om groot patroontabelle of 'n woordeboek te stoor.

Hoofstuk 3

Ontwikkeling van neurale netwerk

3.1 Lettergreepverdeling teenoor afkapping

Soos voorheen genoem, is afkapping die proses waardeur woorde aan die einde van 'n gedrukte reël in twee gedeel word sodat die spasiëring nie uitgerek hoef te word om die reël te vul (by dubbelgeskouerde teks) of die regterkant (by linksgeskouerde teks) nie te veel varieer nie. In 'n afkappingsprosedure moet daarvoor voorsiening gemaak word dat 'n woord as gevolg van redigering weer na sy oorspronklike vorm moet terugkeer indien dit nie meer afgekap hoef te word nie. Die afkappingsprosedure behels 'n hele navorsingstaak op sy eie en sal nie in hierdie projek aangepak word nie.

In hierdie navorsingsprojek word op lettergreepverdeling gekonsentreer aangesien afkapping in Afrikaans op grond van lettergreepverdeling gedoen word. Indien 'n neurale netwerk afgerig kan word om woorde vinnig en korrek in lettergrepe te verdeel, kan 'n afkappingsprosedure daarop gebaseer word.

3.2 Idees uit vorige navorsing

Die volgende idees word uit navorsing wat reeds oor die onderwerp gedoen is, oorgeneem:

- Dit lyk asof die proses om 'n woord deur 'n „venster" te beskou, die aangewese manier is om afrigtingspatrone vir die neurale netwerk te genereer. Hierdie konsep word gebruik maar met die volgende aanpassings:
  - In Afrikaans kom lettergrepe van vier letters algemeen voor. Ons gebruik dus 'n venster wat die konteks tot vier letters weerskante van die afkappingsposisie bevat.
  - 'n Lettergreep in Afrikaans kan uit 'n enkele letter bestaan. Ons beskou dus alle moontlike afkappingsposisies in elke woord.
- Dit is duidelik dat enkellaagnetwerke nie vir die afkappingsprobleem geskik is nie. Netwerke met ten minste twee lae (ten minste een verborge laag) word dus oorweeg.
- 'n Koderingsmetode waar vokale en konsonante van mekaar geskei word, word gebruik.
- 'n Enkele uitvoerneuron wat aandui *ja* kap af of *nee* moenie afkap nie, word gebruik.

3.3 Verkryging van data

Ten einde 'n neurale netwerk af te rig om die afkapping van Afrikaanse woorde outomaties te doen, is 'n groot versameling Afrikaanse woorde met korrekte lettergreepverdelings nodig. So 'n versameling woorde is uit ELHAT, die elektroniese weergawe van die *Verklarende Handwoordeboek van die Afrikaanse Taal* onttrek. Besonderhede oor die proses wat gevolg is, verskyn in Bylae A.

'n Deelversameling van hierdie woorde is aangewend om die neurale netwerk aanvanklik af te rig. Hierna is ander deelversamelings gebruik om die werking van die neurale netwerk te toets en regstellings te maak indien nodig.

3.4 Voorbereiding van invoer

Data wat aan MATLAB se neurale-netwerkpakket verskaf word, moet in 'n spesifieke formaat wees. Vir die probleem van lettergreepverdeling moet die data na matrikse van nulle en ene omgeskakel word. Hierdie matrikse moet aandui watter letters gebruik word, hoe die letters op mekaar volg en waar die lettergreepverdelings is.

Uiteindelik, na afrigting, word woorde sonder koppeltekens aan die neurale netwerk verskaf wat dan deur die neurale netwerk in lettergrepe verdeel moet word. Die afrigtingsdata moet dus deur twee afsonderlike matrikse verteenwoordig word, naamlik

- 'n matriks (sonder lettergreepaanduidings) wat die volgorde van letters duidelik weergee; en
- 'n vektor wat aandui presies waar lettergreepverdelings moet voorkom.

3.4.1 'n Verkorte alfabet

Ten einde die proses van lettergreepverdeling te illustreer, word 'n verkorte alfabet van 7 letters, naamlik e, d, g, l, r, t en i gebruik. Om woorde wat uit hierdie letters bestaan, te kodeer, word syfers tussen 2 en 8 aan die letters toegeken.

In die datalêer word lettergreepverdelings deur 'n koppelteken (-) aangedui en koppeltekens wat in woorde verskyn, deur 'n gelykaanteken (=). Beide hierdie tekens dui die posisie van lettergreepverdelings aan en ons ken die syfer 1 daaraan toe. (Aangesien ons nie afkapping doen nie, onderskei ons nie op hierdie stadium tussen die twee nie. Met afkapping sal aangedui moet word dat die gelykaanteken weer in die woord teruggeplaas moet word indien afkapping nie daar plaasvind nie.)

Die karakters wat moontlik in 'n woord kan voorkom, verskyn in matriks K, terwyl die ooreenstemmende syfers in matriks D verskyn.

$$K = \begin{bmatrix} - & e & d \\ g & l & r \\ t & i & = \end{bmatrix} \qquad D = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix}$$

In die datalêer `OefWrd.dat` verskyn die volgende tien woorde wat letters uit die verkorte alfabet bevat:

```
ge-let-terd   leer-gie-rig   te-ler       tel-ler     rie-te
lig-ter       let-ter        ge-leer-de   gel-di-ge   le-dig
```

MATLAB se *Import Wizard* word gebruik om hierdie datalêer in te voer. Die proses om woorde as syfers te kodeer, word soos volg in MATLAB geprogrammeer:

```
K=['-' 'e' 'd'; 'g' 'l' 'r'; 't' 'i' '='];        %karakters
D=[1 2 3; 4 5 6; 7 8 1];                          %syfers

N=0;
for n=1:1                    %Lees eerste woord uit databasis.
S=char(OefWrd(n));           %Woord uit databasis.
wl=length(S);                %Woordlengte (met koppeltekens)
T1=zeros(wl,1);              %Teiken
W1=zeros(wl,1);              %Woord (met koppeltekens)
for k = 1:wl
    if (S(k) == '-'|S(k) == '=')   %Skep teikenvektor met
       T1(k) = 1;                  %lettergreepaanduidings.
    end
    for i=1:3
        for j=1:3
            if S(k) == K(i,j)
               W1(k) = D(i,j);
            end
        end
    end
end
```

Slegs die eerste woord, naamlik *ge-let-terd*, word ingelees. Dit word geënkodeer sodat dit deur die vektor $[4\ 2\ 1\ 5\ 2\ 7\ 1\ 7\ 2\ 6\ 3]'$ voorgestel word.

Die geënkodeerde woord sonder koppeltekenaanduidings en 'n teikenvektor wat aandui waar die koppeltekens moet voorkom, word egter benodig. Die koppeltekens (ene) word dus uit vektor W1 verwyder, terwyl vektor TT geskep word waarin 'n 1 in 'n sekere posisie aandui dat 'n koppelteken direk na die letter in daardie posisie van vektor W1 moet verskyn en 'n 0 dat daar geen koppelteken na die letter moet wees nie. Hierdie proses word soos volg in MATLAB geprogrammeer:

```
W=W1;
TT=T1;
t=0;
for k=1:wl
    if W1(k) == 1
       W(k-t) = [];          %Woord sonder koppeltekens.
       TT(k-t-1) = [];       %Teiken: koppelteken na element wat 1 bevat.
       t=t+1;
    end
end
WL=wl-t;                     %Werklike woordlengte.
```

Die woord *geletterd* word nou deur die vektor $W = [4\ 2\ 5\ 2\ 7\ 7\ 2\ 6\ 3]'$ verteenwoordig en die teiken $TT = [0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$ dui aan dat koppeltekens na die eerste *e* en na die eerste *t* moet kom.

Die gekodeerde woord word nou na 'n skikking van nulle en ene, waarin elke letter van die alfabet deur 'n sekere kolom verteenwoordig word, omgeskakel. Die eerste ry van die skikking word vir die spasiekarakter ⊔ gereserveer. (Die gebruik hiervan word later duidelik.) Die woord *geletterd* word deur die skikking in Tabel 3.1 verteenwoordig.

		g	e	l	e	t	t	e	r	d
1	⊔	0	0	0	0	0	0	0	0	0
2	e	0	1	0	1	0	0	1	0	0
3	d	0	0	0	0	0	0	0	0	1
4	g	1	0	0	0	0	0	0	0	0
5	l	0	0	1	0	0	0	0	0	0
6	r	0	0	0	0	0	0	0	1	0
7	t	0	0	0	0	1	1	0	0	0
8	i	0	0	0	0	0	0	0	0	0
9	b	0	0	0	0	0	0	0	0	0

Tabel 3.1: Die woord *geletterd* in skikking

Die doelwit van die neurale netwerk is om vir 'n gegewe woord aan te dui dat afkapping op 'n sekere plek in die woord mag plaasvind, of nie. Die teiken moet dus vir opeenvolgende posisies in die woord aandui *kap af* (aangedui met 'n 1) of *moenie afkap nie* (aangedui met 'n 0).

Om hierdie opeenvolgende posisies van 'n woord te identifiseer, skuif ons 'n *venster* van agt karakters oor die woord. Indien afkapping na posisie 4 van die venster moet plaasvind, is die teiken 1 – indien nie, is die teiken 0.

In Afrikaans kan 'n lettergreep aan die begin van 'n woord uit slegs een letter bestaan, soos in *a-simp-to-ties*. 'n Lettergreep aan die einde van 'n woord kan ook uit slegs een letter bestaan, soos in *ge-heu-e*. In ons venster mag daar dus drie spasies voor die eerste letter van die woord en drie spasies na die laaste letter van die woord voorkom.

Die opeenvolgende vensters vir die woord *geletterd*, met die ooreenstemmende teikens, verskyn in Figuur 3.1.

Venster								Teiken
			g	e	l	e	t	0
		g	e	l	e	t	t	1
	g	e	l	e	t	t	e	0
g	e	l	e	t	t	e	r	0
e	l	e	t	t	e	r	d	1
l	e	t	t	e	r	d		0
e	t	t	e	r	d			0
t	t	e	r	d				0

Figuur 3.1: Opeenvolgende vensters (met teikens) vir die woord *geletterd*

Die toegelate spasies in die vensters word deur kolom 1 in die skikking verteenwoordig. Die eerste venster word dus deur Figuur 3.2 voorgestel.

		␣	␣	␣	g	e	l	e	t
1	␣	1	1	1	0	0	0	0	0
2	e	0	0	0	0	1	0	1	0
3	d	0	0	0	0	0	0	0	0
4	g	0	0	0	1	0	0	0	0
5	l	0	0	0	0	0	1	0	0
6	r	0	0	0	0	0	0	0	0
7	t	0	0	0	0	0	0	0	1
8	i	0	0	0	0	0	0	0	0
9	b	0	0	0	0	0	0	0	0

Figuur 3.2: Eerste venster in skikking

Die proses om opeenvolgende vensters van 'n woord op hierdie wyse in skikkings te plaas, word soos volg in MATLAB geprogrammeer:

```
PP=zeros(8,8);
SW=[1; 1; 1; W; 1; 1; 1];          %String waaruit venster kom.
for z=1:WL-1                       %Woord van lengte WL skuif WL-1 keer.
    WW=SW(z:z+7);                  %Venster bestaan uit 8 letters.
    Venster = WW';
    for i=1:8                      %Skikking met kolom vir elke letter:
        PP(i,WW(i),z)=1;           %1-spasie; 2-e; 3-g; ens.
    end
    N=N+1;
    INVOER(:,:,N) = PP(:,:,z)';
    T(N) = TT(z);
end
```

Die opeenvolgende vensters word deur die skikking `INVOER(; , ; , N)` voorgestel en die teikens verskyn in vektor T. Die eerste twee vensters van die woord *geletterd* word deur die volgende twee matrikse voorgestel:

```
INVOER(:,:,1) = 1     1     1     0     0     0     0     0
                0     0     0     0     1     0     1     0
                0     0     0     0     0     0     0     0
                0     0     0     1     0     0     0     0
                0     0     0     0     0     1     0     0
                0     0     0     0     0     0     0     0
                0     0     0     0     0     0     0     1
                0     0     0     0     0     0     0     0

INVOER(:,:,2) = 1     1     0     0     0     0     0     0
                0     0     0     1     0     1     0     0
                0     0     0     0     0     0     0     0
                0     0     1     0     0     0     0     0
                0     0     0     0     1     0     0     0
                0     0     0     0     0     0     0     0
                0     0     0     0     0     0     1     1
                0     0     0     0     0     0     0     0
```

Om lotafrigting (wat meer effektief as sekwensiële afrigting is) te gebruik, moet die invoer vir 'n neurale netwerk in 'n enkele matriks gegee word. (Let wel, afhangende van die probleem onder beskouing lewer sekwensiële afrigting dikwels meer akkurate resultate.) In die invoermatriks moet elke kolom 'n spesifieke venster voorstel terwyl die teikenvektor die ooreenstemmende teiken vir elke venster gee. In MATLAB word elke INVOER-skikking deur die volgende opdrag na 'n kolomvektor omgeskakel:

```
for i=1:N
    Q=INVOER(:,:,i)';
    P(:,i)=Q(:);
end
```

Die invoer vir die woord *geletterd* is 'n 64×8 matriks P waarin elke kolom 'n venster verteenwoordig waarvan die teiken deur die ooreenstemmende kolom in die 1×8 vektor T gegee word. Matriks P en vektor T verskyn in Figuur 3.3.

3.4.2 Die volledige alfabet

Die verkorte alfabet is net gebruik om die illustrasie van die hele proses te vergemaklik. Die volgende karakters is egter nodig om alle moontlike Afrikaanse woorde in te sluit:

- die volledige alfabet van 26 letters;
- letters met kappies, deeltekens en aksente, naamlik

  ä, ë, ê, é, è, ï, ö, ô, ü en û;

- die afkappingsteken (') wat in woorde soos *pa'tjie*, *wag-'n-bietjie*, *foto's*, ens. voorkom; (Hierdie karakter is deur MATLAB gereserveer en mag dus nie in woorde voorkom nie. Dit word met 'n ! in die datalêers vervang.)
- lettergreepaanduidings (-) en koppeltekenaanduidings (=).

Daar is dus altesaam 39 moontlike karakters.

'n Syferwaarde word aan elkeen van die moontlike karakters toegeken. Daar is besluit om die vokale en konsonante apart te groepeer aangesien hulle verskillende fonologiese funksies vervul en omdat resultate uit vorige navorsing [25] toon dat neurale netwerke beter presteer indien 'n duidelike onderskeid tussen die twee groepe gemaak word.

Aangesien ons op hierdie stadium in lettergreepverdeling belangstel, word lettergreepaanduidings en koppeltekenaanduidings deur dieselfde syfer aangedui om programmering te vereenvoudig.

- Aan beide die lettergreepaanduiding (-) en die koppeltekenaanduiding (=) word die syfer 1 toegeken. Die = word in die laaste posisie van die skikking (posisie 40) geplaas sodat daar 'n duidelike onderskeid tussen die twee soorte koppeltekens is;
- syfers 2 tot 16 word aan vokale toegeken (deeltekens, kappies, ens. ingesluit);
- die syfer 17 verteenwoordig die letter y ('n konsonant wat soos 'n vokaal gebruik word);

```
P =  1  1  1  0  0  0  0  0
     0  0  0  0  1  0  1  0
     0  0  0  0  0  0  0  0
     0  0  0  1  0  0  0  0
     0  0  0  0  0  1  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  0  1
     0  0  0  0  0  0  0  0
     1  1  0  0  0  0  0  0
     0  0  0  1  0  1  0  0
     0  0  0  0  0  0  0  0
     0  0  1  0  0  0  0  0
     0  0  0  0  1  0  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  1  1
     0  0  0  0  0  0  0  0
     1  0  0  0  0  0  0  0
     0  0  1  0  1  0  0  1
     0  0  0  0  0  0  0  0
     0  1  0  0  0  0  0  0
     0  0  0  1  0  0  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  1  1
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  0  0
     0  1  0  1  0  0  1  0
     0  0  0  0  0  0  0  0
     1  0  0  0  0  0  0  0
     0  0  1  0  0  0  0  0
     0  0  0  0  0  0  0  1
     0  0  0  0  1  1  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  0  0
     1  0  1  0  0  1  0  0
     0  0  0  0  0  0  0  1
     0  0  0  0  0  0  0  0
     0  1  0  0  0  0  0  0
     0  0  0  0  0  0  1  0
     0  0  0  1  1  0  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  0  1
     0  1  0  0  1  0  0  0
     0  0  0  0  0  0  1  0
     0  0  0  0  0  0  0  0
     1  0  0  0  0  0  0  0
     0  0  0  0  0  1  0  0
     0  0  1  1  0  0  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  1  1
     1  0  0  1  0  0  0  0
     0  0  0  0  0  1  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  1  0  0  0
     0  1  1  0  0  0  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  1  1  1
     0  0  1  0  0  0  0  0
     0  0  0  0  1  0  0  0
     0  0  0  0  0  0  0  0
     0  0  0  0  0  0  0  0
     0  0  0  1  0  0  0  0
     1  1  0  0  0  0  0  0
     0  0  0  0  0  0  0  0

T =  0  1  0  0  1  0  0  0
```

Figuur 3.3: Invoermatriks en teikenvektor vir die woord *geletterd*

- ▷ syfers 18 tot 37 verteenwoordig konsonante;
- ▷ die syfer 38 verteenwoordig die afkappingsteken (!);
- ▷ die syfer 39 word deur 'n nie-woordkarakter # ingeneem om die skikking vol te maak. (Hierdie karakter sal nooit gebruik word nie.)

Opmerking: 'n Ondersoek na die frekwensie waarmee letters saam in Afrikaans voorkom, mag moontlik daartoe lei dat die volgorde van letters nie alfabeties moet wees vir optimale afkapping nie.

Die toekenning van syfers verskyn in Tabel 3.2.

-	a	ä	e	ë	ê	é	è	i	ï
1	2	3	4	5	6	7	8	9	10
o	ö	ô	u	ü	û	y	b	c	d
11	12	13	14	15	16	17	18	19	20
f	g	h	j	k	l	m	n	p	q
21	22	23	24	25	26	27	28	29	30
r	s	t	v	w	x	z	!	#	=
31	32	33	34	35	36	37	38	39	1

Tabel 3.2: Die volledige alfabet met syfertoekennings

Die MATLAB-program `VolAlf.m` wat Afrikaanse woorde na die geskikte vorm vir die neurale netwerk omskakel, lyk soos volg:

```
K=['-' 'a' '\"{a}' 'e' '\"{e}' '\^{e}' '\'{e}' '\`{e}' 'i' '\"{\i}';
   'o' '\"{o}' '\^{o}' 'u' '\"{u}' '\^{u}' 'y' 'b' 'c' 'd';
   'f' 'g' 'h' 'j' 'k' 'l' 'm' 'n' 'p' 'q';
   'r' 's' 't' 'v' 'w' 'x' 'z' '!' '#' '='];  %! verteenwoordig '

D=[ 1  2  3  4  5  6  7  8  9 10;        %syfers
   11 12 13 14 15 16 17 18 19 20;
   21 22 23 24 25 26 27 28 29 30;
   31 32 33 34 35 36 37 38 39  1];       %= ook afkapping

N=0;                                     %Teller vir vensters.
for n=1:1238                             %Lees woorde een vir een
    S=lower(char(WRD1(n)));              %uit databasis.
    wl=length(S);                        %Woordlengte (met koppeltekens)
    T1=zeros(wl,1);                      %Teiken (met koppeltekens)
    W1=zeros(wl,1);                      %Woord (met koppeltekens)
    for k = 1:wl
        if (S(k) == '-'|S(k) == '=')
           T1(k) = 1;
        end
        for i=1:4
            for j=1:10
                if S(k) == K(i,j)
```

```
                W1(k) = D(i,j);
            end
        end
    end
end
W=W1;
TT=T1;
t=0;
for k=1:wl                          %Haal koppeltekens uit.
    if W1(k) == 1
       W(k-t) = [];                 %Woord sonder koppeltekens.
       TT(k-t-1) = [];              %Teiken: koppelteken na element
       t=t+1;                       % wat 1 bevat.
    end
end
WL=wl-t;                            %Werklike woordlengte.

PP=zeros(8,40);                     %Skikking waarin invoer geplaas word.
SW=[1; 1; 1; W; 1; 1; 1];           %String waaruit venster kom.
for z=1:WL-1                        %Woord van lengte WL skuif WL-1 keer.
    WW=SW(z:z+7);                   %Venster bestaan uit 8 letters.
    Venster = WW';
    for i=1:8                       %Skikking met kolom vir elke letter:
        PP(i,WW(i),z)=1;            %1-koppelteken; 2-a; 3-\"{a}; ens.
    end
    N=N+1;
    Q = PP(:,:,z)';
    P(:,N)=Q(:);                    %Kolom P(:,N) is matriks PP(:,:,N).
    T(N) = TT(z);
  end
end
save WRD1tr P T
```

Die datalêer `WRD1.dat` bestaan uit 1 238 woorde uit `AFKAP.dat` en word deur middel van MATLAB se *Import Wizard* ingevoer. Die dimensies van die matriks P wat deur die program `VolAlf.m` uit die 1 238 woorde in `WRD1.dat` gegenereer word, is $320 \times 10\,088$.

- ▷ Die 320 rye verteenwoordig die 40 karakters maal die 8 kolomme van die venster.
- ▷ Die 10 088 kolomme verteenwoordig al die moontlike vensters wat uit hierdie 1 238 woorde gevorm word.

T is 'n $1 \times 10\,088$ matriks wat die teiken van elke ooreenstemmende venster verteenwoordig.

Invoermatriks P en teikenvektor T, word in die lêer `WRD1tr.mat` gestoor.

3.5 Die neurale netwerk

3.5.1 Afrigting

MATLAB se *Neural Network Toolbox* word gebruik om 'n netwerk af te rig om lettergreepverdeling te doen. Ons begin deur die invoer-en-teikenpare in `WRD1tr.mat` vir afrigting te gebruik.

Die MATLAB-instruksies om 'n basiese neurale netwerk met een verborge laag af te rig, verskyn in `NNBasies.m` wat hier volg:

```
load('WRD1tr');
S1 = 20;                              %Aantal neurone in verborge laag.
[R,Q] = size(P);                      %Dimensies van P.
[S2,Q] = size(T);                     %Dimensies van T.

net = newff(minmax(P),[S1 S2],{'tansig' 'purelin'},'trainrp');

net.trainParam.goal = 0.015;          %Doelwit: MSE=0,015.
net.trainParam.show = 20;
net.trainParam.epochs = 200;
[net,tr] = train(net,P,T);
```

Die opdragte in hierdie program doen die volgende:

- Die invoermatriks (P) en teikenvektor (T) wat uit `WRD1.dat` gegenereer is, word gebruik. Ons laai dus die lêer `WRD1tr.mat` deur die opdrag `load WRD1tr`.
- Laag 1 ('n verborge laag) se aantal neurone word as 20 gekies, dus `S1=20`.
- Die aantal invoerelemente (R) word deur die aantal rye in die invoermatriks P gegee en die aantal afrigtingspare (Q) deur die aantal kolomme. Die dimensies van P kan deur middel van die volgende opdragte in MATLAB verkry word:

```
>> load WRD1tr;
>> size(P)

ans =

         320        10088
```

- Laag 2 is die uitvoerlaag wat uit 'n enkele neuron bestaan – `size(T)` gee die dimensies van T, naamlik $1 \times 10\,088$.
- Uit vorige navorsing en uit MATLAB se voorbeelde is dit duidelik dat terugpropagering die aangewese model is om vir patroonherkenningsprobleme van hierdie aard te gebruik.

  Die opdrag

```
net = newff(minmax(P),[S1 S2],'tansig' 'purelin','trainrp');
```

skep 'n nuwe vorentoe-voer (*feed forward*) terugpropageringsnetwerk (newff) met die volgende eienskappe:

- Die grense waarbinne die invoerelemente voorkom word deur `minmax(P)` bepaal – hier is dit [0; 1];
- Die aantal neurone in die lae van die netwerk word deur `[S1 S2]` gegee;
- Die oordragfunksies wat gebruik word, is `tansig` vir laag 1 en `purelin` vir laag 2 (Hierdie oordragfunksies is gebruik na aanleiding van die patroonherkenningsprobleme wat in MATLAB se handleiding beskryf is. Dit sal later heroorweeg word.);
- Die veerkragtige (*resilient*) terugpropageringsalgoritme (RP) word vir afrigting gebruik, dus `trainrp`.

▷ Die volgende parameters word deur die gebruiker bepaal:

- `net.trainParam.goal = 0.015;` stel die doelwit van die neurale netwerk op 0,015. Dit beteken dat die netwerk moet voortgaan om af te rig totdat die gemiddelde kwadraatfout (*MSE*) kleiner as 0,015 is.
- `net.trainParam.show = 20;` bepaal dat elke 20ste epog se resultate in MATLAB se opdragvenster en op die grafiek wat van die afrigtingsproses getrek word, getoon moet word.
- `net.trainParam.epochs = 200;` bepaal dat hoogstens 200 epogge uitgevoer moet word.

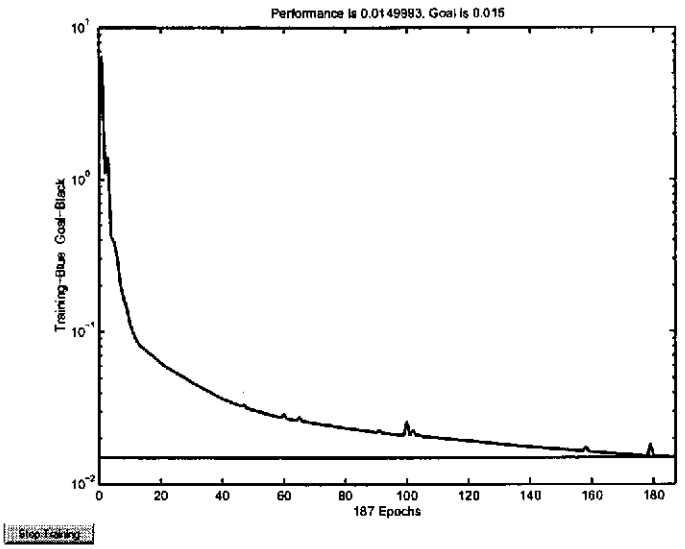
▷ Die opdrag `[net,tr] = train(net,P,T);` rig die netwerk af met P en T as invoer en teiken.

Om die afgerigte netwerk te stoor, word die opdrag `save netRPtp20 net` in MATLAB se opdragvenster ingevoer. Die afgerigte netwerk word as `netRPtp20.mat` gestoor. (Die naam van die netwerk is afgelei van die afrigtingsalgoritme `trainrp`, dus netRP, plus tp vir *tansig* en *purelin*, plus die aantal neurone in die verborge laag.)

Deur die opdrag `NNBasies` in MATLAB in te voer, word die netwerk afgerig. Die inligting van elke 20ste epog verskyn in MATLAB:

```
TRAINRP, Epoch 0/600, MSE 1.60071/0.015, Gradient 7.33076/1e-006
TRAINRP, Epoch 20/600, MSE 0.0631671/0.015, Gradient 0.0225839/1e-006
TRAINRP, Epoch 40/600, MSE 0.0367173/0.015, Gradient 0.02378/1e-006
TRAINRP, Epoch 60/600, MSE 0.0289489/0.015, Gradient 0.0633754/1e-006
TRAINRP, Epoch 80/600, MSE 0.0235534/0.015, Gradient 0.0212425/1e-006
TRAINRP, Epoch 100/600, MSE 0.025909/0.015, Gradient 0.0766952/1e-006
TRAINRP, Epoch 120/600, MSE 0.0193987/0.015, Gradient 0.0111546/1e-006
TRAINRP, Epoch 140/600, MSE 0.0176904/0.015, Gradient 0.00964981/1e-006
TRAINRP, Epoch 160/600, MSE 0.0164073/0.015, Gradient 0.00822961/1e-006
TRAINRP, Epoch 180/600, MSE 0.0155535/0.015, Gradient 0.0246642/1e-006
TRAINRP, Epoch 187/600, MSE 0.0149993/0.015, Gradient 0.00540412/1e-006
TRAINRP, Performance goal met.
```

Die neurale netwerk het dus 187 epogge nodig gehad om 'n gemiddelde kwadraatfout van minder as 0,015 te bereik. Die grafiek van die afrigtingsproses verskyn in Figuur 3.4.



Figuur 3.4: Basiese afrigting met WRD1tr

3.5.2 Evaluering van die neurale netwerk se prestasie

Noudat ons die netwerk afgerig het, wil ons bepaal hoe goed dit 'n datastel wat ander woorde as dié in `WRD1.dat` bevat, in lettergrepe sal verdeel. Ons gebruik die datastel `DAT1.dat` hiervoor en verwerk dit na 'n invoermatriks (P) en teikenvektor (T) in `DAT1tr.mat`. Deur die neurale netwerk met P te simuleer en die uitvoer van die netwerk met die ingevoerde teiken T te vergelyk, kan vasgestel word hoe effektief die netwerk is.

Om die effektiwiteit te bepaal, word elke patroon wat korrek geïdentifiseer word, as 'n sukses beskou. Elke posisie in die woord word as 'n potensiële afkappingspunt beskou. Indien die neurale netwerk 'n posisie as *kap af* of as *moenie afkap nie* reg klassifiseer, word dit as 'n sukses beskou. Die totale aantal patrone wat deur die program `VolAlf` gegenereer word, is potensiële suksesse (of mislukkings).

Die volgende program, `NNSTAT.m`, bepaal hoe effektief die netwerk is om die woorde in `DAT1.dat` in lettergrepe te verdeel:

```
load DAT1tr;              %Laai datastel.
[R,N] = size(P);          %Dimensies van invoermatriks.
load netRPtr20;           %Laai neurale netwerk.
a=sim(net,P);             %Simuleer met P. (Kry uitvoer van netwerk.)
t=round(a);               %Uitvoer (nulle en ene).
e=T-t;                    %Foute in simulasie.
tel=0;                    %Tel aantal verkeerd.
for i=1:N;
    if e(i)==1;
        tel=tel+1;
    end;
end;
STAT=(1-tel/N)*100;
Persentasie_korrek = STAT
```

Deur die opdrag NNSTAT in MATLAB in te voer, kry ons die volgende resultaat:

```
>> NNSTAT

Persentasie_korrek =

   95.2453
```

Die netwerk kan dus 95,25% van die patrone in DAT1tr.mat reg klassifiseer. Die resultate vir die vyf datalêers DAT1.dat ... DAT5.dat, elk met 1 000 woorde, en vir die afrigtingsdata in WRD1.dat verskyn in Tabel 3.3.

DAT1tr	DAT2tr	DAT3tr	DAT4tr	DAT5tr	Gemiddelde	WRD1tr
95,25	95,15	95,14	95,58	95,63	95,35	98,81

Tabel 3.3: Prestasieresultate van netRPtp20

Die netwerk het dus gemiddeld 95,35% van die patrone in die vyf lêers met onbekende data en 98,81% van die afrigtingspatrone reg geklassifiseer.

3.5.3 Bepaal die beste neurale netwerk vir die probleem

Om die beste neurale netwerk vir die afkappingsprobleem te bepaal, word die volgende netwerkeienskappe beskou:

- Variërende aantal neurone in die verborge laag;
- Verskillende oordragfunksies;
- Meer as een verborge laag;
- Verskillende afrigtingsalgoritmes;
- Vroeë beëindiging;
- Meer en beter afrigtingsdata;
- Kort woorde bygevoeg.

Variërende aantal neurone in verborge laag

Ons gebruik dieselfde neurale-netwerkkonfigurasie as voorheen (trainrp met *tan-sigmoid* en *purelin* oordragfunksies) en rig weer met WRD1tr af. Deur die aantal neurone in die verborge laag (S1) te vermeerder, word die resultate in Tabel 3.4 verkry.

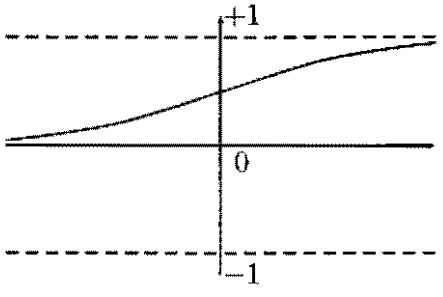
Uit hierdie resultate is dit duidelik dat die veralgemeningsvermoë van die neurale netwerk afneem soos die aantal neurone in die verborge laag toeneem. Terselfdertyd neem die prestasie met die afrigtingsdata toe. Dit dui daarop dat die netwerk die afrigtingsdata al hoe beter memoriseer soos die neurone in die verborge laag toeneem.

Aantal neurone	Aantal epogge	Gemiddelde prestasie met nuwe data	Prestasie met afrigtingsdata
20	187	95,35	98,81
30	183	95,18	99,18
40	167	94,43	99,40
70	193	91,72	99,55

Tabel 3.4: Prestasie met verskillende aantal neurone in verborge laag

Verskillende oordragfunksies

Die *log-sigmoidale* oordragfunksie forseer die uitvoer om in die interval (0, 1) te val en verskyn in Figuur 3.5.



Figuur 3.5: Die `log-sigmoidale` funksie

Aangesien die invoer en uitvoer vir die afkappingsprobleem uit nulle en ene bestaan, maak dit sin om die *log-sigmoidale* oordragfunksie in beide neuronlae te gebruik. Die opdrag in `NNBasies` verander na

```
net = newff(minmax(P),[S1 S2],'logsig' 'logsig','trainrp');
```

Deur hierdie netwerk weereens met veerkragtige terugpropageringsalgoritme (RP) vir verskillende aantal neurone in die verborge laag af te rig, kry ons die resultate wat in Tabel 3.5 verskyn. Die netwerke is telkens as `netRP`, plus `11` vir `logsig logsig`, plus die aantal neurone in die verborge laag gestoor. Die netwerk met 20 neurone in die verborge laag is byvoorbeeld as `netRP1120` gestoor.

Die volgende kan uit hierdie resultate afgelei word:

- Veel minder epogge is nodig om die netwerk af te rig.
- Die veralgemeningsvermoë van die netwerke neem toe soos die aantal neurone in laag 1 toeneem. By 80 neurone is die prestasie 'n maksimum, waarna dit weer begin afneem. Dit blyk dus dat daar wel 'n optimale aantal neurone vir die verborge laag is – dit sal met meer data verder ondersoek word.
- Die prestasie met die afrigtingsdata verbeter in verhouding tot die prestasie met nuwe data. Afrigtingsdata word dus nie gememoriseer nie.

Aantal neurone	Aantal epogge	Gemiddelde prestasie met nuwe data	Prestasie met afrigtingsdata
20	45	94,95	98,72
30	43	94,93	98,60
40	33	95,00	98,70
50	29	95,02	98,49
60	31	95,15	98,72
70	33	95,18	98,67
80	35	95,26	98,69
100	36	95,12	98,60

Tabel 3.5: Prestasie met `log-sigmoidale` oordragfunksie in beide lae

Alhoewel die netwerk met *tansig*- en *purelin*-oordragfunksies effens beter resultate as die *log-sigmoidale* oordragfunksie in beide lae gelewer het (95,35% teenoor 95,26%), sal ons voortaan van laasgenoemde netwerke gebruik maak, aangesien dit baie vinniger konvergeer en nie die data memoriseer nie.

Meer as een verborge laag

Vervolgens word 'n tweede verborge laag in die netwerk ingevoer en die aantal neurone in beide lae word gevarieer. Die volgende MATLAB-program, `NNBAS2.m`, is gebruik:

```
load('WRD1tr');
S1 = 40;
S2 = 10;
[R,Q] = size(P);
[S3,Q] = size(T);
net = newff(minmax(P),[S1 S2 S3],{'logsig' 'logsig' 'logsig'},'trainrp');
net.trainParam.goal = 0.015;
net.trainParam.show = 20;
net.trainParam.epochs = 200;
[net,tr] = train(net,P,T);
```

Let op die volgende in hierdie program:

- ▷ WRD1tr word weereens vir afrigting gebruik;
- ▷ S1 en S2 se aantal neurone word deur die gebruiker gekies;
- ▷ S3 se aantal neurone word deur die aantal rye in T bepaal (in hierdie geval een);
- ▷ die *logsig*-oordragfunksie word in al drie lae gebruik. Dit is so gekies om vergelyking met vorige waarnemings moontlik te maak. Die gebruik van ander funksies mag later oorweeg word.

Die netwerke is onder die naam net plus die aantal neurone in elke laag gestoor. Die netwerk met 20 neurone in die eerste en 10 neurone in die tweede laag word dus deur net2010 geïdentifiseer. Die resultate verskyn in Tabel 3.6.

		Prestasie: data			Prestasie: data			Prestasie: data	
S1	S2	nuut	afrig	S2	nuut	afrig	S2	nuut	afrig
20	10	94,39	98,57	–	–	–	–	–	–
30	10	94,99	98,62	15	94,68	98,66	20	95,21	98,62
40	10	95,08	98,72	15	94,90	98,67	20	94,90	98,74
50	10	95,31	98,62	15	95,17	98,70	20	95,10	98,63
60	10	95,14	98,60	15	95,16	98,63	–	–	–

Tabel 3.6: Prestasie met 2 verborge lae

Die aantal epogge wat nodig was om hierdie netwerke af te rig, het tussen 28 en 39 gewissel. Dit is vergelykbaar met (of selfs gouer as) die afrigting met net een verborge laag.

Uit hierdie resultate is dit duidelik dat 'n bykomende laag wel nuttig kan wees. Die 50-10-1 netwerk het die beste veralgemeningsprestasie van 95,31% gelewer. Dit is beter as die prestasie met 'n enkele verborge laag met 80 neurone, en behoort verder ondersoek te word.

Verskillende afrigtingsalgoritmes

Die skrywers van MATLAB se neurale-netwerkpakket het die verskillende afrigtingsmetodes vir verskillende toepassings met mekaar vergelyk. Vir patroonherkenningsprobleme het Veerkragtige-terugpropagering (RP), Geskaalde verwante-hellingmetode (SCG), Powell-Beale Herstelling (CGB) en Polak-Ribiére Opdatering (CGP) telkens die beste resultate ten opsigte van spoed en die gebruik van geheue gelewer. Slegs hierdie afrigtingsmetodes word vir die afkappingsprobleem beskou. Die resultate verskyn in Tabel 3.7.

		RP			SCG			CGB			CGP	
		Data			Data			Data			Data	
S1	Epog	Nuut	Afrig	Epog	Nuut	Afrig	Epog	Nuut	Afrig	Epog	Nuut	Afrig
20	45	94,95	98,72	78	94,74	98,52	51	94,64	98,62	87	94,04	98,54
40	33	95,00	98,70	93	94,78	98,59	48	94,69	98,60	84	94,76	98,58
60	31	95,15	98,72	83	94,74	98,58	55	94,78	98,58	162	94,33	98,09
80	35	95,26	98,69	–	–	–	61	94,24	98,55	–	–	–
100	36	95,12	98,60	–	–	–	–	–	–	–	–	–

Tabel 3.7: Prestasie van verskillende afrigtingsalgoritmes

Hierdie vergelyking toon aan dat RP die beste ten opsigte van veralgemeningsvermoë is en die minste epogge nodig het om die doelwit te bereik. Ons sal dus voortaan op RP konsentreer.

Vroeë beëindiging

Vroeë beëindiging word gebruik om te voorkom dat 'n neurale netwerk die afrigtingsdata memoriseer en die vermoë om te veralgemeen, verloor. Ten einde met die vorige resultate te kan vergelyk, rig ons weereens die neurale netwerk met dieselfde stel afrigtingspatrone in `WRD1tr.mat` af. Vir valida-

sie gebruik ons 'n soortgelyke datastel WRD10tr. Hierdie twee datastelle word deur die volgende MATLAB-opdragte aaneengeskakel (*concatenate*) en in `AFRIG.mat` gestoor:

```
>> load WRD1tr
>> P1=P; T1=T;
>> load WRD10tr
>> P2=P; T2=T;
>> P=[P1 P2];
>> T=[T1 T2];
>> save AFRIG P T
```

`AFRIG` bevat nou 'n $320 \times 20\,033$ invoermatriks P en 'n $1 \times 20\,033$ teikenvektor T. Die volgende program, `NNESval.m`, gebruik vroeë beëindiging om die neurale netwerk af te rig:

```
load('AFRIG');
S1 = 20;
[R,Q] = size(P);
[S2,Q] = size(T);
iitr=1:10088;
iival=10089:11089;
val.P=P(:,iival); val.T=T(:,iival);
ptr=P(:,iitr); ttr=T(:,iitr);
net = newff(minmax(P),[S1, S2],{'logsig' 'logsig'},'trainrp');
net.trainParam.goal = 0.015;
net.trainParam.show = 20;
net.trainParam.epochs = 200;
[net,tr] = train(net,ptr,ttr,[],[],val);
```

Die netwerk word met die eerste 10 088 invoerpatrone (wat `WRD1tr.mat` verteenwoordig) afgerig en met die eerste 1 000 invoerpatrone van `WRD10tr.mat` gevalideer. Die resultate verskyn in Tabel 3.8.

Aantal neurone	Aantal epogge	Gemiddelde prestasie met nuwe data	Prestasie met afrigtingsdata
20	41	95,06	98,47
40	41	95,01	98,61
60	38	95,09	98,64
80	40	95,16	98,61
100	35	95,13	98,56

Tabel 3.8: Vroeë beëindiging

Geen memorisering van afrigtingsdata het voorgekom nie. Vroeë beëindiging was dus nie nodig nie en afrigting kon telkens voortgaan totdat die doelwit bereik is. Die resultate is dus dieselfde as wat sonder vroeë beëindiging verkry is.

Meer en beter afrigtingsdata

In die proses om afrigtingsdata uit die elektroniese HAT te onttrek, kon alle moontlikhede nie aanvanklik voorsien word nie. Met nadere ondersoek is opgemerk dat die volgende gebreke in die datastel voorkom:

- herhaling van woorde;
- woorde met ontbrekende dele – waarskynlik as gevolg van nie-woordkarakters wat gebruik is;
- verkeerde lettergreepverdelings; en
- ontbrekende koppeltekens.

Daar kan nie verwag word dat 'n neurale netwerk wat met foutiewe data afgerig is, goeie resultate sal lewer nie. Die lys van woorde is deurgewerk en skoongemaak om uiteindelik 'n datalêer `AFKAPSK.dat` met 51 080 woorde te kry. Tien *willekeurige* datalêers (`RND1.dat` ... `RND10.dat`) elk met 1 000 woorde is uit `AFKAPSK.dat` te onttrek. Die PERL-program wat hierdie taak verrig, verskyn in Bylae A.

Die MATLAB-program `VolAlf.m` is weereens gebruik om hierdie datalêers na bruikbare invoerdata vir die afrigting van neurale netwerke om te skakel. Die afrigtingsdata is in RND1tr ... RND10tr gestoor. Die aantal afrigtingspatrone in elke lêer verskyn in die volgende tabel:

RND1tr	RND2tr	RND3tr	RND4tr	RND5tr	RND6tr	RND7tr	RND8tr	RND9tr	RND10tr
8 276	8 238	8 100	8 041	8 345	8 022	8 260	8 132	7 984	8 221

Ons gebruik `RND1tr.mat` om af te rig. Hierdie nuwe, verbeterde data lewer die resultate in Tabel 3.9.

Aantal neurone	Aantal epogge	Gemiddelde prestasie met nuwe data	Prestasie met afrigtingsdata
40	35	95,61	98,73
60	33	95,68	98,62
80	37	95,88	98,80
100	42	95,74	98,79

Tabel 3.9: Afrigting met verbeterde data

Alhoewel die neurale netwerk met 10 088 − 8 276 = 1 812 minder afrigtingspatrone afgerig is, is die prestasie beter as voorheen. Die netwerk met 80 neurone in die verborge laag het weereens die beste presteer. Hierdie prestasie is 95, 88 − 95, 26 = 0, 62 persent hoër as die beste prestasie met onverbeterde data.

Om te bepaal watter effek groter afrigtingsdatastelle op die prestasie van die neurale netwerk sal hê, skakel ons eers `RND1tr` en `RND2tr` aaneen om die datastel TWEE te kry wat uit 16 514 afrigtingspatrone bestaan. Ons toets die resulterende netwerk deur die gemiddelde prestasie oor vyf datastelle, naamlik `RND3tr` ... `RND7tr` te bepaal. Daarna voeg ons `RND3tr` by TWEE om datastel DRIE met 24 614 afrigtingspatrone te kry en toets met `RND4tr` ... `RND8tr`, ensovoorts. Die resultate verskyn in Tabel 3.10.

	EEN (8 276)			TWEE (16 514)			DRIE (24 614)			VIER (32 655)		
		Data			Data			Data			Data	
S1	Epog	Nuut	Afrig	Epog	Nuut	Afrig	Epog	Nuut	Afrig	Epog	Nuut	Afrig
40	35	95,61	98,73	48	96,33	98,63	57	96,91	98,52	–	–	–
60	33	95,68	98,62	45	96,40	98,61	56	96,91	98,48	–	–	–
80	37	95,88	98,80	46	96,45	98,62	50	97,02	98,56	60	97,16	98,57
100	42	95,74	98,79	43	96,44	98,61	52	96,94	98,58	–	–	–

Tabel 3.10: Prestasie met meer afrigtingspatrone

Die netwerke is telkens as `netEEN20.mat`, `netTWEE80.mat`, ens. gestoor na gelang van die afrigtingsdata wat gebruik is en die aantal neurone in die verborge laag.

Let wel: As gevolg van rekenaarbeperking is groter afrigtingsdatalêers nie oorweeg nie.

Uit hierdie resultate is dit duidelik dat die prestasie beslis met meer afrigtingsdata verbeter. Die neurale netwerk met 80 neurone wat met 32 655 patrone afgerig is (`netVIER80`) lewer die beste prestasie as dit met nuwe data getoets word. Dit het gemiddeld 97,16% van die afkappingsposisies in die 5 000 woorde van `RND5` ... `RND9` reg geklassifiseer.

Dit is ook duidelik dat 80 neurone in die verborge laag die beste resultate lewer vir hierdie probleem.

Aangesien ons vroeër gesien het dat 'n 50-10-1 netwerk beter resultate as die netwerk met 80 neurone in een verborge laag gee, rig ons nou ook netwerke met twee verborge lae met die datastel `VIERtr` af. Die resultate verskyn in Tabel 3.11.

Aantal neurone in laag		Aantal	Gemiddelde prestasie met	Prestasie met
S1	S2	epogge	nuwe data	afrigtingsdata
50	10	49	97,02	98,48
80	10	45	97,09	98,50
30	20	65	97,08	98,48

Tabel 3.11: Neurale netwerke met twee verborge lae

Hierdie netwerke lewer nie beter resultate nie, en word dus nie verder oorweeg nie.

Kort woorde bygevoeg

Die moontlikheid dat die byvoeging van kort woorde (dit is woorde wat nie in lettergrepe verdeel kan word nie) die neurale netwerk sal verbeter, word vervolgens ondersoek. In Bylae A word 'n lys van 2 410 kort woorde uit die ELHAT onttrek en in KORT.dat gestoor. Deur middel van die program `VolAlf.m` word hierdie woorde in 7 346 afrigtingspare omgeskakel en in `KORTtr.mat` gestoor.

`KORTtr.mat` word nou saam met `EEN`, `TWEE`, ensovoorts gevoeg om EENKORT, TWEEKORT, ens. te vorm. In Tabel 3.12 verskyn die prestasie van twee netwerke wat met hierdie data afgerig is (`netEENKORT` en `netTWEEKORT`) asook van die netwerke sonder kort woorde (`netEEN` en `netTWEE`). Netwerke met 80 neurone in die verborge laag is telkens gebruik.

Met kort woorde				Sonder kort woorde			
Netwerk	Aantal epogge	Nuwe data	Afrigtings-data	Netwerk	Aantal epogge	Nuwe data	Afrigtings-data
netEENKORT	24	95,31	97,14	netEEN80	37	95,88	98,80
netTWEEKORT	34	96,21	97,85	netTWEE80	46	96,45	98,62

Tabel 3.12: Die effek van kort woorde

Uit bostaande vergelyking is dit duidelik dat die netwerke telkens swakker presteer met die kort woorde bygevoeg as daarsonder. Dit lyk dus nie asof enige voordeel uit die byvoeging van kort woorde spruit nie en dit sal nie verder ondersoek word nie.

Afrigting sal voortaan sonder die byvoeging van kort woorde gedoen word, maar die finale netwerk sal op data wat kort woorde insluit, getoets word.

3.5.4 Om die neurale netwerk te gebruik

Die neurale netwerk kan op twee maniere aangewend word. Dit kan gebruik word om 'n enkele woord wat in MATLAB se opdragvenster ingevoer word, of 'n stel woorde wat deur middel van die *Import Wizard* ingevoer word in lettergrepe te verdeel.

'n Enkele woord

Die program, `NNCEBR1.m`, wat hier volg, verdeel een woord wat in die MATLAB-opdragvenster ingevoer is, in lettergrepe.

```
K=['-' 'a' '\"{a}' 'e' '\"{e}' '\^{e}' '\'{e}' '\'{e}' 'i' '\"{\i}';
   'o' '\"{o}' '\^{o}' 'u' '\"{u}' '\^{u}' 'y' 'b' 'c' 'd';
   'f' 'g' 'h' 'j' 'k' 'l' 'm' 'n' 'p' 'q';
   'r' 's' 't' 'v' 'w' 'x' 'z' '!' '#' '='];    %! verteenwoordig '

D=[ 1  2  3  4  5  6  7  8  9 10;
   11 12 13 14 15 16 17 18 19 20;
   21 22 23 24 25 26 27 28 29 30;
   31 32 33 34 35 36 37 38 39  1];              %= ook afkapping

N=0;                                            %Teller vir vensters.
S=lower(char(woord));                           %Ingevoerde woord.
WL=length(S);
W1=zeros(WL,1);
for k = 1:WL
    for i = 1:4
        for j = 1:10
            if S(k) == K(i,j)
                W1(k) = D(i,j);
            end
```

```
        end
    end
end
W=W1;
Pp = zeros(8,40);
SW = [1; 1; 1; W; 1; 1; 1];                    %Vensters.
for z = 1:WL-1
    WW = SW(z:z+7);
    Venster = WW';
    for i = 1:8
        Pp(i,WW(i),z) = 1;
    end
    N=N+1;
    PI(:,:,N) = Pp(:,:,z)';
end
for i=1:N
    Q=PI(:,:,i);
    p(:,i)=Q(:);
end
load netVIER80;                                %Laai neurale netwerk
a=sim(net,p);                                  %Simuleer
T=round(a);                                    %Afkappingsposisies
t=[T 0 0]
KT=0;
for k=1:WL+KT
    if t(k) == 1
        KW(k+KT) = S(k);
        KT = KT+1;
        KW(k+KT) = '-';
        for i=k+KT+1:WL+KT
            KW(i)=S(i-KT);
        end
    else KW(k+KT)=S(k);
    end
end
Lettergrepe = KW
clear
```

Hierdie program

- ▷ lees een ingevoerde woord vanaf die MATLAB-opdragvenster in,
- ▷ skakel dit om na invoerdata vir die neurale netwerk,
- ▷ laai die neurale netwerk (`netVIER80`),
- ▷ bepaal die die uitvoer van die neurale netwerk (die afkapposisies),
- ▷ gebruik die uitvoer om lettergreepverdeling te doen,
- ▷ lewer die woord in lettergrepe verdeel.

Indien die woord *oordragfunksie* ingevoer word en die program `NNGEBR1.m` word geroep, kry ons die volgende resultaat:

```
>> woord='oordragfunksie';
>> NNGEBR1

Lettergrepe =

oor-drag-funk-sie
```

Verskeie woorde

Ons wil ook verskeie woorde wat in 'n lêer voorkom deur middel van die neurale netwerk in lettergrepe verdeel. Die volgende program, `NNGEBR.m`, word gebruik om tien woorde in die datastel `TOETS.dat` in lettergrepe te verdeel en die resultate in 'n uitvoerlêer `TOETS.out` te plaas. `TOETS.dat` word deur middel van MATLAB se *Import Wizard* ingevoer.

```
K=['-' 'a' '\"{a}' 'e' '\"{e}' '\^{e}' '\'{e}' '\`{e}' 'i' '\"{\i}';
   'o' '\"{o}' '\^{o}' 'u' '\"{u}' '\^{u}' 'y' 'b' 'c' 'd';
   'f' 'g' 'h' 'j' 'k' 'l' 'm' 'n' 'p' 'q';
   'r' 's' 't' 'v' 'w' 'x' 'z' '|' '#' '='];

D=[ 1  2  3  4  5  6  7  8  9 10;
   11 12 13 14 15 16 17 18 19 20;
   21 22 23 24 25 26 27 28 29 30;
   31 32 33 34 35 36 37 38 39  1];

fid = fopen('TOETS.out','w');          %Maak l\^{e}er oop om in te skryf.

for n=1:10                             %Aantal woorde in l\^{e}er.
    N=0;
    clear S WL SW T p Q KW;
    S=lower(char(TOETS(n)));           %Woord uit TOETS.dat.
    WL=length(S);
    W1=zeros(WL,1);
    for k = 1:WL
        for i = 1:4
            for j = 1:10
                if S(k) == K(i,j)
                   W1(k) = D(i,j);
                end
            end
        end
    end
    W=W1;
    Pp = zeros(8,40);
    SW = [1; 1; 1; W; 1; 1; 1];
```

```
    for z = 1:WL-1
        WW = SW(z:z+7);                    %Venster
        for i = 1:8
            Pp(i,WW(i),z) = 1;
        end
        N=N+1;
        PI(:,:,N) = Pp(:,:,z)';
    end
    for i=1:N
        Q=PI(:,:,i);
        p(:,i)=Q(:);
    end

    load netVIER80;                        %Laai neurale netwerk.
    a=sim(net,p);                          %Simuleer.
    T=round(a);
    t=[T 0];                               %Uitvoer van NN.
    KT=0;                                  %Woordlengte vermeerder met KT.
    for k=1:WL+KT
        if t(k) == 1                       %As t=1 plaas koppelteken in posisie.
            KW(k+KT) = S(k);
            KT = KT+1;
            KW(k+KT) = '-';
            for i=k+KT+1:WL+KT
                KW(i)=S(i-KT);
            end
        else KW(k+KT)=S(k);
        end
    end
    y = KW;
    fprintf(fid,'%s\n',y);                 %Skryf na l\^{e}er TOETS.out.
end
status = fclose(fid);                      %Maak l\^{e}er toe.
```

Die uitvoer in TOETS.out verskyn in Tabel 3.13.

Beskou die volgende foute van die neurale netwerk `netVIER80`:

- ▷ Dit het nagelaat om 'n koppelteken na die *a* in *atro-de-se* te plaas. Die gesimuleerde uitvoere van die netwerk, naamlik

  ```
  Sim_uitvoer = 0.2801 0.2224 0.0049 0.9955 0.0037 0.9996 0.0000
  ```

  toon dat die uitvoer vir 'n koppelteken na die *a* wel hoër is as dié vir posisies waar geen koppelteken moet kom nie, maar nie naastenby hoog genoeg nie. (Let daarop dat geen uitvoer vir die laaste letter in die woord gegee word nie.)

- ▷ Die woord *ve-re-nig-baar* is verkeerd afgekap. Die gesimuleerde uitvoer is

  ```
  Sim_uitvoer = 0.0000 0.6889 0.0188 0.9767 0.0002 0.0183 0.9778 0.0029 0.0139 0.0000
  ```

atro-de-se
be-ïn-vloed
ek-stro-vert
on-buds-man
vo-ël-tjie
stel-sel-ont-le-der
ve-re-nig-baar
ver-moei-end
waar-s-kyn-lik-heids-leer
sel-foo-n-aan-bod

Tabel 3.13: `TOETS.out` met netVIER80

Hier is die uitvoer vir die verkeerde afkapping weer baie laer as dié vir die korrekte afkappings, maar steeds redelik hoog.

- In *waar-s-kyn-lik-heids-leer* verskyn'n ongeoorloofde koppelteken. Die gesimuleerde uitvoer is

```
Sim_uitvoer = 0.0000 0.0147 0.0012 0.6709 0.6487 0.0002 0.0527 0.9922 0.0000

        0.0009 0.9941 0.0000 0.0000 0.0809 0.3855 0.5969 0.0027 0.0127 0.0000
```

Die uitvoer vir die korrekte koppelteken is 0,6709 teenoor 0,6487 vir die verkeerde een. Alhoewel die regte een se waarde wel hoër as dié van die verkeerde een is, is beide redelik hoog en na aan mekaar.

Dit is dus duidelik dat daar baie teenstrydighede bestaan. Verdere verbetering van die netwerk is nodig.

3.5.5 Verdere verbetering van afrigtingsdata

Deur middel van 'n paar lukraak eksperimente is bepaal dat afkapping verkeerd gedoen word indien

- die tweede deel van 'n saamgestelde woord met 'n vokaal begin (soos verenigbaar, doodeenvoudig, eienaardig, grondeienaar, ens.); en
- vreemde letterkombinasies (wat nie in die afrigtingsdata voorgekom het nie) in woorde voorkom, soos atrodese, kalkswael, majestueus, ens.

Vreemde letterkombinasies en saamgestelde woorde waarvan die tweede deel met 'n vokaal begin, is per hand uit die datalêer `AFKAPSK.dat` gesoek en in die datalêer `KOMB.dat` geplaas. Hierdie 777 woorde is met `VolAlf.m` na die invoermatriks P en teikenvektor T omgeskakel en in `KOMBtr.mat` gestoor. `VIERtr.mat` en `KOMBtr.mat` is aaneengeskakel en in `VIERKOMB.mat` gestoor. Die neurale netwerk is met hierdie 39 302 afrigtingspare afgerig. Die resultate met doelwit $MSE = 0,015$ en $MSE = 0,01$ verskyn in Tabel 3.14.

Let op die verbetering in veralgemeningsvermoë indien die netwerk afgerig word tot die gemiddelde kwadraatfout minder as 0,01 is. (Dit het 'n Pentium 4 rekenaar met 1,4 gigahertz verwerkerspoed omtrent 8 uur geneem om hierdie netwerk af te rig.)

Naam	Doelwit: MSE	Aantal epogge	Prestasie met nuwe data	Prestasie met afrigtingsdata
netVIERKOMB	0,015	80	97,45	98,63
net4KOMB	0,01	118	97,56	99,50

Tabel 3.14: Data verder verbeter

Indien ons hierdie netwerk gebruik om die woorde in `TOETS.dat` af te kap, kry ons die resultaat in Tabel 3.15.

a-tro-de-se
be-ïn-vloed
eks-tro-vert
on-buds-man
vo-ël-tjie
stel-sel-ont-le-der
ver-e-nig-baar
ver-moei-end
waar-skyn-lik-heids-leer
sel-foon-aan-bod

Tabel 3.15: `TOETS.out` met `net4KOMB`

Al die woorde is nou reg afgekap. Hulle kom egter in `KOMB.dat` voor en is dus deel van die afrigtingsdata. Let op hoe hoog die gesimuleerde uitvoere vir die afkappingsposisies in die volgende woorde is:

- *atrodese*

```
Sim_uitvoer = 0.9200  0.1219  0.0433  0.9985  0.0582  0.9994  0.0000
```

- *verenigbaar*

```
Sim_uitvoer = 0.0000 0.0597 0.9566 0.9816 0.0014 0.0120 0.9850 0.0045 0.0111 0.0001
```

Toets met onbekende data

Om vas te stel of die neurale netwerk `net4KOMB` werklik beter as `netVIER80` presteer, word uittreksels uit 'n paar artikels in die Sarie van 9 Januarie 2002 beskou. Die woorde uit die uittreksels is soos volg aangepas ten einde sinvolle data te verkry:

- Die woorde is alfabeties gerangskik met elke woord op 'n nuwe reël.
- Alle herhalings van woorde is verwyder sodat elke woord net een keer voorkom.
- Engelse woorde soos *queen*, *fellowship*, *maintenance*, ens. is verwyder.
- Eiename van vreemde herkoms soos *George*, *Arthur*, *Tchaikofsky*, ens. is verwyder, maar Afrikaanse eiename soos *Klaradyn* en *Paternoster* nie.

Die netwerke netVIER80 en net4KOMB is met hierdie 510 woorde, wat 2 636 moontlike afkappingsposisies bevat, geëvalueer. Die woorde wat deur die netwerke verkeerd afgekap is, verskyn in Tabel 3.16. Die netwerk netVIER80 het foute in 35 woorde gemaak en net4KOMB 30. Die netwerk net4KOMB het in drie woorde elk twee foute gemaak (aangedui met 'n pyltjie). Die twee netwerke het dieselfde 17 woorde (gemerk met 'n ⋆) verkeerd afgekap.

		netVIER80		net4KOMB
1	⋆	agtja-ri-ge	⋆	ag - tja-ri-ge
2		au-gustus	⋆	baie
3	⋆	baie	⋆	be -s- ka-wing - sont-wik-ke-ling →
4	⋆	be-ska-wing -s- ont-wik-ke-ling	⋆	be - ste
5	⋆	be - ste		feetjies
6		dansskoe-ne		ge -s- on-de
7		ein -t- lik	⋆	ge -s- wig
8		fê - nsie		karmo-syn
9		gea-mu-seerd	⋆	lae
10	⋆	ge -s- wig	⋆	lankal
11		glimlag	⋆	leen-tjie - sklip
12	⋆	lae	⋆	me - teens
13	⋆	lan - kal	⋆	mi - v–vigs
14	⋆	leen-tjie -s- klip	⋆	na-vor-sing - sraad
15	⋆	me - teens	⋆	po-niestert
16	⋆	mi - v–vigs	⋆	pre-si-dent - sak-ker
17	⋆	na-vor-sing - sraad	⋆	re - goor
18		on-weer - swol-ke		saal -s- ak-ke
19	⋆	po-nie -s- tert		seepunt
20	⋆	pre-si-dent - sak-ker		sek-sue-le
21	⋆	re - goor	⋆	skoo - lu-niform →
22		s - chee-pers		stof-ko -r- rel-tjie
23		sexy		sui-ker -k- lont–feetjie →
24	⋆	skoo - lu-ni-form	⋆	tea-ters
25		sla - gor-kes		vanjaar
26		suid–afri-ka		verd - wyn
27	⋆	tea-ters		vin -g- er-tjie
28		teen-ver-ou-de-ring - spro-duk-te		volkspe-le
29		te-rug -s- taar		vrag-mo -t- ors
30		ts - wa-na	⋆	wa - ar's
31		ver-beel-din - gryk-ste		
32	⋆	wa - ar's		
33		wer - klik		
34		wer - klik-heid		
35		weskus		

Tabel 3.16: Afkappingsfoute deur netVIER80 en net4KOMB gemaak

Die volgende soorte foute het voorgekom:

1. 'n Lettergreep word nie met 'n koppelteken aangedui nie, soos in *agtja-ri-ge* en *glimlag* by `netVIER80` en *feetjies* en *lankal* by `net4KOMB`. Dit word in die tabel aangedui deur die posisie waar die koppelteken ontbreek, te onderstreep. Hierdie is minder ernstige foute, aangesien afkapping bloot nie sal plaasvind nie.

2. 'n Koppelteken verskyn op 'n verkeerde plek in die woord, soos in *be-ste* en *on-weer-swol-ke* by `netVIER80` en *ag-tja-ri-ge* en *verd-wyn* by `net4KOMB`. Hierdie is werklike foute aangesien dit tot verkeerde afkapping lei. Hier onderskei ons tussen twee soorte foute wat verdere aandag verdien:

   (a) Die koppelteken verskyn op die verkeerde plek in die woord. (Dit word met 'n grys blokkie aangedui.)

   (b) Waar die eerste deel van 'n saamgestelde woord op 'n konsonant eindig en die tweede deel met 'n vokaal begin, word voor die konsonant afgekap. (Dit word aangedui deur te omkring.)

3. Koppeltekens verskyn weerskante van 'n konsonant, soos in *po-nie-s-tert* in `netVIER80` en *ein-t-lik* in en *ge-s-on-de* in `net4KOMB`. Hierdie foute dui op onsekerheid en word deur 'n grys raampie aangedui.

By koppeltekenwoorde soos *Suid-Afrika*, *hop-badkamer* en *MIV-vigs* verskyn die koppelteken as 'n dubbelkoppelteken in die uitvoer. Dit word so gelaat om die onderskeid tussen die twee soorte koppeltekens te behou. Dit kan tydens die werklike afkappingsproses gebruik word.

'n Vergelyking van die fout-uitvoere van die twee neurale netwerke in terme van bogenoemde soorte foute lewer die resultate in Tabel 3.17.

	`netVIER80`		`net4KOMB`	
Foute	Aantal	%	Aantal	%
Totaal	35	1,33	33	1,25
Soort 1	11	0,42	13	0,49
Soort 2	18	0,68	12	0,46
Soort 3	6	0,23	8	0,30
Woorde	35	6,86	30	5,88

Tabel 3.17: Vergelyking van uitvoere

Ons kom tot die gevolgtrekking dat `net4KOMB` wel beter met onbekende woorde presteer as `netVIER80` en wel om die volgende redes:

- ▷ Die totale persentasie foute is laer;
- ▷ Die persentasie van soort 1 foute (nie ernstig) is hoër;
- ▷ Die persentasie van soort 2 foute (werklike foute) is laer;
- ▷ Die persentasie van soort 3 foute (onsekerheid) is hoër;
- ▷ Die persentasie woorde wat verkeerd afgekap is, is laer.

Beskou die gesimuleerde uitvoere van soort 3 foute:

- Die woord *gesonde* wat as *ge-s-on-de* afgekap is:
  `Sim_uitvoer = 0.0001 0.9951 0.6105 0.0032 0.9990 0.0000`

- Die woord *geswig* wat as *ge-s-wig* afgekap is:
  `Sim_uitvoer = 0.0000 0.9789 0.6180 0.0000 0.0015`

- Die woord *vingertjie* wat as *vin-g-er-tjie* afgekap is:
  `Sim_uitvoer = 0.0000 0.0006 0.9642 0.5938 0.2249 0.8868 0.0347 0.0000 0.0003`

Dit is duidelik dat die uitvoer vir die tweede (verkeerde) koppelteken telkens heelwat laer as die korrekte een is. Hierdie probleem kan dus tydens 'n nasorgprogram reggestel word deur, in die geval waar koppeltekens weerskante van 'n konsonant voorkom, die uitvoere met mekaar te vergelyk en 'n koppelteken slegs in die posisie met die die grootste uitvoerwaarde te plaas.

In die tabel op bladsy 90 verskyn die volledige resultate van `net4KOMB` op 510 woorde uit Sarie van 9 Januarie 2002. Dit wys hoe suksesvol die neurale netwerk vreemde woorde in lettergrepe verdeel het. Geen kort woorde is afgekap nie en baie komplekse saamgestelde woorde is suksesvol in lettergrepe verdeel. Die 30 woorde met afkappingsfoute verskyn in grys blokkies.

3.6 *Goeie* en *slegte* afkappingsfoute

Die drie soorte foute wat in die vorige afdeling geïdentifiseer is, kan in twee groepe verdeel word, naamlik

- 'n koppelteken word tussen geldige lettergrepe weggelaat; en

- 'n koppelteken verskyn op 'n ongeoorloofde plek in die woord.

Ons sê die eerste soort fout is 'n *goeie* fout, want alhoewel afkapping nie in daardie posisie sal plaasvind nie, behoort daar in lang woorde wel ander moontlike afkappingspunte in die nabyheid te wees. Die tweede soort is egter 'n *slegte* fout, aangesien dit 'n verkeerde afkapping kan veroorsaak.

Die program `NNSTAT.m` is aangepas om tussen hierdie twee soorte foute te onderskei. Die aangepaste program is soos volg:

```
%%%%%%%%%%%%%
%NNSTATGS.m%
%%%%%%%%%%%%%
load RND10tr;
[R,N] = size(P);
load net4KOMB;              %Laai neurale netwerk
a=sim(net,P);               %Simuleer met toets P
t=round(a);                 %Teiken van toetsdata

tel1=0; tel2=0; tel3=0; tel4=0;
for i=1:N;
```

```
    if (t(i)==1 & T(i)==1|t(i)==0 & T(i)==0);
        tel1=tel1+1;
    end;
    if (t(i)==0 & T(i)==1);
        tel2=tel2+1;
    end;
    if (t(i)==1 & T(i)==0);
        tel3=tel3+1;
    end
end;
STAT1=tel1/N*100;
Korrek = STAT1              %Persentasie korrekte afkappings
STAT2=tel2/N*100;
Goeie_foute = STAT2         %Persentasie goeie foute
STAT3=tel3/N*100;
Slegte_foute = STAT3        %Persentasie slegte foute
```

As ons net4KOMB met die afrigtingsdata in VIER evalueer, kry ons die volgende resultate:

```
>> NNSTATGS

Korrek = 99.5008

Goeie_foute = 0.2756

Slegte_foute = 0.2235
```

Hieruit is dit duidelik dat die netwerk slegs 0,2235% van die 32 655 moontlike afkappingsposisies in VIER werklik verkeerd klassifiseer. Dit doen 99,5008% korrek en laat na om 0,2756% van die koppeltekens in te sit. Aangesien die netwerk tot 'n *MSE* van 0,01 afgerig is, verwag ons dat dit 99% van die afrigtingsdata korrek sal hanteer. In hierdie geval hanteer dit $99,5008 + 0,2756 = 99,7764\%$ korrek of aanvaarbaar.

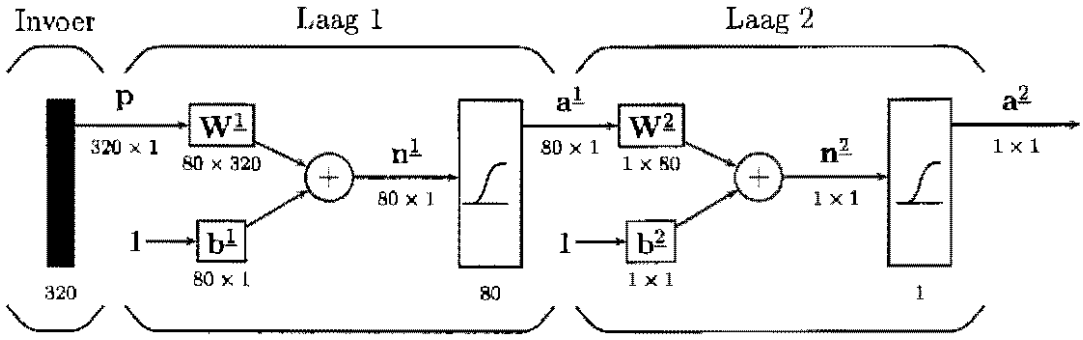
Met die onbekende data in RND5 ... RND9 is die gemiddelde resultate soos volg:

Korrek:	97,56%
Goeie foute:	1,30%
Slegte foute:	1,15%

Die netwerk maak dus gemiddeld 1,15% foute met onbekende data en lewer aanvaarbare resultate in $97,56 + 1,30 = 98,86\%$ van die gevalle.

3.7 Argitektuur van die netwerk

Die argitektuur van die twee-laag (80-1) vorentoe-voer terugpropageringsnetwerk wat die beste resultate gelewer het, verskyn in Figuur 3.6. Let op die volgende eienskappe van die neurale netwerk:



Figuur 3.6: Argitektuur van `net4KOMB`

- Die invoer bestaan telkens uit 'n kolomvektor $\mathbf{p}$ met 320 elemente uit die invoermatriks $\mathbf{P}$. Die invoer word nie as 'n laag van die netwerk gereken nie.
- Laag 1:
  - Die gewigte verskyn in $\mathbf{W}^1$ ('n 80×320 matriks).
  - Die beladings verskyn in $\mathbf{b}^1$ ('n kolomvektor met 80 elemente).
  - Die netto invoer word gegee deur $\mathbf{n}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1$.
  - Die uitvoer $\mathbf{a}^1$ word deur die *log-sigmoidale* funksie uit $\mathbf{n}^1$ bepaal ('n kolomvektor met 80 elemente). Dit is ook Laag 2 se invoer.
- Laag 2:
  - Die gewigte verskyn in $\mathbf{W}^2$ ('n ryvektor met 80 elemente).
  - 'n Enkele belading verskyn in $\mathbf{b}^2$.
  - Die netto invoer word gegee deur $\mathbf{n}^2 = \mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2$ ('n enkele waarde).
  - Die uitvoer $\mathbf{a}^2$ is 'n enkele waarde en word deur die *log-sigmoidale* funksie uit $\mathbf{n}^2$ bepaal.
- Die uitvoer van die netwerk is die afgeronde waarde van $\mathbf{a}^2$ en dui vir elke venster ($\mathbf{p}$) aan *ja* kap af (1) of *nee* moenie afkap nie (0).

3.8 Gevolgtrekking

Dit is duidelik dat neurale netwerke wel as effektiewe afkappingstegniek vir Afrikaans gebruik kan word, aangesien 'n netwerk wat met minder as 5 000 woorde afgerig is, meer as 98% van afkappingsposisies in vreemde woorde reg klassifiseer.

In die studie wat uitgevoer is, is op lettergreepverdeling gekonsentreer. Die proses van afkapping wat die besluit oor waar 'n woord aan die einde van 'n reël verdeel moet word, behels, is hierop gebaseer en vereis verdere navorsing.

Die volgende areas vir moontlike verbetering behoort verder ondersoek te word:

- ▷ Datavoorbereiding
  - Deur die afrigtingsdata planmatig op grond van taalkundige kennis te kies, eerder as om willekeurige woorde te gebruik, kan beter resultate moontlik verkry word.
  - Die probleem van ongewone letterkombinasies wat as gevolg van die samestelling van woorde ontstaan, sal waarskynlik nooit heeltemal uitgeskakel kan word nie, maar kan aangespreek word deur soveel moontlik letterkombinasies in die afrigtingsdata in te sluit.
- ▷ Afrigting
  - Die enkoderingsmatriks (Tabel 3.2) kan aangepas word om sekere taalkundige eienskappe te weerspieël. 'n Taalkundige ontleding van die frekwensie waarmee sekere letters saam voorkom, mag daartoe lei dat die letters anders op mekaar moet volg en dalk anders saam gegroepeer moet word om beter resultate te kry.
  - Daar kan ook met die toekenning van syferwaardes geëksperimenteer word, soos om byvoorbeeld waardes tussen 0 en 1, eerder as heelgetalle, te gebruik.
  - Deur 'n kragtiger rekenaar en/of 'n ander neurale-netwerkpakket of programmeertaal te gebruik, kan 'n neurale netwerk met meer woorde (sê 10 000) afgerig word. Die prestasie behoort verder te verbeter.
- ▷ Uitsetontleding

  Deur die gesimuleerde uitvoere van probleemgevalle te bestudeer, kan oplossings moontlik geïdentifiseer word, soos in die geval van koppeltekens weerskante van konsonante.

Afkappingsresultate van net4KOMB met 510 woorde uit Sarie van 9 Januarie 2002:

'n
aan
aan-gaan
aan-ge-bou
aan-ge-dui
aan-ge-sit
af
af-ge-ry
a-fri-ka
af-stoot
agt
ag-tja-ri-ge
al
al-leen
al-mal
al-tyd
an-der
ant-woord
as
a-sem
au-gus-tus
baie
bak-kies
be-derf
be-geer-lik
be-gin
be-gra-we
be-sef
be-sem-stok
be-s-ka-wing-sont-wik-ke-ling
be-ste
bin-ne-ver-sie-ring
blad-sy
bloem-fon-tein
bloem-fon-tei-ner
blou
bo
boe-ke
bog
boon-ste
bor-ste
bran-ders
breed
bring
bro-se
bui-te
by
daar-mee
daar-voor
dag
dan
dans
dan-sen-de
dans-skoe-ne

ge-woon-te
glim-lag
glo
goed
goed-ko-per
graad
graf-ste-ne
gra-niet-steen
gras
grond-slag
groot
groot-kop
groot-te
grys
haar
hal-lo
har-de
har-ders
ha-re
hart
hê
heel-te-mal
het
heup
heu-pe
hier
hoe
hom
hom-pe
hon-derd
hoor
hop-bad-ka-mer
hou
hou-ding
hout-vloer
huis
hui-sie
huis-maat
hul
hul-le
hy
i-den-ti-fi-seer
il-lu-sies
in
in-kar-na-sie
in-kom
in-kom-mers
in-wo-ners
is
jaar
ja-re
je-sus
jou
jou-self

mi-v-vigs
mo-del
moet
mon-ster
mooi
mu-siek
muur
my
my-ne
na
na-by
na-dat
na-me
na-vor-sing-sraad
nei-gings
nes
net
nie
nie-mand
niks
nog
nou
nu-we
of
o-gies
om
om-gang
om-ge-val-le
om-trent
on-der
on-dier
on-langs
on-per-soon-li-ke
ons
ont-vou
on-ver-sorg-de
on-vol-maak-te
on-weers-wol-ke
ook
oor
oor-bel
oor-bluf
oor-da-di-ge
op
op-ge-hang
ou
pa
paar
pad
pak
pas-sies
pa-ter-nos-ter
per
plan-ne

stap-pe
stem-mig
stil-hou
stil-weg
stoel
stof-ko-r-rel-tjie
storm
stra-te
strik-kies
stuk-ken-de
stuk-kie
styf
suid
suid-a-fri-ka
sui-de-li-ke
sul-ker-k-lont-jie
suk-ses
su-per-mo-del
su-per-teef
sus
swaar-te-krag
swart
sy
sy-'t
taal
tas
te
tea-ters
teen
teen-ver-ou-de-rings-pro-duk-te
tel
te-rug-staar
ter-wyl
tien
toe
toe-ge-neem
tog
toi-let
tot
tra-di-si-o-neel
trap
trek
tri-lo-gie
troos-kos
troos-prys
trou-plan-ne
tswa-na
tuis-ge-bring
tus-sen
uit-spoel
uit-trek
uur
val
van

da-rem
dat
deel
deel-ne-mers
der-de
des-pe-raat
de-tails
deur
die
die-self-de
dig-bun-del
dik
dink
dis
dit
don-ker
dood
dood-slaan
draai
dras-ties
dreun
drup-pel
duld
een
eens
een-ver-trek-huis
eer-ste
eeu
ef-fe
ei-land-va-kan-sie-na-me
eint-lik
ei-se
ek
el-ke
en
en-gel
en-gel-ge-sig-gie
en-gels
e-ni-ge
fe-bru-a-rie
feetjies
fên-sie
film
flen-ter
flen-te-ri-ge
fliek
flir-ta-sie
fyn
gaan
gal-joen
ge-a-mu-seerd
ge-bid
ge-bo-re
ge-de-bu-teer
ge-gee
ge-glo
juf-frou
juf-frous-hoog-te
jy
kaal-kop
kan
karmo-syn
kat
kie-wiet
kie-wiet-vlug
kla-ra-dyn
klein-tyd
kle-re
klim
klip-bank
klip-hard
koes-slag
kom
kon
ko-ning
koop
kop
kort
kort-ston-di-ge
kos-blik
kraai
kreef
krimp
kry
kry-send
kui-kens
kui-te
kuns-wed-stryd
kwaai
kyk
laag
laas
laat-mid-dag
lae
lag
lank
lankal
la-ter
lê
leeg
leen-tjie-sklip
leer
leg-kaart
lied-jie
lief-ge-hê
lie-we
lood
loop
loop-baan
luis-ter
lyf
ma
plek
ple-sier
plonks
po-niestert
praat
pre-si-dent-sak-ker
pre-sies
raak-ge-sien
rak
re-goor
ri-a-na
ring
rings
rit-me
roep
ruk
ry
saal-s-ak-ke
saam
saam-pak
sag-te
sa-ha-ra
sal
sand-bult
sa-rie
schee-pers
se
sê
see
seepunt
seks
sek-sue-le
self-beeld
sel-lu-liet
sen-sus-man
ses
seun-tjie
sex-y
siek-te
sien
siens
sim-pel
sing
sit
skaam
skep-pers
skeur-tjies
skoe-ne
skool
skool-kon-sert
skoo-lu-niform
skrik
skry-wer
skry-wers-vrou
slag
slag-or-kes
vang
vanjaar
van-uit
vas
ver-beel
ver-beel-ding-ryk-ste
ver-by
ver-dien
verd-wyn
ver-fris-sen-de
ver-hoog
ver-krag
ver-krag-ting
ver-le-de
ver-skil
ver-soen-baar
ver-stom
ver-tel
ver-val
ver-wag
ver-won-derd
vigs
vin-g-er-tjie
vir
vloot
voel
voe-te
voe-te-val
volkspe-le
voor
voor-beel-di-ge
vo-ri-ge
vrag-mo-t-ors
vrees-be-van-ge
vrees-li-ke
vriend
vrien-de
vroe-ër
vrou
vrou-lik-heid
waar
wa-ar's
waar-van
wag
wan-neer
want
was
wat
week
wees
weet
wél
wê-reld
wê-reld-wyd
werk
werk-lik

ge-le-de
ge-lief-koos
ge-loof
ge-lyk-na-mi-ge
ge-nees
ge-noeg
ge-oe-fen
ge-praat
ge-sê
ge-sels
ge-sien
ge-sit
ge-s-on-de
ge-sond-heid
ge-staan
ge-s-wig
ge-troos
ge-wo-ne
maagd
maag-de
maak
maar
maat-jie
maat-jies
ma-gie-se
man
ma-triek
me-die-se
meer
mei-sie-kind
mens
men-se
met
me-teens
min-der
mi-te
slan-ke
slim
smaak
smelt
so
sok-kies
soms
son-bruin
soos
sou
speel
speel-ka-mer
spie-ël
spoel
spreek-woor-de-lik
staan
stan-derd
sta-pels
werk-lik-heid
werk-see
werk-ses-sies
wes-kus
wes-ter-se
wie-le
wil
wil-le-mien
wit
woon-buurt
word
wreed
wuif
wyd-ver-spreid
wy-sie
wys-maak

Bylae A

Data

A.1 Die bron van data wat gebruik word

ELHAT, die elektroniese weergawe van die *Verklarende Handwoordeboek van die Afrikaanse Taal* is deur die Universiteit van Stellenbosch uitgegee. Hierdie elektroniese woordeboek bevat al die Afrikaanse woorde (natuurlik sonder alle moontlike samestellings) met die korrekte aksente en lettergreepverdelings aangetoon. 'n Voorbeeld van die teks verskyn in Figuur A.1.

pof ww. (gepof)
In warm as of op 'n warm plaat braai, laat skiet: *Gepofte mielies, rys.*

pof'ad·der
Giftige slang wat hom opblaas as hy kwaad word; *Bitis arietans.*

pof'ad·der·gei·tjie
Klein geitjie van die Karoo met 'n kleurtekening soos dié van 'n pofadder; *Pachydactylus maculatus.*

pof'broek
Sakbroek.

pof'fer·tjie (-s)
Deegkoekie wat in 'n pot of pan met kokende vet of olie gebak word.

Figuur A.1: 'n Voorbeeld van teks in ELHAT

Let daarop dat die posisie waar die aksent van 'n sekere woord moet kom deur 'n aksentteken (') net na die lettergreep wat die aksent dra, aangedui word. Hierdie teken verteenwoordig ook 'n lettergreepverdeling. Verdere lettergrepe word deur 'n punt (·) aangedui. Let daarop dat dit nie 'n punt soos aan die einde van 'n sin is nie, maar in die middel van die reël voorkom.

In die reël voor elke woord waarin die aksente en lettergreepverdelings aangedui word, verskyn die woord sonder enige spesiale tekens. Hierdie woorde is in wit gedruk sodat dit nie op die skerm sigbaar is nie. Die uitsluitlike doel van hierdie woorde is om die soektog na woorde moontlik te maak. In Figuur A.2 verskyn dieselfde teks as voorheen met die onsigbare woorde ingesluit.

pof gepof
pof ww. (gepof)
In warm as of op 'n warm plaat braai, laat skiet: *Gepofte mielies, rys.*
pofadder
pof·ad·der
Giftige slang wat hom opblaas as hy kwaad word; *Bitis arietans.*
pofaddergeitjie
pof·ad·der·gei·tjie
Klein geitjie van die Karoo met 'n kleurtekening soos dié van 'n pofadder; *Pachydactylus maculatus.*
pofbroek
pof·broek
Sakbroek.
poffertjie
pof·fer·tjie (-s)
Deegkoekie wat in 'n pot of pan met kokende vet of olie gebak word.

Figuur A.2: Teks uit ELHAT met onsigbare woorde ingesluit

A.2 'n Volledige datastel

'n Volledige datastel waarmee die neurale netwerk afgerig en getoets kan word, bestaan uit die volgende:

- woorde waarin die korrekte lettergreepverdelings aangedui word; en
- kort woorde sonder lettergreepverdelings.

Hierdie twee datastelle moet uit ELHAT onttrek word. Die programmeertaal PERL (*Practical Extraction and Report Language*) word gebruik om hierdie taak te verrig.

A.2.1 Woorde met lettergreepverdelings

Die volgende stappe is deurloop om die data voor te berei:

- Die sentrale teks (die hele woordeboekgedeelte) van ELHAT is as `HAT.dat` in die teksredigeerder WinEdt gelaai.
- Om die woorde met lettergreepverdelings uit ELHAT te kan onttrek, is dit nodig om nie-woordkarakters, soos koppeltekens en letters met deeltekens en kappies, deur woordkarakters te vervang. (Woordkarakters is die 26 letters van die alfabet in klein- of hoofletters, en die syfers van 0 tot 9.) Hierdie woordkarakters moet egter herkenbaar wees, aangesien dit later weer teruggeplaas moet word.
  - In ELHAT dui die karakter (') aan dat die klem op die voorafgaande lettergreep moet val en terselfdertyd dat dit 'n lettergreepverdeling is. Gewone lettergreepverdelings word met 'n middelpunt (·) aangedui. Beide die ' en die · word deur QQ vervang om koppeltekens aan te dui. (Dubbele hoofletters word gebruik sodat dit nie later nodig sal wees om te kontroleer dat woorde wat met 'n hoofletter Q moet begin, dalk met 'n koppelteken begin nie.) DOS (of Unix) se GSAR.exe (*General Search And Replace*) word gebruik om hierdie vervanging te doen. Die opdragte is soos volg:

* Die heksadesimale voorstelling van die karakter ' is B4. Die volgende opdrag in DOS vervang al die B4s in die lêer `HAT.dat` met QQ:

  gsar -s:xb4 -rQQ -o hat.dat

  [Hierdie opdrag sê: soek (-s) die heksadesimale karakter (:x) b4 en vervang (-r) dit met die karakter QQ. Oorskryf (-o) die oorspronklike data in die lêer `HAT.dat`.]
* Die heksadesimale voorstelling van die karakter · is B7. Die volgende opdrag in DOS vervang al die B7s in `HAT.dat` met QQ:

  gsar -s:xb7 -rQQ -o hat.dat

[Al die nodige veranderings word egter gelyk met 'n lotlêer in DOS gedoen, soos in die volgende punt aangetoon.]

– Die volgende karakters word ook vervang:

* Karakters soos ê, ë, ï, ens. word ook deur heksadesimale karakters aangedui en word nie as ASCII-karakters herken nie.
* Koppeltekens wat in woorde moet voorkom, soos in *nou-nou*, *direkteur-generaal*, ens. dui lettergreepverdelings aan maar moet nie dieselfde as gewone lettergreepverdelings hanteer word nie. Dit moet uiteindelik as sulks in die woord verskyn.
* Die verbindings-s in woorde soos *gevegsknoper* is opsioneel en word in ELHAT aangedui deur die *s* in hakies te plaas. Die woord verskyn dus as *geveg(s)knoper*. Daar is besluit om net die weergawe met die verbindings-s in die datalêer op te neem en die hakies word dus verwyder.
* Die afkappingsteken ' wat by woorde soos *pa'tjie*, *wag-'n-bietjie*, ens. voorkom word deur HN vervang.

Al die vreemde karakters, hul heksadesimale voorstelling (waar van toepassing) en die karakters waarmee dit vervang word, verskyn in die volgende tabel:

Vreemde karakter	Heksadesimale voorstelling	Vervang met
'	B4	QQ
·	B7	QQ
-		KT
ä	E4	AU
ë	EB	EU
ê	EA	EK
ï	EF	IU
ö	F6	OU
ô	F4	OK
ü	FC	UU
û	FB	UK
è	E8	EA
é	E9	AE
'	27	HN
(	28	
)	29	

'n Sogenaamde lotlêer genaamd `vorentoe.bat` is in DOS geskep om al hierdie veranderings aan te bring. Dit lyk soos volg:

```
gsar -s:xb4 -rQQ -o %1
gsar -s:xb7 -rQQ -o %1
gsar -s- -rKT -o %1
gsar -s:xe4 -rAU -o %1
gsar -s:xeb -rEU -o %1
gsar -s:xea -rEK -o %1
gsar -s:xef -rIU -o %1
gsar -s:xf6 -rOU -o %1
gsar -s:xf4 -rOK -o %1
gsar -s:xfc -rUU -o %1
gsar -s:xfb -rUK -o %1
gsar -s:xe8 -rEA -o %1
gsar -s:xe9 -rAE -o %1
gsar -s:x27 -rHN -o %1
gsar -s:x28 -r  -o %1
gsar -s:x29 -r  -o %1
```

Die %1 aan die einde van elke reël dui aan dat die eerste lêer wat in die DOS-opdrag gespesifiseer word, verander moet word. Die volgende opdrag word in DOS ingevoer om die veranderinge in die dataléers HAT.dat aan te bring:

```
vorentoe HAT.dat.
```

Noudat die data in 'n bruikbare vorm is, kan die PERL-program AFKwrde.pl wat net die woorde met koppeltekens uithaal, geloop word. Die program lyk soos volg:

```
#!perl -w        #Haal alle woorde met koppeltekens (QQ) uit datal\^{e}er.

open(INVOER, "<elHAT.dat");              #Maak datal\^{e}er oop.
@hat = <INVOER>;                         #Plaas data in skikking.
close(INVOER);                           #Maak datal\^{e}er toe.
open(HOOFLEER, "+>>AFKAP.dat");          #Maak hoofl\^{e}er oop. (+>> voeg by)
foreach(@hat){                           #Elke string in skikking:
        if(m/^\w+[QQ]/){                 #Vir elke woord wat QQ bevat.
                                         #Verwyder:
                s/QQ\s/ /;               #klem (QQ) aan einde;
                s/QQ,/ /;                #klem met , direk daarna;
                s/QQ1/ /;                #klem met 1 direk daarna;
                s/1\s/ /;                #'n 1 direk na woord;
                s/1,\s/ /;               #'n 1 en komma direk na woord.
                if(m/\w+[\d]/){          #Ignoreer woorde met syfer;
                    $woord=[];
                    }
                elsif(m/\w+[,\d]/){      #of met komma en syfer;
                    $woord=[];
                    }
                elsif(m/^QQ/){           #of wat met koppelteken begin;
                    $woord=[];
```

```
            }
        elsif(m/KT\s/){          #of met 'n koppelteken eindig.
            $woord=[];
            }
        else{
            $woord = '\w+';      #Plaas eerste woord in $woord.
            }
        ($AFKAP) = (m/($woord)/);      #Plaas in AFKAP.
        print HOOFLEER ("$AFKAP\n");  #Skryf na AFKAP.dat.
        }
    }
close(HOOFLEER);
```

Die program doen die volgende:

- ▷ Maak die datalêer oop, plaas die data in die skikking `@hat` en maak die datalêer weer toe.
- ▷ Maak HOOFLÊER oop sodat die woorde een vir een na die lêer `AFKAP.dat` geskryf kan word.
- ▷ Vir elke string wat 'n woord met lettergreepverdelings bevat, met ander woorde wat QQ bevat, doen die volgende:
  - Indien die klem aan die einde van die woord val (soos in *gro·tesk'*) verskyn daar QQ gevolg deur 'n spasie aan die einde van die woord. Hierdie QQ sal 'n koppelteken lewer wanneer ons terugvervang. Dit is nie 'n geldige lettergreepverdeling nie, en ons verwyder die QQ.
  - Die klemaanduiding, gevolg deur 'n komma (soos in *ba·ga·tel', ook bakatel*) lewer dieselfde probleem as hierbo. Ons verwyder dus alle QQ's gevolg deur 'n komma.
  - Wanneer daar meer as een alternatief vir dieselfde woord is, word dit in ELHAT met 'n boskrif aangedui, soos in *ag·ter·me·kaar'* [1]. (In die teksredigeerder verskyn dit as *ag·ter·me·kaar'1.*) Indien hierdie syfer op 'n klemaanduiding volg, het ons weer dieselfde probleem as voorheen. Aangesien ons elke woord net een keer in die datastel wil hê, verwyder ons die klem met 'n 1 direk daarna.
  - Indien 'n syfer direk na die woord staan (soos *aan·vaar1*) word die syfer as deel van die woord gesien en sal so in die datastel opgeneem word. Aangesien ons elke woord net een keer in die datastel wil hê, verwyder ons net die klem met 'n 1 direk daarna.
  - Om dieselfde rede as hierbo, verwyder ons 'n 1 wat direk deur 'n komma gevolg word.
  - Plaas nou woorde sonder
    - 'n syfer aan die einde;
    - 'n komma gevolg deur 'n syfer aan die einde;
    - 'n koppelteken aan die begin of einde (voor- of agtervoegsels)

    in die woordelys AFKAP en skryf dit na die lêer `AFKAP.dat`.

DOS se GSAR word nou weereens gebruik om die vreemde karakters terug te vervang. Oop reëls tussen woorde word verwyder deur die laaste opdrag in die volgende *batch file* genaamd `terug.bat`:

```
gsar -sAU -r:xe4 -o  %1
gsar -sEU -r:xeb -o  %1
gsar -sEK -r:xea -o  %1
gsar -sIU -r:xef -o  %1
gsar -sOU -r:xf6 -o  %1
gsar -sOK -r:xf4 -o  %1
gsar -sUU -r:xfc -o  %1
gsar -sUK -r:xfb -o  %1
gsar -sEA -r:xe8 -o  %1
gsar -sAE -r:xe9 -o  %1
gsar -sHN -r' -o %1
gsar -sQQ -r- -o %1
gsar -sKT -r= -o %1
gsar -s:x0d:x0a:x0d:x0a -r:x0d:x0a -o %1
```

Dit was nodig om wysigings per hand aan te bring. In ELHAT is die lettergreepverdelings by heelwat woorde, soos *afskadu*, *bangbroek*, *beskaaf*, ens. uitgelaat. Hierdie woorde is uitgesoek en koppeltekens is per hand ingeplaas.

Die lêer `AFKAP.dat` bevat 52 167 woorde met koppeltekens.

A.2.2 Kort woorde

'n Volledige lys van kort woorde wat nie lettergrepe bevat nie, soos *loop* en *staan*, is nodig om as kontrole te dien in die afrigtingsproses van die neurale netwerk. 'n PERL-program is geskryf om hierdie woorde uit ELHAT te onttrek.

Die feit dat 'n onsigbare woord in ELHAT in die geval van kort woorde presies dieselfde as die daaropvolgende woord is, word gebruik om die kort woorde te identifiseer.

Die PERL-program `KORTwrde.pl` lyk soos volg:

```
#!perl -w

open(INVOER, "<HAT.dat");                  #Maak datal\^{e}er oop.
@hat = <INVOER>;                           #Plaas data in skikking.
close(INVOER);                             #Maak datal\^{e}er toe.
open(HOOFLEER, ">lys.afk");                #Maak hoofl\^{e}er oop.
foreach(@hat){                             #Vir elke string verwyder:
              s/\d\s/ /;                   #syfer direk na woord;
              s/\d,\s/ /;                  #syfer en komma direk na woord;
              s/,\s/ /;                    #komma direk na woord.
              s/\./p/;                     #punt direk na woord.
              $woord = '\w+';                   #Eerste woord in string.
              ($lys) = (m/($woord)/);           #Plaas in lys.
              print HOOFLEER ("$lys\n");        #Skryf na lys.afk.
              }
close(HOOFLEER);
open(WRDE, "<lys.afk");
```

```
@woorde = <WRDE>;
close(WRDE);
open(KORTLEER, ">>KORT.dat");
for($index = 1; $index <= $#woorde; $index++){      #Vergelyk opeenvolgende
                                                    #woorde.
        if ($woorde[$index-1] eq $woorde[$index]){  #Indien dieselfde,
                $woord = $woorde[$index];
                ($KORT) = ($woord);                 #plaas in skikking KORT.
                print KORTLEER ("$KORT\n");         #Skryf na KORT.dat.
        }
}
close(KORTLEER);
```

Hierdie program doen die volgende:

- ▷ Vir elke woord in die skikking `@hat`, verwyder
  - syfer direk na 'n woord;
  - syfer en 'n komma direk na 'n woord;
  - 'n komma direk na 'n woord; en
  - 'n punt direk na 'n woord;
- ▷ Die eerste woord in elke string word in `lys.afk` geskryf.
- ▷ Hierdie lys van woorde word in die skikking `@woorde` geplaas.
- ▷ Beskou twee opeenvolgende woorde.
- ▷ Indien die woorde dieselfde is, plaas die tweede woord in die skikking `KORT`.
- ▷ Skryf die woorde na die lêer `KORT.dat`.

Dit was weereens nodig om wysigings per hand aan te bring. Aangesien die lettergreepverdelings by heelwat woorde uitgelaat is, het woorde wat nie kort woorde is nie verkeerdelik in die lys van kort woorde beland. Voorbeelde hiervan is *aanwend*, *afsnoer*, ens. Hierdie woorde is per hand uit `KORT.dat` verwyder en in `AFKAP.dat` geplaas.

A.2.3 Deelversamelings

Sekwensiële deelversamelings

Ten einde die neurale netwerk af te rig om afkapping te doen, het ons 'n verteenwoordigende deelversameling van die volledige datalêer `AFKAP.dat` nodig. Aangesien ons ten minste 1 000 woorde nodig het vir afrigting, is 'n PERL-program geskryf om elke 40ste woord (insluitend die eerste woord) van `AFKAP.dat` in die sublêer `WRD1.dat` te skryf.

Hierdie program lyk soos volg:

```
open(LEES, "<AFKAP.dat");
@LYS = <LEES>;
close(LEES);
open(SKRYF, "+>>WRD1.dat");
$index=0;                                    #Die eerste woord.
foreach(@LYS){
        if ($index % 40 == 0){               #Deelbaar deur 40
              $woord = $LYS[$index];
              ($WRD1) = ($woord);
              print SKRYF ("$WRD1\n");
        }
        $index++;
}
close(SKRYF);
```

Deur telkens die eerste woord uit `AFKAP.dat` te verwyder, is deellêers `WRD2.dat` ... `WRD6.dat` (wat elk uit 1 238 woorde bestaan) uit `AFKAP.dat` onttrek. Indien nodig, kan 'n verdere 34 deellêers op dié manier onttrek word.

Willekeurige deelversamelings

Die program hieronder onttrek 1 000 woorde willekeurig uit die skoongemaakte datalêer `AFKAPSK.dat`.

```
open(LEES, "<AFKAPSK.dat");
@LYS = <LEES>;
close(LEES);
open(SKRYF, "+>>RND1.dat");
for ($i=1; $i<1001; $i++)
          {
          $rnd = rand;
          $index = int($rnd*51080);
          $woord = $LYS[$index];
          ($RND1) = ($woord);
          print SKRYF ("$RND1\n");
          }
close(SKRYF);
```

Bibliografie

[1] J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research* (Cinbridge, MA: MIT Press, 1988).

[2] Barth, Kodydek, and Schonhacker, 'Reliable and Sense-Conveying Hyphenation for the German Language (SiSiSi)', (2000). `http://www.apm.tuwein.ac.at/research/SiSiSi.htm`.

[3] J. D. Berry, 'The Hyphenation of Justification', *Creativpro.com, Inc.* (2000). `http://www.creativepro.com/printerfriendly/story/8868.htm`.

[4] J. D. Berry, 'The Justification for Hyphenation', *Creativpro.com, Inc.* (2000). `http://www.creativepro.com/printerfriendly/story/8656.htm`.

[5] S. Brunak and B. Lautrop, *Neural Networks: Computers with Intuition* (World Scientific, Singapore, 1990).

[6] Circle Noetic Sevices Inc., 'Dashes Hyphenation', (1999). `http://www.mv.com/ipusers/noetic/dashes.htm`.

[7] W. Daelemans and A. V. D. Bosch, *Generalization performance of backpropagation learning on a syllabification task* (1992). `http://ilk.kub.nl/downloads/pub/antalb/twlt3-92.ps.gz`.

[8] H. B. Demuth and M. Beale, *Neural Network Toolbox: For Use with MATLAB* (The MathWorks, Inc., 1992 – 2001).

[9] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications* (Prentice-Hall, Inc., 1994).

[10] D. Fawthrop, 'Hyphenation by Algorithm of English/American and Other Languages', *Computer Hyphenation Ltd.* (1999). `http://www.hyphenologist.co.uk/descript.htm`.

[11] B. Fritzke and C. Nasahl, 'A neural network that learns to do hyphenation', in *Artificial Neural Networks, Eds.* T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas (North-Holland, Amsterdam, Netherlands, 1991), 1375–1378.

[12] Q. H. Gee, *Automatic Hyphenation of Afrikaans* (Master's thesis, University of Witwatersrand, 1987).

[13] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design* (PWS Publishing Company, 1996).

[14] S. Haykin, *Neural Networks: A Comprehensive Foundation* (Prentice-Hall, Inc., second ed., 1999).

[15] Instituut Voor Nederlandse Lexicologie, *Woordenlijst Van de Nederlandse Taal* ('s-Gravenhage: SDU, 1990).

[16] J. S. Judd, *Neural Network Design and the Complexity of Learning* (Massachusetts Institute of Technology, 1990).

[17] F. M. Liang, *Word hy-phen-a-tion by com-pu-ter* (Ph.D. thesis, Stanford University, 1983).

[18] F. M. Liang and P. Breitenlohner, 'PATGEN.WEB in Text Format', (1996). `http://ftp.cs.stanford.edu/pub/tex/unsupported/textware/patgen.web`.

[19] J. L. McClelland and D. E. Rumelhart, *Explorations in Parallel Distributed Processing* (Cambridge, MA: MIT Press, 1988).

[20] R. McIntosh and D. Fawthrop, *Hyphenation* (third ed., 1990). `http://www.hyphenologist.co.uk/book/book-ed3.htm`.

[21] S. Nizhnikov, 'Hyphenation and the Good Life', *Electric Word* (1990). `http://www.lim.nl/articles/nizhnikov.htm`.

[22] F. F. Odendal, P. C. Schoonees, et al., *Verklarende Handwoordeboek van die Afrikaanse Taal (HAT)* (Perskor-Boekdrukkery, 1983).

[23] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. (eds.), *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations* (@MIT, 1986).

[24] T. J. Sejnowski and C. R. Rosenberg, 'Parallel Networks That Learn to Pronounce English Text', *Complex Systems* Vol. 1 (1987), 145 – 168.

[25] P. Smrž and P. Sojka, 'Word hy-phen-a-tion by Neural Networks', *FI MU Report Series* (1996).

[26] P. Sojka, 'Notes on Compound Word Hyphenation in TEX', *FI MU Report Series* (1995).

[27] P. Sojka, 'Hyphenation on Demand', in *TEX Users Group Annual Meeting* (1999), 1085 – 1091.

[28] Suid-Afrikaanse Akademie vir Wetenskap en Kuns. Taalkommissie, *Afrikaanse Woordelys en Spelreëls* (Kaapstad: Tafelberg, 1991).

[29] TALO, 'The Hyphenation Rules for the European Languages', (1997). `http://www.xs4all.nl/\~talo/talo/hrules.htm`.

[30] TEX Users Group, 'Just What is TEX', (2002). `http://www.tug.org/whatis.html`.