Faculteit
Ingenieurswetenschappen

Departement
Elektrotechniek – ESAT

KATHOLIEKE
UNIVERSITEIT
LEUVEN

Zelflerende Spraakherkenning:

akoestische eenheden & woordmodellen

Eindwerk voorgedragen tot het behalen van het diploma van Burgerlijk Elektrotechnisch ingenieur, optie multimedia & signaalverwerking

Joost De Tollenaere

Promotor:
Prof. dr. ir. Hugo Van hamme
Dagelijkse begeleiding:
dr. ir. Kris Demuynck
ir. Joris Driesen

2007 – 2008

Voorwoord

Aan het begin van deze thesis wil ik graag een dankwoord richten aan al degenen die mij op één of andere manier hebben geholpen bij het maken van dit eindwerk. Sommigen van hen wil ik hier in het bijzonder bedanken.

In de eerste plaats dank ik mijn promotor prof. Hugo Van hamme. Zijn enthousiaste uitleg op de thesisbeurs vorig jaar heeft er zeker toe bijgedragen dat ik voor dit onderwerp heb gekozen. Het werd een boeiende kennismaking met de wereld van de spraakherkenning die door hem in goede banen werd geleid.

Mijn begeleiders Kris Demuynck en Joris Driesen wil ik graag bedanken voor de uitstekende begeleiding. Joris dank ik vooral voor zijn hulp op het einde, wanneer de deadline van het eindwerk dichterbij kwam. Kris dank ik voor het vele werk dat hij in dit project heeft gestoken en voor de antwoorden op al mijn vragen.

Ook mijn familie wil ik hierbij zeker niet vergeten. Mijn ouders dank ik voor de steun die ze het hele jaar gegeven hebben. Mijn broer en mijn zus, die zelfs tijdens de drukke verhuis nog tijd vond om mij te helpen, verdienen zeker een dankjewel.

Tenslotte dank ik nog alle vrienden voor hun inbreng en steun.

Abstract

Het is een heel opmerkelijk feit dat peuters in staat zijn om complexe klankpatronen van de taal te ontdekken en te leren, dit zowel op niveau van fonemen als van woorden. De huidige generatie van automatische spraakherkenners, die kennis gebruiken die door experten op gebied van spraakherkenning is ingevoerd, zijn nog steeds niet in staat een dergelijke performantie te behalen. Het doel van deze thesis is na te gaan of het mogelijk is voor computers om, net zoals kinderen, geheel ongesuperviseerd de akoestische klankpatronen te ontdekken in een taal. Wanneer deze gevonden zijn wordt hierop verder gebouwd om woorden te leren en te herkennen.

Het vinden van zo'n patronen vereist een clustering. Om een geschikte clustertechniek te vinden werden twee groepen van algoritmes bestudeerd: niet-negatieve matrixfactorisatie en spectrale clustering. Dit zijn technieken die de laatste jaren populair geworden zijn in heel wat domeinen van unsupervised learning. Toch hebben ze beide het probleem dat er slechts convergentie is naar een lokaal optimum. Bij NMF uit dit zich in een sterke afhankelijkheid tussen de performantie en de initialisatie. Er wordt een initialisatie voorgesteld die gebruikt maakt van een tussenresultaat van spectrale clustering als initialisatie voor NMF. De clustertechnieken worden uitgetest en vergeleken op een artificieel probleem. De voorgestelde techniek blijkt de beste performantie te hebben van de onderzochte technieken.

Deze techniek wordt dan aangewend om het eerste deel van de doelstelling te realiseren, nl. het vinden van akoestische eenheden in spraakuitingen. Er wordt een clustering uitgevoerd op de database Wall Street Journal (WSJ). Dit resulteert in een set van akoestische eenheden, die sterke banden vertonen met de fonemen uit de Engelse taal. Toch is dit alles behalve een 1-op-1 mapping. Verscheidene akoestische eenheden mappen hetzelfde foneem en omgekeerd. Via een studie van de spectrogrammen van enkele woorden, blijkt dat de problemen voortkomen uit het sterke gelijkenissen tussen fonemen binnen dezelfde categorie. Akoestische eenheden volgen wel goed de karakteristieken van het geluidssignaal. Dit zorgt ervoor dat akoestische eenheden als bouwblokken kunnen worden aangewend voor spraakherkenning op hoger niveau.

In het derde deel worden de akoestische eenheden gebruikt als basis voor woordherkenning. Er wordt een akoestisch model getraind via de segmentatie in akoestische eenheden. De uitingen in een trainingsset worden compact voorgesteld in een matrix. Aan deze matrix worden tags toegevoegd die aangeven welke sleutelwoorden voorkomen in de uiting. Via een niet-negatieve matrixfactorisatie worden woordmodellen geleerd voor elk van deze sleutelwoorden. Deze woordmodellen worden in een volgende fase gebruikt om te detecteren of het sleutelwoord voorkomt in een uiting. Ook hiervoor wordt NMF gebruikt. De eerste resultaten van deze experimenten blijken nog heel slecht zijn. Dit deel van het eindwerk kan in de toekomst zeker nog verbeterd worden. De resultaten tonen aan dat het veranderen van parameters een duidelijk

effect heeft op de performantie van de woordherkenning.

Inhoudsopgave

Lijst van symbolen

V	matrix die ontbonden wordt via niet-negative matrixfactorisatie
W	matrix met nieuwe basisvectoren in NMF
H	indicatormatrix in NMF
D	diagonaalmatrix
P	probabiliteitsmatrix
S	similariteitsmatrix
M	dimensie van datavectoren
N	aantal datavectoren
T	aantal frames in de spraakdata
R	aantal clusters of aantal akoestische eenheden
U	aantal uitingen in de spraakdata
S	aantal sleutelwoorden in de spraakdata
A_t	gesegmenteerde akoestische eenheid in het frame met index t
P_t	gesegmenteerd foneem in het frame met index t
O_t	gegenereerde featurevector in het frame met index t
knn(i)	de verzameling van de k dichtste buren van een punt i
$\text{diag}_j\, a_j$	de diagonaalmatrix met diagonaalelement j gelijk aan a_j

Lijst van figuren

Lijst van tabellen

Hoofdstuk 1

Inleiding

Op zeer jonge leeftijd leren kinderen de levensnoodzakelijke vaardigheid om te communiceren via taal. Hoewel gesproken taal niets meer is dan een akoestisch signaal, zijn peuters in staat om hierin complexe klankpatronen van de taal te ontdekken. Ook bouwen kinderen op zeer jonge leeftijd een grote woordenschat op, enkel en alleen door een taal te horen en deze vaak herhaalde combinaties van klanken te onderscheiden en te leren.

Hiertegenover staat dat de huidige generatie van automatische spraakherkenners, die kennis gebruiken die door experten op gebied van spraakherkenning is ingevoerd, nog steeds niet in staat zijn om een performantie te behalen gelijk aan die van peuters. De spraakherkenning heeft in dit gebied nog veel groeiruimte en er wordt bijgevolg ook veel onderzoek naar verricht.

In deze thesis wordt nagegaan of ook computers in staat zijn om klankpatronen in spraak te herkennen en op basis van deze klankpatronen woorden te herkennen. De beperking die wordt opgelegd is dat dit hele leerproces ongesuperviseerd is, de computer moet zonder enige inbreng van kennis van buitenaf de klankpatronen zoeken. Daarbij mag de computer enkel gebruik maken van grote hoeveelheden spraakdata en natuurlijk de nodige hoeveelheid rekenkracht. In hoofdstuk 3 wordt op zoek gegaan naar de akoestische eenheden die de computer herkent als bouwstenen voor spraakuitingen. In hoofdstuk 4 worden deze gevonden akoestische eenheden aangewend om er spraakherkenningsexperimenten mee uit te voeren.

De technieken die in deze problemen gebruikt worden, zijn niet-negatieve matrixfactorisatie (NMF) en spectraal clusteren. Beide technieken kunnen worden aangewend om een clusteringsprobleem op te lossen. NMF wordt daarnaast ook gebruikt om aan woordherkenning te doen. Deze technieken worden uitgebreid besproken in hoofdstuk 2.

Hoofdstuk 2

Clustertechnieken

Het eerste deel in dit eindwerk behandelt het clusteren van features van spraakdata. Bepaalde clustertechnieken werden onderzocht a.d.h.v artificiële testdata om de beste methode te kiezen voor het vervolg van dit eindwerk. Deze experimenten worden in dit hoofdstuk besproken. Eerst wordt een overzicht gegeven van de onderzochte clustertechnieken, vervolgens wordt dieper ingegaan op de uiteindelijk gekozen methode, en tenslotte worden de resultaten voorgesteld van de experimenten op de testdata. Het is belangrijk eerst een antwoord te formuleren op de vraag: wat is clusteren?

> Clusteren is het classificeren van datapunten in groepen, waarbij de punten binnen een groep zo groot mogelijke gelijkenissen vertonen, en punten van verschillende groepen zo weinig mogelijk gelijkenissen vertonen.

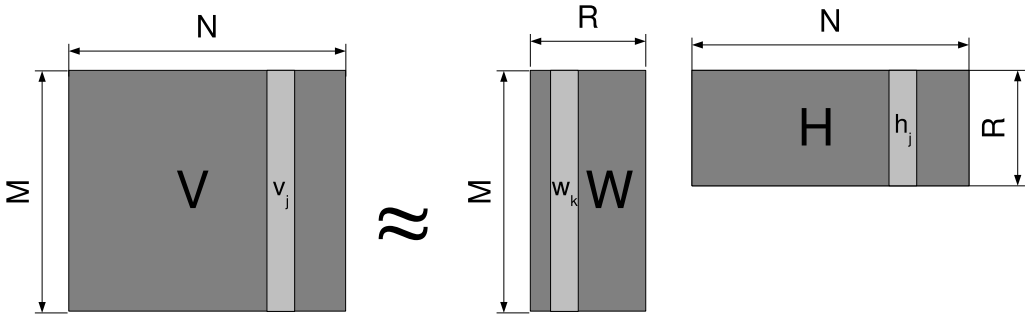
2.1 Niet-negative matrixfactorisatie

Niet-negatieve matrixfactorisatie (NMF) is een techniek waarbij een niet-negatieve matrix benaderd wordt door het product van 2 matrices met een kleinere dimensie. De opgelegde beperking is dat de matrices W en H enkel niet-negatieve elementen bevatten.

$$V \approx W \cdot H \tag{2.1}$$

De factorisatie wordt gebruikt om een decompositie uit te voeren van multivariate data. Stel dat er N datavectoren zijn van dimensie M. Deze vectoren worden in de kolommen geplaatst van de matrix V. Deze matrix heeft als grootte $M \times N$. V wordt benaderd via NMF en dit resulteert in de matrices W en H met respectievelijke dimensies $M \times R$ en $R \times N$. De matrix W is de voorstelling van de nieuwe basis van de gereduceerde R-dimensionale ruimte. De vectoren van V (de originele datavectoren) worden voorgesteld in deze basis: ze worden geschreven als een lineaire combinatie van de basisvectoren: $\vec{v}_j = \sum_k h_{kj}\vec{w}_k$. De coëfficiënten h_{kj} zijn de elementen in de overeenkomstige kolom van H. In figuur 2.1 is dit schematisch voorgesteld.

Omdat beide matrices W en H enkel niet-negatieve elementen bevatten, is deze lineaire combinatie een som van delen die elkaar niet kunnen opheffen. De basisvectoren in W zijn de delen waaruit de data is opgebouwd. Lee & Seung [1] beschreven dat dit vaak tot gevolg



Figuur 2.1: Schematische voorstelling van de dimensies van de NMF-matrices

heeft dat H een spaarse structuur heeft: de kolommen bevatten de indicatoren die aangeven uit welke delen de overeenkomstige datavectoren in V zijn opgebouwd. H wordt daarom ook wel de indicatormatrix genoemd.

Wanneer NMF een dataset nauwkeurig benadert d.m.v. een beperkt aantal 'delen', betekent dit dat het algoritme een latente structuur heeft gevonden in de data. NMF is daarom een techniek die gebruikt wordt om verborgen structuren in data op te sporen en aan het licht te te leggen. Dit heeft heel wat toepassingen en dit maakt van NMF een populair ongesuperviseerd leeralgoritme.

2.1.1 Knn-matrix

De niet-negatieve matrixfactorisatie kan gebruikt worden om te clusteren. Stel dat N datapunten $\vec{x_i}$ moeten worden geclusterd en dat deze zich bevinden in een M-dimensionale dataruimte. Er moet een matrix V gevonden worden die ervoor zorgt dat de bekomen H-matrix na NMF de indicaties bevat van de clusters. De verticale dimensie van deze matrix is N: elke datapunt correspondeert met een kolom in V. De horizontale dimensie M moet groter zijn dan R, het gewenste aantal clusters. Indien deze M kleiner is dan R, dan wordt immers een triviale oplossing bekomen. In H zijn de eerste M rijen gelijk aan de rijen van V. In de matrix W vormen de eerste M kolommen de eenheidsmatrix, en de overblijvende $R - M$ kolommen bevatten enkel nullen. De 'delen' die hier worden gevonden zijn precies de M richtingen in de dataruimte. Dit biedt dus geen nieuwe informatie.

In de hier geteste clustertechnieken wordt de matrix V berekend op basis van de nabijheid van punten tot buren. Dit heeft het voordeel dat het resultaat van de clustering veel minder zal afhangen van de gebruikte afstandsmaat tussen de datapunten. De berekening gaat als volgt in zijn werk. Van elk punt worden de k dichtste buren berekend, dit heet *k nearest neighbors* of kNN. Elke relatie van een punt tot zijn buur krijgt ook een similariteitswaarde, dit is een maat voor de nabijheid van het punt tot die buur. Dit alles wordt voorgesteld in V, die de

knn-matrix genoemd wordt:

$$V_{ij} = \begin{cases} s_{ij} & \text{als } i \in \text{knn}(j) \\ 0 & \text{als } i \notin \text{knn}(j) \end{cases} \tag{2.2}$$

Hier is s_{ij} de similariteitsmaat tussen punt i en punt j. V is een vierkante matrix van dimensie $N \times N$, met N het aantal datapunten. Deze matrix is niet noodzakelijk symmetrisch: een punt A kan één van de k dichtste buren zijn van punt B, terwijl punt B niet behoort tot de k dichtste buren van punt A. De similariteitsmaten zijn wel symmetrisch:

$$s_{ij} = s_{ji} \tag{2.3}$$

De matrix kan wel symmetrisch gemaakt worden indien nodig, dit heet dan *mutual k nearest neighbors*. De datapunten, die zich oorspronkelijk bevonden in de M-dimensionale featureruimte, zijn nu getransformeerd naar de N-dimensionale kNN-ruimte.

2.1.2 Algoritmes voor niet-negatieve matrixfactorisatie

Algoritmes om de NMF te berekenen zijn ontworpen om een kostfunctie te minimaliseren, deze kostfuncties vergelijken de matrix V met zijn benadering $\tilde{V} = W.H$. In praktijk komen twee kostfuncties voor: het divergentiecriterium (zie vergelijking 2.4) en het MSE-criterium (zie vergelijking 2.5), dat staat voor mean squared error.

$$\text{Div}(V\|\tilde{V}) = \sum_{i,j} \left(V_{ij} \log \frac{V_{ij}}{\tilde{V}_{ij}} - V_{ij} + \tilde{V}_{ij} \right) \tag{2.4}$$

$$\|V - \tilde{V}\|^2 = \sum_{i,j} (V_{ij} - \tilde{V}_{ij})^2 \tag{2.5}$$

NMF met het divergentiecriterium geeft hetzelfde resultaat als PLSA, *probabilistic latent semantic analysis*. Deze techniek start van een andere achtergrond, maar heeft dezelfde updateformules en bijgevolg ook hetzelfde resultaat als NMF met divergentiecriterium. Deze gelijkenis heeft interessante gevolgen: de bekomen clustering is nu statistisch onderbouwd.

Zowel bij NMF met divergentiecriterium als bij NMF met MSE-criterium vinden de algoritmes geen globaal minimum, maar convergeren ze naar een lokaal minimum. Verschillende initialisaties kunnen dus verschillende resultaten opleveren. Het is daarom belangrijk een goede initialisatie te vinden voor de algoritmes. Ook het invoeren van beperkingen (*constraints*) kan de oplossing beïnvloeden, zodat een beter lokaal minimum gevonden wordt.

- Een eerste constraint is de spaarsheid van de matrix H. Aangezien deze matrix de indicatievectoren bevat, zal deze bij een goede clustering in elke kolom slechts één of enkele elementen bevatten die niet nul zijn, omdat het punt slechts bij één of enkele clusters behoort. Bij een goed resultaat zal de matrix H dus spaars zijn.
- Een andere mogelijke constraint is de orthogonaliteit. Een orthogonale matrix die enkel niet-negatieve elementen bevat, zal in elke kolom slechts één element hebben dat niet gelijk is aan nul. Als de H-matrix orthogonaal is, zal er dus voor elke indicatievector slechts één element niet-nul zijn, en zal elk punt aan één cluster worden toegekend. Het

resultaat is dus een harde clustering. De kostfunctie van de NMF met MSE-criterium en orthogonaliteitsconstraint is:

$$\min_{W\geq 0, H\geq 0} \|V - W \cdot H\|^2, \text{ s.t. } H \cdot H^T = I \tag{2.6}$$

In [2] wordt aangetoond dat dit equivalent is met *k-means clustering*. NMF met MSE-criterium is equivalent met k-means clustering, waarbij de orthogonaliteitsconstraint gerelaxeerd is. NMF is m.a.w. een zachte clusteringsvariant van k-means clustering.

- Een derde mogelijke beperking is symmetrie. Indien de matrix V symmetrisch is, kan deze symmetrie behouden worden in de factorisatie; dit is het opzet van *symmetrische NMF*. De kostfunctie voor symmetrische NMF met MSE-criterium is:

$$\min_{H\geq 0} \|V - H^T H\|^2 \tag{2.7}$$

Deze symmetrische factorisatie heeft enkele interessante eigenschappen. In [3] wordt aangetoond dat symmetrische NMF equivalent is aan Kernel K-means clustering en Laplacian-based Spectral clustering. Dit wordt verder beschreven in volgende paragraaf.

2.2 Spectrale clustering

Spectrale clustering is een techniek die de laatste jaren heel wat aan populariteit heeft ingewonnen. 'Spectraal' betreft het werken met de eigenwaardenontbinding van een matrix. Het zal blijken dat spectrale clustering heel wat voordelen heeft ten opzichte van klassieke clustertechnieken zoals K-means clustering.

Algoritmes voor spectrale clustering zijn opgebouwd uit vier stappen:

1. Opstellen van de similariteitsmatrix
2. Berekening van de laplaciaan uit de similariteitsmatrix
3. Eigenwaardenontbinding van de laplaciaan
4. Clustering van een of meerdere eigenvectoren met klassieke clustering

Voor elk van de stappen bestaan er verscheidene varianten, die aanleiding geven tot de verschillende spectrale clusteralgoritmes. De vier stappen worden hieronder verder verduidelijkt.

2.2.1 Similariteitsmatrix

Het idee achter spectrale clustering is gebaseerd op grafe-snedes (*graph cuts*). Eerst wordt de similariteitsgrafe $G = (V, E)$ opgesteld die de gelijkenissen tussen de datapunten aangeeft: de knopen $v \in V$ van de grafe stellen de datapunten voor, de bogen $e \in E$ geven de gelijkenissen weer tussen de verschillende datapunten onderling. Om die gelijkenissen te meten is er een similariteitsmaat nodig: een positief getal dat aangeeft hoe sterk de punten op elkaar lijken. Als de similariteit van twee punten nul is, dan zijn de overeenkomstige knopen in de grafe niet verbonden. De similariteit is meestal gebaseerd op de nabijheid van de punten en zal in dat geval berekend worden via de afstand tussen de punten. Een veelgebruikte similariteitsmaat is de volgende:

$$s_{ij} = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{\sigma^2}\right) \tag{2.8}$$

Deze geeft de similariteit aan tussen de punten $\vec{x}_i$ en $\vec{x}_j$. De σ is een parameter die per probleem moet worden geschat of geoptimaliseerd. De invloed van deze parameter is dat, hoe groter de σ, hoe kleiner de inbreng zal zijn van punten die op een grote afstand van elkaar liggen. Vergelijking 2.8 heet de radiale basis-functie of gaussiaanse kernelfunctie.

Bij spectrale clustering wordt deze grafe voorgesteld met behulp van een matrix. Deze *similariteitsmatrix* W heeft als dimensie $N \times N$, waarbij N het aantal datapunten is. De elementen zijn de gewichten tussen de knopen: $W_{ij} = s_{ij}$ als knoop i verbonden is met knoop j, $W_{ij} = 0$ als de respectievelijke knopen niet verbonden zijn.

Aangezien de afstand tussen twee punten nooit oneindig is, zal deze similariteitsmaat nooit nul zijn. Bijgevolg zijn alle knopen in de grafe onderling verbonden. Dit heet een volledig verbonden grafe (fully connected graph). Het gevolg hiervan is dat de similariteitsmatrix volledig dens is, wat de rekentijd in de volgende stap sterk zal vergroten. De oplossing hiervoor is het aantal bogen in de grafe te verminderen, dit kan op twee manieren. De eerste mogelijkheid is een drempelwaarde in te stellen voor de similariteitsmaat: enkel indien de similariteit tussen twee punten groter is dan deze threshold, zijn de overeenkomstige knopen in het netwerk verbonden. Het resultaat is dan een *ϵ-neighborhood graph*. In de matrix W vertaalt dit zich als volgt:

$$W_{ij} = \begin{cases} s_{ij} & \text{voor } s_{ij} \geq \epsilon \\ 0 & \text{voor } s_{ij} < \epsilon \end{cases} \tag{2.9}$$

Een tweede methode is de *k nearest neighbors graph*. Voor elk punt i worden de k dichtste buren gezocht - de k punten met grootste similariteit t.o.v punt i - en enkel deze k dichtste buren blijven verbonden met knoop i in de grafe. Dit wordt herhaald voor elk punt in de dataset. In de similariteitsmatrix komt dit op hetvolgende neer:

$$W_{ij} = \begin{cases} s_{ij} & \text{als } i \in \text{knn}(j) \\ 0 & \text{als } i \notin \text{knn}(j) \end{cases} \tag{2.10}$$

Merk op dat de grafe nu gericht is: er kan een boog gaan van knoop A naar knoop B, terwijl er geen boog is in omgekeerde richting. Dit kan als punt B meer dan k buren heeft die dichter liggen dan punt A, terwijl bij punt A de punten verder weg liggen en punt B wel bij de k dichtste buren behoort van punt A. De gerichte grafe heeft tot gevolg dat de similariteitsmatrix W asymmetrisch kan zijn.

Elke knoop van de grafe heeft ook een graad, nl. de graad van knoop i is

$$d_i = \sum_{j=1}^{N} W_{ij} \tag{2.11}$$

De graad-matrix D bevat deze graden als elementen van zijn diagonaal:

$$D_{ij} = \begin{cases} d_i & \text{als } i = j \\ 0 & \text{als } i \neq j \end{cases} \tag{2.12}$$

2.2.2 Graph Cut

Nu de similariteitsgrafe en -matrix gedefinieerd zijn, kan het probleem van de clustering, dat is aangegeven in het begin van dit hoofdstuk, opnieuw omschreven worden, deze keer in termen van de similariteitsgrafe.

Clusteren is het partitioneren van de grafe, waarbij bepaalde bogen zullen doorsneden worden, en heeft als doelstellingen:

1. De gewichten van de bogen binnen een partitie moeten zo groot mogelijk zijn
2. De gewichten van de doorsneden bogen (dit zijn de similariteiten tussen punten van een verschillende cluster), moeten zo klein mogelijk zijn.

De snede of cut tussen twee disjuncte deelverzamelingen $A, B \subset V$, met V de verzameling van knopen in de grafe, wordt als volgt gedefinieerd:

$$\text{cut}(A, B) = \sum_{i \in A} \sum_{j \in B} W_{ij} \tag{2.13}$$

Wanneer de verzameling V in r partities $A_1, \ldots, A_r$ is onderverdeeld, dan is de totale snede van deze partitie:

$$\text{cut}(A_1, \ldots, A_r) = \sum_{i=1}^{r} \text{cut}(A_i, \bar{A}_i) \tag{2.14}$$

$\bar{A}_i$ is hier het complement van de deelverzameling A_i over V. Het minimaliseren van dit criterium zorgt ervoor dat de tweede doelstelling van het clusteringsprobleem bereikt wordt. Dit heet het *min-cut* probleem. Er bestaan verscheidene algoritmes die dit probleem oplossen. Deze algoritmes geven als output partities van de grafe, maar vaak zal het voorkomen dat er partities zijn die slechts een heel klein aantal afgelegen punten bevatten. Het optimalisatiecriterium moet zodoende ook de eerste doelstelling omvatten om gebalanceerde clusters te hebben als resultaat. Bij *Normalized Cut* (Ncut) is het optimalisatiecriterium:

$$\text{NCut}(A_1, \ldots, A_r) = \sum_{i=1}^{r} \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)} \tag{2.15}$$

$$\text{vol}(A_i) = \sum_{k \in A_i} \sum_{j \in V} W_{kj} \tag{2.16}$$

Het clusterprobleem is nu herleid tot het minimaliseren van de Normalized Cut.

2.2.3 Laplaciaan & eigenwaardenontbinding

De oplossingen van deze grafesnede-problemen wordt berekend met een eigenwaardenontbinding. Om dit in te zien wordt een standaardprobleem hier uitgewerkt. Het minimum-cut probleem met 2 clusters is de basis van deze optimalisatieproblemen. De uitwerking gaat als volgt. Er zijn N datapunten in de dataset, die ofwel bij cluster A horen, ofwel bij cluster B. Merk op dat het hier een harde clustering betreft. Voor elk datapunt i geeft q_i weer bij welke cluster het punt hoort, via deze indicatie:

$$q_i = \begin{cases} 1 & \text{als } i \in clusterA \\ -1 & \text{als } i \in clusterB \end{cases} \tag{2.17}$$

Deze vormen de elementen van de vector q. Het algoritme gaat op zoek naar een vector q die het min-cut criterium minimaliseert:

$$\min_q J(q) \qquad \text{met } J(q) = cut(A, B) \tag{2.18}$$

De doelfunctie wordt geschreven in functie van de gewichten W_{ij}, de graden d_i en de indicatorvector q.

$$\begin{aligned}
J(q) &= \sum_{i\in A}\sum_{j\in B} W_{ij} \\
&= \sum_i \sum_j W_{ij}\frac{(q_i - q_j)^2}{4} \\
&= \left[\frac{1}{4}\sum_i q_i^2 \sum_j W_{ij}\right] + \left[\frac{1}{4}\sum_j q_j^2 \sum_i W_{ij}\right] - \left[\frac{1}{2}\sum_i\sum_j q_i q_j\right] \text{ met } W_{ij} = W_{ji} \\
&= \frac{1}{2}\sum_i q_i^2 d_i - \frac{1}{2}\sum_i\sum_j q_i q_j W_{ij} \\
&= \frac{1}{2}\sum_i\sum_j q_i(d_i\delta_{ij} - W_{ij})q_j \quad \text{met } \delta_{ij} \text{ de Kronecker-delta} \\
&= \frac{1}{2}q^T(D-W)q
\end{aligned} \tag{2.19}$$

Indien de discrete waarden in q gerelaxeerd worden naar continue waarden, is de minimum waarde van J(q) gelijk aan de kleinste eigenwaarde van de matrix $D - W$. De vector q is de bijhorende eigenvector. De matrix $L = D - W$ wordt de laplaciaan genoemd. Deze heeft als kleinste eigenwaarde nul, en de bijhorende eigenvector is $q = (1, \ldots, 1)^T$. Dit is inderdaad de kleinst mogelijke grafesnede: alle clusters worden toegekend aan cluster A, en dus wordt geen enkele boog doorsneden. De totale snede is dan inderdaad gelijk aan de eigenwaarde, nl. nul Aangezien deze triviale oplossing geen waarde heeft, wordt de oplossing van het min-cut probleem gegeven door de eigenvector horend bij de op één na kleinste eigenwaarde. De toewijzing aan de clusters gebeurt op basis van deze eigenvector. De doelfunctie van het min-cut probleem is ongevoelig voor het optellen van een constante c bij elk element.

$$J(q+c) = \sum_i\sum_j W_{ij}\frac{\Big((q_i - c) - (q_j - c)\Big)^2}{4} = J(q) \tag{2.20}$$

De toewijzing aan de clusters kan dus niet uitgevoerd worden via een thresholdbewerking, maar maakt gebruik van de mediaan:

$$\begin{aligned}
A &= \{i \mid q_i \geq \text{mediaan}(q)\ \} \\
B &= \{i \mid q_i < \text{mediaan}(q)\ \}
\end{aligned} \tag{2.21}$$

Analoog kan ook voor Normalized Cut aangetoond worden dat de oplossing van deze problemen gegeven wordt via het oplossen van een eigenwaardenprobleem. Shi & Malik [4] bewezen dat het minimaliseren van Ncut equivalent is aan:

$$\min_x \text{Ncut}(x) = \min_y \frac{y^T(D-W)y}{y^T D y} \text{ s.t. } yD\mathbf{1} \tag{2.22}$$

met $\forall i : x_i \in \{-1, 1\}$ een indicatorfunctie voor de twee partities. Dit probleem is NP-compleet. Analoog met het voorbeeld van MinCut, wordt ook hier voorwaarde gerelaxeerd

dat indicatorvector enkel discrete waarden mag bevatten. De oplossing van het Ncut-probleem wordt dan gegeven door volgend veralgemeend eigenwaardenprobleem op te lossen.

$$(D - W)y = \lambda Dy \tag{2.23}$$

Ook hier geeft de eigenvector horend bij de op één na kleinste eigenwaarde de oplossing van dit probleem.

2.2.4 K-way spectraal clustering

Het besproken algoritme kan enkel gebruikt worden indien twee clusters gezocht worden. Als er meer clusters gezocht worden, spreekt men van k-way clustering. Shi & Malik hebben het Ncut algoritme uitgebreid voor k-way clustering.

Algoritme 1 (Shi & Malik) *NCut*

1. *Zoek de veralgemeende eigenvectoren van* $(D - W)y = \lambda Dy$
2. *Neem de k kleinste eigenvectoren* y_i*, start bij de op één na kleinste*
3. *Vorm de matrix X van de k kolomvectoren:* $X = [y_1 y_2 \dots y_k] \in \mathbb{R}^{n \times k}$
4. *Normaliseer de rijen van X:* $X_{ij} = \frac{X_{ij}}{(\sum_j X_{ij}^2)^{1/2}}$
5. *De rijen van X vormen nu N punten in de k-dimensionale ruimte. Cluster deze punten met k-means.*

Ng, Jordan en Weiss [5] stelden variant van dit algoritme voor:

Algoritme 2 (Ng, Jordan & Weiss) .

1. *Vorm de matrix* $A = W$*, maar stel* $A_{ii} = 0$
2. *Construeer de laplaciaan genormaliseerd met zijn kolom en rij-graad:* $L_{ij} = \frac{A_{ij}}{\sqrt{D_{ii} Djj}}$
3. *Neem de k grootste eigenvectoren van L:* y_i
4. *Vorm de matrix X van de k kolomvectoren:* $X = [y_1 y_2 \dots y_k] \in \mathbb{R}^{n \times k}$
5. *Normaliseer de rijen van X:* $X_{ij} = \frac{X_{ij}}{(\sum_j X_{ij}^2)^{1/2}}$
6. *De rijen van X vormen nu N punten in de k-dimensionale ruimte. Cluster deze punten met k-means.*

2.3 Gekozen clustermethode

De methode die uiteindelijk gebruikt wordt in de spraakherkenningsexperimenten steunt zowel op niet-negatieve matrixfactorisatie als op spectrale clustering. Aangezien dit wat extra uitleg behoeft, is het volgende onderdeel van dit hoofdstuk aan deze methode gewijd. De verschillende stappen in de methode worden achtereenvolgens behandeld.

2.3.1 Similariteitsmatrix

Deze methode begint vrij analoog aan NMF en spectrale clustering. Net zoals bij die twee technieken wordt ook hier een similariteitsmatrix opgesteld. Omdat er toch een aantal verschillen zijn met de similariteitsmatrix van spectrale clustering, wordt de volledige opbouw van de similariteitsmatrix hieronder beschreven.

De dataverzameling bestaat uit N punten $\{\vec{x}_i \mid i = 1 \dots N\}$ in een M-dimensionale ruimte. Vooreerst wordt het kwadraat van de Euclidische afstand tussen elk van deze punten berekend en voorgesteld in de matrix A met grootte $N \times N$.

$$A_{ij} = \|\vec{x}_i - \vec{x}_j\|_2^2 \ = (x_i - x_j)^T (x_i - x_j) \tag{2.24}$$

Nadien wordt elke kolom van A met een verschillende factor geschaald. Dit komt neer op een rechtse vermenigvuldiging van A met een diagonaalmatrix D^1.

$$\begin{aligned} &B = A \cdot D^1 \\ &\text{met } D^1 = \operatorname*{diag}_j(\frac{1}{\sigma_j}) \\ &\text{en } \sigma_j = \frac{A_{\alpha(j),j}}{\ln(0.01)} \end{aligned} \tag{2.25}$$

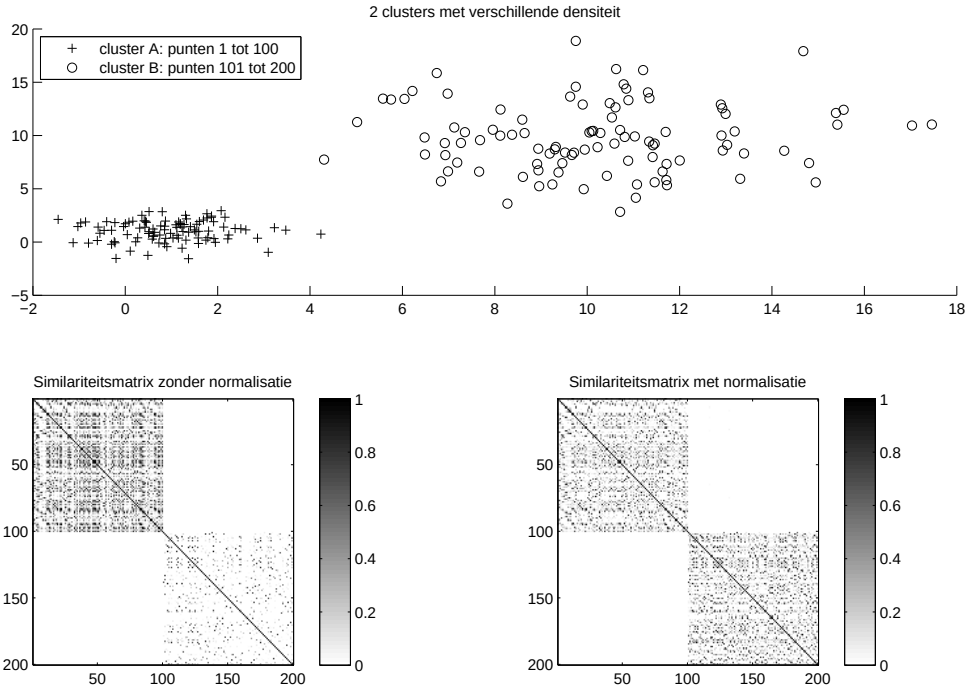
Hierbij is $\alpha(j)$ de index van het k-de kleinste element in de verzameling $\{A_{ij} \mid i = 1 \dots N\}$ voor een gegeven j. De similariteitsmatrix is de matrix waarbij elk element de exponentiële is van het overeenkomstige element in B. Net zoals bij de similariteitsmatrix van spectrale clustering, wordt deze matrix spaars gemaakt, door enkel de k dichtste buren over te houden van elk punt.

$$V_{ij} = \begin{cases} \exp(B_{ij}) & \text{als } i \in \text{knn}(j) \\ 0 & \text{als } i \notin \text{knn}(j) \end{cases} \tag{2.26}$$

Nu wordt het resultaat van de normalisatie D^1 duidelijk: het punt op de k-de kleinste afstand van elk punt heeft nu dezelfde kleine similariteitswaarde van 0.01, of in formulevorm: $\forall j : V_{\alpha(j),j} = 0.01$. Punten die verder liggen hebben een similariteit 0, aangezien enkel de k dichtste buren worden overgehouden. De waarde 0.01 geldt dus ook als een drempelwaarde voor de similariteiten. De similariteitsgrafe die overeenkomt met de genormaliseerde matrix is zowel een ϵ-neighborhood graph als een k-nearest neighbors graph. Het doel van de normalisatie is het wegwerken van verschillen in spreidingsgraad tussen de verschillende clusters. Wanneer clusters met verschillende graad van spreiding voorkomen, zal dit in een ongenormaliseerde similariteitsmatrix zichtbaar zijn. Punten in een cluster met grote spreiding hebben in een ongenormaliseerde similariteitsmatrix veel lagere waardes dan punten in een cluster met kleine spreiding. De normalisatie D^1 zal deze verschillen die veroorzaakt worden door het verschil in spreidingsgraad wegwerken. Op figuur 2.2 is het effect van deze normalisatie visueel voorgesteld.

2.3.2 Eigenwaardenontbinding

De volgende stap in het algoritme verloopt gelijkaardig aan het algoritme van Ng, Jordan en Weiss. Het verschil is dat hier niet de similariteitsmatrix gebruikt wordt zoals die beschreven



Figuur 2.2: Effect van de normalisatie D^1 op de similariteitsmatrix. Bovenaan de voorstelling van cluster A met kleine spreiding en cluster B met grote spreiding. Onderaan links de similariteitsmatrix zonder normalisatie. Onderaan rechts de similariteitsmatrix met normalisatie.

staat in 2.2.1, maar de similariteitsmatrix die in het vorige onderdeel opgebouwd is. Deze vertonen grote gelijkenissen, enkel de normalisatie D^1 is toegevoegd. De similariteitsmatrix is door deze normalisatie en door de knn-selectie niet meer symmetrisch. Voor de spectrale clusteringsstap uit te voeren is het echter beter dat de similariteitsmatrix symmetrisch is: de theorie waarop spectraal clusteren gebouwd is gaat hier immers van uit. Om de matrix V terug symmetrisch te maken, kunnen verrschillende bewerkingen worden uitgevoerd:

$$W_{min} = \min(V, V^T) \tag{2.27}$$

$$W_{max} = \max(V, V^T) \tag{2.28}$$

$$W_{or} = \frac{V + V^T}{2} \tag{2.29}$$

$$W_{and} = \sqrt{V. * V^T} \tag{2.30}$$

Merk op dat W_{and} en W_{min} bij dit algoritme niet hetzelfde resultaat geven, aangezien de matrix waarop de knn-selectie werd uitgevoerd ook al niet meer symmetrisch was. Bij de

spectrale clustering geven deze twee bewerkingen wel hetzelfde resultaat. W_{min} en W_{and} zullen verscheidene connecties tussen knopen in de grafe verwijderen, W_{or} en W_{max} zullen nieuwe connecties bijmaken. De matrix W die zo bekomen wordt is nu geschikt als input voor het algoritme 2. Van dit algoritme worden echter enkel de eerste 3 stappen uitgevoerd: de eigenwaardenontbinding van de Laplaciaan wordt berekend en de k grootste eigenvectoren worden geselecteerd. De k-means clustering, dat het sluitstuk is van het spectraal clusteren wordt hier niet uitgevoerd.

Het eigenwaardenprobleem van Ng, Jordan en Weiss dat hier ook gebruikt wordt kan als volgt uitgewerkt worden:

$$L \cdot Y = Y \cdot \Lambda \tag{2.31}$$

$$D^{-1/2} A D^{-1/2} \cdot Y = Y \cdot \Lambda \tag{2.32}$$

Hierbij is D de graad-matrix zoals gedefinieerd in 2.12. Stel $U = D^{-1/2} \cdot Y$ of anders geschreven $Y = D^{1/2} \cdot U$, dan wordt dit

$$D^{-1/2} A D^{-1/2} \cdot D^{1/2} U = D^{1/2} U \cdot \Lambda \tag{2.33}$$

$$A \cdot U = D \cdot U \cdot \Lambda \tag{2.34}$$

De matrix U bevat de veralgemeende eigenvectoren van de matrix A. Herinner dat deze matrix A gelijk is aan de similariteitsmatrix W waarbij de diagonaalelementen op nul worden gesteld. Deze bewerking komt in de grafenvoorstelling overeen met het verwijderen van de connectie van een knoop naar zichzelf. Het gevolg is dat verafgelegen knopen met enkel een connectie naar zichzelf nu geen aparte clusters gaan opeisen met enkel het punt zelf in de cluster.

2.3.3 Q-matrix

In dit algoritme worden de eigenvectoren niet gebruikt om deze met k-means te clusteren. In dit algoritme wordt de matrix van eigenvectoren na een transformatie de initialisatie van het NMF-algoritme. Dit is de vernieuwing, die spectraal clusteren en niet-negatieve matrixfactorisatie verenigt.

Y bevat de eigenvectoren van een symmetrische matrix en is dus orthonormaal. Aangezien enkel de eigenvectoren horend bij de k-grootste eigenwaarden worden geselecteerd, wordt Y hier getrunkeerd, dit word aangeduid met een tilde boven de matrix.

$$L \cdot \tilde{Y}^T \cdot D \cdot \tilde{U} = I \tag{2.35}$$

Nu wordt er een orthogonale matrix Q gezocht die het product $Y \cdot Q$ zo goed mogelijk niet-negatief maakt. Dit gebeurt aan de hand van een iteratief algoritme. Voor elke index (i, j) in Q wordt de Givens-rotatie berekend die de som van de derde machten van de elementen in kolommen i en j van Y zo groot mogelijk maakt:

$$\theta_{ij} = \arg\max_{\theta} \sum_{k} x_{ki}(\theta)^3 + x_{kj}(\theta)^3 \tag{2.36}$$

$$\text{met } X(\theta) = Y \cdot G(i, j, \theta) \tag{2.37}$$

Elke iteratie wordt de Y-matrix geüpdatet: de givensrotatie die 2.36 maximaliseert wordt daadwerkelijk ook uitgevoerd:

$$Y^{new} = Y^{old} \cdot G(i, j, \theta_{ij}) \tag{2.38}$$

De matrix Q is het product van al deze Givens-rotaties. Hij kan per iteratie mee geüpdatet worden. Uiteindelijk wordt een matrix H bekomen die het product is van de eigenvectormatrix Y (die orthogonaal is) en de matrix Q, dat op zich een product is van alle Givens-rotaties, en dus ook orthogonaal is. Deze matrix heeft zoveel mogelijk niet-negatieve elementen. Aan het einde worden alle negatieve elementen op nul gezet, zodat de matrix H ongeveer gelijk blijft en helemaal niet-negatief wordt.

$$H = [Y^{orig} \cdot G(1,1,\theta_{11})G(1,2,\theta_{12}) \dots G(1,R,\theta_{1,R})G(2,1,\theta_{21}) \dots G(R,R,\theta_{RR})]_+ \tag{2.39}$$

of korter samengevat:

$$H = [Y^{orig} \cdot Q]_+ \tag{2.40}$$

Deze matrix wordt nu gebruikt als initialisatie voor het niet-negatieve trifactorisatie algoritme met symmetrie-contraints. Dit is een variant van het NMF-algoritme en minimaliseert volgende doelfunctie:

$$\min_{H \geq 0} \|V - H^T \cdot S \cdot H\|^2 \tag{2.41}$$

De bekomen H-matrix bevat de (zachte) indicaties van de gezochte clusters.

2.4 Vergelijkingsexperimenten tussen de clustertechnieken

Voor de spraakherkenningsexperimenten, die beschreven worden in hoofstuk 3, moet een goede clustertechniek gevonden worden. Een belangrijke voorwaarde hierbij is dat het hele algoritme zelflerend moet zijn. Er mag geen kennis van buitenaf in het algoritme worden ingebouwd, dus ook niet in de clustertechniek. Hoewel de twee besproken algoritmes, nl. niet-negatieve matrixfactorisatie en spectrale clustering, beide zelflerend zijn, zijn er toch soms bepaalde parameters die een bepaalde schatting vereisen. De verschillende clustertechnieken, die in secties 2.1 tot 2.3 toegelicht zijn, werden uitgetest op een artificieel probleem, om zo de geschikte methode te kunnen kiezen voor de spraakherkenningsexperimenten.

2.4.1 Artificieel Probleem

Het artificiële probleem werd in het leven geroepen omdat de clustertechnieken niet rechtstreeks op de spraakdata zouden moeten worden getest. Bij de spraakdata kan het resultaat van de clustering niet direct geëvalueerd worden. Dit om twee redenen. Ten eerste is het niet mogelijk de data visueel voor te stellen. De featurevectoren waaruit de spraakdata bestaat, zijn van een dimensie die veel groter is dan drie. Het is dus niet mogelijk visueel na te gaan of de gevonden clusters weldegelijk groepen vormen in deze featureruimte. Ten tweede is het niet vooraf gekend

wat de clusters zullen zijn, dit wordt uitgelegd in hoofdstuk 3. Het bekomen resultaat kan dus niet vergeleken worden met een juist resultaat, omdat dit niet bekend is.

Om deze redenen is er een artificieel probleem opgesteld. Er wordt gewerkt met 2-dimensionale data, die gemakkelijk grafisch kan weergegeven worden. Deze data wordt willekeurig gegenereerd, en bevindt zich in op voorhand gekende clusters. Er zijn drie verschillende configuraties die de clusters kunnen aannemen, deze zijn afgebeeld in figuur 2.3. In elk probleem kan het aantal punten gekozen worden. In deze experimenten werd gewerkt met 4000 punten. Dit aantal is groot genoeg om geen afwijkingen te krijgen door toevallige configuraties van de punten, en is klein genoeg om de rekentijd van de algoritmes laag te houden, zodat er veel experimenten kunnen worden uitgevoerd.

In elk van deze testconfiguraties is een bepaalde moeilijkheidsfactor ingebouwd.

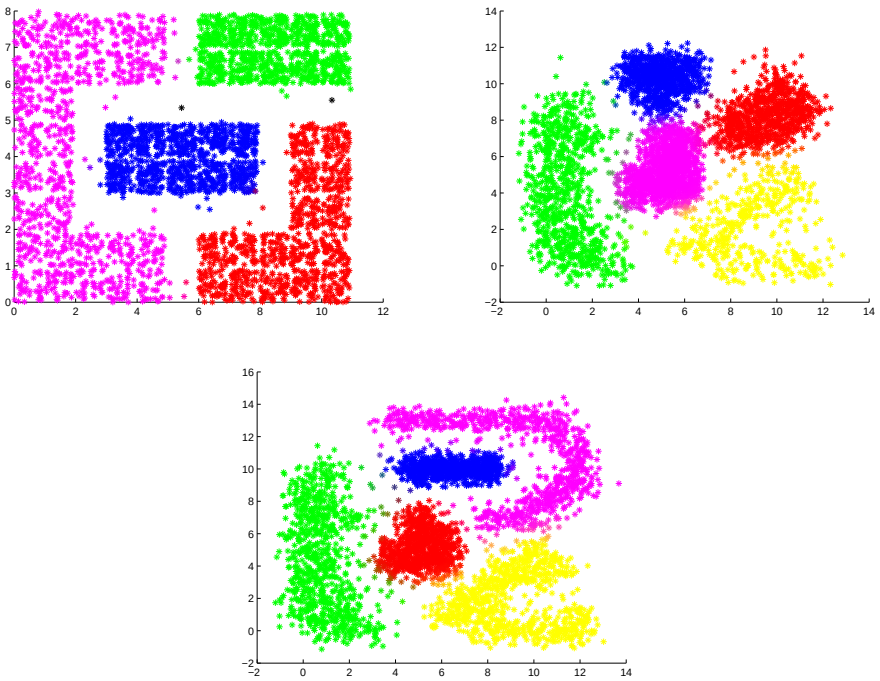
- In de eerste testset is er een cluster die groter is dan de andere: de linkse cluster op de figuur is zowel in oppervlakte als in puntenaantal ongeveer dubbel zo groot als de middelste cluster. De moeilijkheid bij deze testconfiguratie is dus de ongebalanceerdheid van de clusters.
- In de tweede testset liggen de clusters dicht bij elkaar. Dit zal het effect hebben dat het vaak voorkomt dat punten veel dichtste buren hebben uit andere clusters.
- In de derde testset is er een cluster die binnen een andere ligt. Dit is vaak een aanleiding tot fouten in clusteralgoritmes.

Per experiment werd ook 'ruis' toegevoegd aan de data. Dit zijn punten die tussen de clusters in liggen en de scheiding tussen de clusters minder duidelijk maakt.

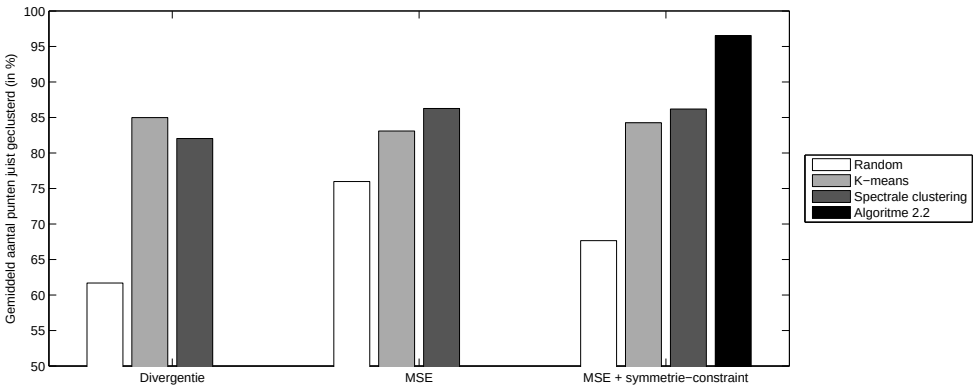
Aangezien de exacte clustering bekend is, kan de performantie van de clustertechniek berekend worden als het aantal punten dat aan de juiste cluster werd toegekend. Er wordt eerst een permutatie berekend van de rijen van de H-matrix om deze zo goed mogelijk te laten overeenstemmen met de juiste clustering. Deze permutatie is nodig omdat de performantie van de clustering niet zou afhangen van het feit of het algoritme dezelfde volgorde gekozen heeft als de correcte oplossing. Deze performantiemaat geeft een goede indicatie van de correctheid van clustering, maar is niet helemaal juist. De correcte oplossing kan minderwaardig zijn aan een gevonden oplossing omdat punten die voor een bepaalde cluster gegenereerd zijn zo ver komen van het centrum, dat ze zich in een andere cluster bevinden. Een clusteralgoritme moet het punt dan toekennen aan de cluster waarin het zich bevindt, de performantiemaat zal dit echter fout rekenen omdat het label van het punt een andere cluster aangeeft.

2.4.2 Resultaten

De resultaten van de experimenten zijn te zien op figuur 2.5. De resultaten voor de drie testconfiguraties worden onder elkaar afgebeeld. Voor elke clustertechniek is de performantie weergegeven. In tabel 2.1 staat welke de techniek is horend bij het nummer in de figuur. De eerste drie clustertechnieken zijn NMF-varianten, met verschillende criteria en constraints. De volgende vier hebben gemeenschappelijk dat op de invoerdata K-means wordt uitgevoerd: de eerste techniek is K-means zelf, de volgende drie gebruiken dit resultaat als initialisatie voor NMF. Clustertechnieken 8 tot en met 11 zijn implementaties van spectrale clustering. Het verschil in deze implementaties bevindt zich na de berekening van de eigenvectoren. In



Figuur 2.3: 3 testconfiguraties van het artificiële probleem. Linksboven testset 1, rechtsboven testset 2, onderaan testset 3



Figuur 2.4: Performantie van clustering via NMF met verschillende initialisaties.

de technieken 9 en 11 worden de rijen van de eigenvectormatrix genormaliseerd voor de K-means clustering. Een tweede verschil is de afstandsmaat die gebruikt wordt bij de K-means clustering. Techniek 8 en 9 gebruiken de Euclidische afstand, technieken 10 en 11 gebruiken de cosinus-afstand:

$$d(\vec{x}, \vec{y}) = 1 - \frac{x^T y}{\|x\| \cdot \|y\|} \tag{2.42}$$

De volgende technieken, nummer 12 tot en met 14, zijn opnieuw varianten van NMF. Deze gebruiken het resultaat van de spectrale clustering met cosinus-afstand en rijnormalisatie (techniek 11) als initialisatie voor de H-matrix. De laatste techniek (nummer 15) is de techniek die beschreven is in sectie 2.3.

Uit deze resultaten kunnen enkele conclusies getrokken worden. Een eerste is over het criterium dat gebruikt wordt in Niet-negatieve Matrixfactorisatie. Uit de resultaten blijkt dat het MSE-criterium een meer geschikt criterium is voor clustering dan het divergentiecriterium. Wanneer het NMF-algoritme random geïnitialiseerd wordt, is de performantie van de NMF met MSE-criterium 15% tot 20% hoger dan de performantie van NMF met divergentiecriterium.

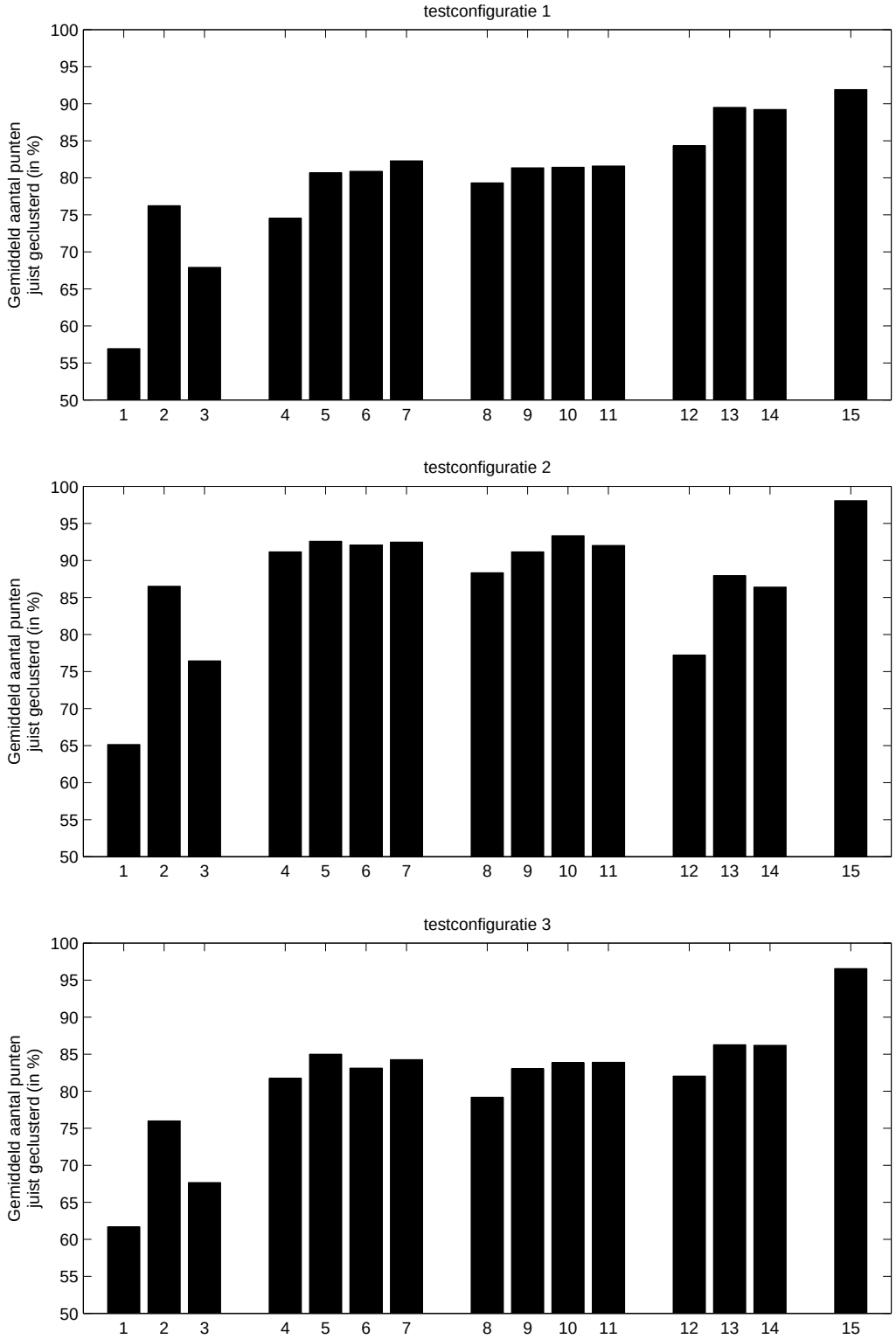
In deze experimenten is ook duidelijk naar voor gekomen dat de initialisatie van de matrices W en H heel belangrijk is voor het slagen van de clustering. Er werden verschillende initialisaties getest. Ten eerste is er de random initialisatie, alle elementen worden willekeurig gegenereerd tussen 0 en 1, ten tweede wordt eerst het k-means algoritme uitgevoerd op de data, en de uitkomst hiervan wordt gebruikt als initialisatie voor de H-matrix. De W-matrix wordt nog steeds random geïnitialiseerd. Een derde methode gebruikt de uitkomst van spectrale clustering als initialisatie voor de H-matrix van NMF. De vierde techniek is degene die beschreven is in sectie 2.3, deze is niet dezelfde als de initialisatie met spectrale clustering, aangezien enkel de eigenvectoren van de spectrale clustering gebruikt worden, en niet het resultaat van de spectrale clustering. In figuur 2.4 zijn de resultaten van deze technieken weergegeven. Het is duidelijk te zien dat de random initialisatie de minste performantie heeft. Initialisaties via k-means en spectrale clustering verhogen de performantie aanzienlijk, nl. tot zo'n 80 à 85 procent. De techniek uit sectie 2.3 heeft de beste performantie, nl. gemiddeld worden 96% van de punten juist geclusterd.

Op de vraag of spectraal clusteren dan wel niet-negatieve matrixfactorisatie de beste resultaten geeft, is het antwoord minder duidelijk. Spectraal clusteren presteert merkelijk beter dan NMF met random initialisatie. Wanneer de niet-negatieve matrixfactorisatie via k-means wordt geïnitialiseerd, is de performantie vergelijkbaar aan die van spectraal clusteren. Het resultaat van de spectrale clustering gebruiken als initialisatie voor NMF leidt niet steeds tot betere resultaten dan de spectrale clustering zelf. Indien de spectrale clustering berekend is, loont het dus niet de moeite om dit resultaat te gebruiken als initialisatie voor NMF.

De laatste conclusie is dat de voorgestelde methode, beschreven in sectie 2.3, een significante verbetering aanbrengt aan het algoritme van NMF. Voor de drie testconfiguraties worden respectievelijk performanties behaald van 91%, 98% en 97%.

1	NMF met divergentiecriterium
2	NMF met MSE-criterium
3	NMF met MSE-criterium en symmetrie-constraint
4	K-means
5	NMF met divergentiecriterium; initialisatie met K-means
6	NMF met MSE-criterium; initialisatie met K-means
7	NMF met MSE-criterium en symmetrie-constraint; initialisatie met K-means
8	Spectrale clustering met Euclidische afstand
9	Spectrale clustering met Euclidische afstand; eigenvectoren genormaliseerd
10	Spectrale clustering met cosinus-afstand
11	Spectrale clustering met cosinus-afstand; eigenvectoren genormaliseerd
12	NMF met divergentiecriterium; initialisatie via spectrale clustering
13	NMF met MSE-criterium; initialisatie via spectrale clustering
14	NMF met MSE-criterium en symmetrie-constraint; initialisatie via spectrale clustering
15	Algoritme zoals beschreven in onderdeel 2.3

Tabel 2.1: Geteste clustertechnieken met het nummer in figuur 2.5



Figuur 2.5: Performantie van de verschillende clustertechnieken in de verschillende testconfiguraties. Zie tabel 2.1 voor de technieken die overeenkomen met de nummers in de plots.

Hoofdstuk 3

Herkenning van akoestische eenheden

3.1 Inleiding

In de taalwetenschappen worden uitingen in gesproken of geschreven taal opgesplitst in steeds kleinere elementen: een uiting bestaat uit zinnen, een zin bestaat uit woorden, een woord is opgebouwd uit klanken. Deze opdeling wordt door de mensen intuïtief aangenomen: wanneer we naar spraak luisteren, herkennen we deze klanken, woorden en zinnen. In spraakherkenners wordt hiervan uitgegaan om de modellen op te stellen op basis waarvan de spraak herkend wordt. In deze masterproef wordt nagegaan of die opdeling in *fonemen* aanwezig is in de gesproken taal. Welke akoestische eenheden duiken op in de geluidspatronen van de uitingen in een bepaalde taal?

In deze experimenten worden de fysische geluidsgolven eerst bemonsterd en ingedeeld in frames. Per frame worden dan de features eruit geëxtraheerd. Deze features vormen een compacte voorstelling van de akoestische eigenschappen in het tijdsframe. In deze verzameling van features worden akoestische eenheden gezocht waaruit de spraak is opgebouwd. Dit wordt gerealiseerd door de features te clusteren met de clustertechniek uit hoofdstuk 2. De bekomen clusters - de automatisch geleerde akoestische eenheden - worden vergeleken met de taalkundige akoestische eenheden, nl. de fonemen.

Voor de spraakdata wordt gebruikt gemaakt van de *Wall Street Journal (WSJ) database*. Deze database bestaat vooral uit voorgelezen spraak met teksten afkomstig uit de krant 'Wall Street Journal'. Deze database werd gebouwd om onderzoek mogelijk te maken voor Continuous Speech Recognition (CSR) systemen met grote vocabularia. WSJ is dus een database met een grote woordenschat en nagenoeg alle Engelse fonemen, nl. 43. De symbolen voor deze fonemen zijn weergeven in bijlage 3.1.

3.2 Opbouw van de methode

In dit onderdeel wordt de volledige methode beschreven die de akoestische eenheden via een zelflerend algoritme uit de spraakdata extraheert.

3.2.1 Clustering

De bemonsterde geluidsdata wordt verdeeld in overlappende frames. Voor elk frame wordt een featurevector berekend. Deze featurevector hangt echter ook af van de omliggende frames, omdat gewerkt wordt met afgeleiden tussen de frames. Deze verzameling featurevectoren dient nu als input voor de tweede stap.

In de tweede stap wordt het algoritme uit sectie 2.3 toegepast op de featuredata. Uit de WSJ-database werden een drie testsets geselecteerd. Het aantal frames in deze testsets is respectievelijk 500000, 1 miljoen en 2 miljoen. Voor deze stap zijn er twee parameters, nl. de methode voor symmetrisatie van de similariteitsmatrix - and- of or-bewerking - en het aantal buren in de knn-matrix. In de experimenten werden volgende waardes gebruikt: 63, 127, 255 en 511. Tel daarbij nog één op voor het punt in kwestie zelf, en het totale aantal elementen in elke kolom van de knn-matrix vormt een macht van twee.

Dit clusteralgoritme heeft als output de ontbinding van de knn-matrix V:

$$V = H^T \cdot S \cdot H \tag{3.1}$$

De matrix H bevat de indicatorvectoren, die voor elke featurevector aangeven tot welke cluster die behoort. Vaak zal één element in die vector dominant zijn, wat erop wijst dat de overeenkomstige featurevector tot die cluster behoort. Merk op dat de kolomindex in H overeenkomt met de kolomindex in V, wat op zich weer overeenkomt met de frame-index. De matrix H geeft dus voor alle uitingen in de trainingsset de clusterindicaties weer in chronologische volgorde. De matrix H moet dus als volgt geïnterpreteerd worden: de horizontale as geeft de tijd weer, de verticale as bevat de verschillende clusters.

3.2.2 Oplijning van de H-matrix

Nu de matrix H gekend is, kan gezocht worden naar de eigenlijke segmentatie van de uitingen in de spraakdata.

Een segmentatie vereist dat op elk tijdstip eenduidig gekend is welk foneem (of in dit geval akoestische eenheid) uitgesproken wordt. Als in elk frame het grootste element in de overeenkomstige kolomvector van H geselecteerd wordt, kan deze segmentatie op een eenvoudige manier bereikt worden. De rij-index van dit element geeft weer welke akoestische eenheid op dat moment geuit wordt.

Dit is echter niet de beste oplossing. De spraakproductie is een continu proces: voor het vormen van de verschillende klanken moeten de tong, lippen, kaken enz. overgaan van van de ene stand in de andere, zodat er telkens overgangen zijn tussen de verschillende klanken. Het is bijgevolg niet gemakkelijk duidelijke grenzen af te bakenen voor de fonemen of akoestische eenheden. Het

kan dan ook voorkomen dat bij deze overgang van foneem A naar B heel tijdelijk een foneem C geproduceerd wordt. Daardoor zal dit in de matrix H het grootste element van de kolom zijn.

Een ander probleem is dat er ruis in de data voorkomt, die ervoor kan zorgen dat de akoestische eigenschappen van het frame in kleine mate afwijken van wat normaal verwacht wordt. Op zijn beurt zou dit als gevolg kunnen hebben dat dit frame in de H-matrix een hogere waarde heeft voor een cluster die eigenlijk niet de akoestische eenheid voorstelt, dan dat het frame zonder ruis zou voorstellen.

Deze problemen kunnen opgelost worden met een Viterbi-oplijning. Viterbi berekent het meest waarschijnlijke pad dat gevolgd werd in de uitingen. Op die manier wordt er rekening gehouden met het feit dat een korte overgang naar een ander foneem - om daarna weer terug te keren - veel minder waarschijnlijk is dan dat het foneem aangehouden blijft.

De Viterbi-oplijning is een iteratief algoritme met als doel de likelihood van de featurevectoren te maximaliseren. Daarbij worden bepaalde veronderstellingen gemaakt en benaderingen gebruikt. Om dit duidelijk te stellen, worden de updateformules van het algoritme hier afgeleid.

$\phi_t(j)$ is de kans dat de featuredata tot op tijdstip t gegenereerd is en dat de akoestische eenheid op tijdstip t gelijk is aan i. Om deze notatie eenvoudig te houden, worden hier 2 symbolen ingevoerd: A_t is de akoestische eenheid op tijdstip t, O_t is de featurevector op tijdstip t. De kans wordt als volgt uitgewerkt:

$$\begin{aligned}
\phi_t(i) &= P(A_t = i, O_1, \ldots, O_{t-1}, O_t) && (3.2)\\
&= P(O_t|A_t = i, O_1, \ldots, O_{t-1}) P(A_t = i, O_1, \ldots, O_{t-1}) && (3.3)\\
&= P(O_t|A_t = i, O_1, \ldots, O_{t-1}) \sum_j P(A_t = i, A_{t-1} = j, O_1, \ldots, O_{t-1}) && (3.4)\\
&= P(O_t|A_t = i, O_1, \ldots, O_{t-1}) \\
&\qquad \sum_j P(A_t = i|A_{t-1} = j, O_1, \ldots, O_{t-1})\ P(A_{t-1} = j, O_1, \ldots, O_{t-1}) && (3.5)
\end{aligned}$$

Hier worden enkele veronderstellingen gemaakt. De kans dat een featurevector gegenereerd wordt door het model is enkel afhankelijk van de akoestische eenheid op dat tijdstip. De tweede veronderstelling is dat de kans op een akoestische eenheid in een frame enkel afhankelijk is van de akoestische eenheid in het vorige frame. De kans $\phi_t(i)$ wordt nu verder uitgewerkt met deze veronderstellingen.

$$\begin{aligned}
\phi_t(i) &= P(O_t|A_t = i) \sum_j P(A_t = i|A_{t-1} = j)\ P(A_{t-1} = j, O_1, \ldots, O_{t-1}) && (3.6)\\
&= H_{it} \sum_j a_{ji} \phi_{t-1}(j) && (3.7)
\end{aligned}$$

In formule 3.7 werden de modelparameters ingevuld. a_{ji} is de transitiekans: de kans dat overgegaan wordt van akoestische eenheid j naar akoestische eenheid i. De matrix H bevat de emissiekansen: de kans dat de featurevector in frame t verklaard wordt door de akoestische eenheid i. De laatste kans is opnieuw ϕ, ditmaal voor tijdstip $t-1$ en akoestische eenheid j. Het algoritme kan dus recursief berekend worden, beginnend bij $t = 1$.

Dit algoritme wordt echter nog vereenvoudigd. De som in formule 3.7 wordt benaderd door de grootste term. Dit is een goede benadering indien er één term voorkomt in de som die veel

groter is dan de andere. De waarde die hier berekend wordt is dus slechts een benadering van de oorspronkelijke kans ϕ. Merk op dat in elke iteratie een benadering gemaakt wordt, deze benaderingen accumuleren dus per frame. ϕ wordt dus niet exact uitgerekend, maar enkel de benaderende $\tilde{\phi}$.

$$\phi_t(i) \approx H_{it} \max_j a_{ji}\phi_{t-1}(j) \tag{3.8}$$

$$\tilde{\phi}_t(i) = H_{it} \max_j a_{ji}\ \tilde{\phi}_{t-1}(j) \tag{3.9}$$

In elke iteratie wordt ook bijgehouden welke voorgaande akoestische eenheid de maximale waarde gaf voor formule 3.9:

$$\psi_t(i) = \arg\max_j a_{ji}\ \tilde{\phi}_{t-1}(j) \tag{3.10}$$

Wanneer het einde van de matrix (met frame-index T) bereikt wordt, wordt de maximale waarde van $\phi_T(i)$ gezocht voor alle akoestische eenheden i. De akoestische eenheid I die de maximale waarde geeft, is meteen ook het laatste element van het optimale pad. Via de bijhorende waarde $\psi_T(I)$ kan het hele optimale pad bekomen worden d.m.v. backtracking met deze ψ-waarden.

Omdat de waarden van ϕ voor elke iteratie kleiner worden, kan dit problemen geven als de ondergrens van de nauwkeurigheid in de computer bereikt wordt. Daarom wordt dit algoritme in het log-domein uitgevoerd:

$$\log \tilde{\phi}_t(i) = \log H_{it} + \max_j\{\log a_{ji} + \log \tilde{\phi}_{t-1}(j)\} \tag{3.11}$$

$$\psi_t(i) = \arg\max_j\{\log a_{ji} + \log \tilde{\phi}_{t-1}(j)\} \tag{3.12}$$

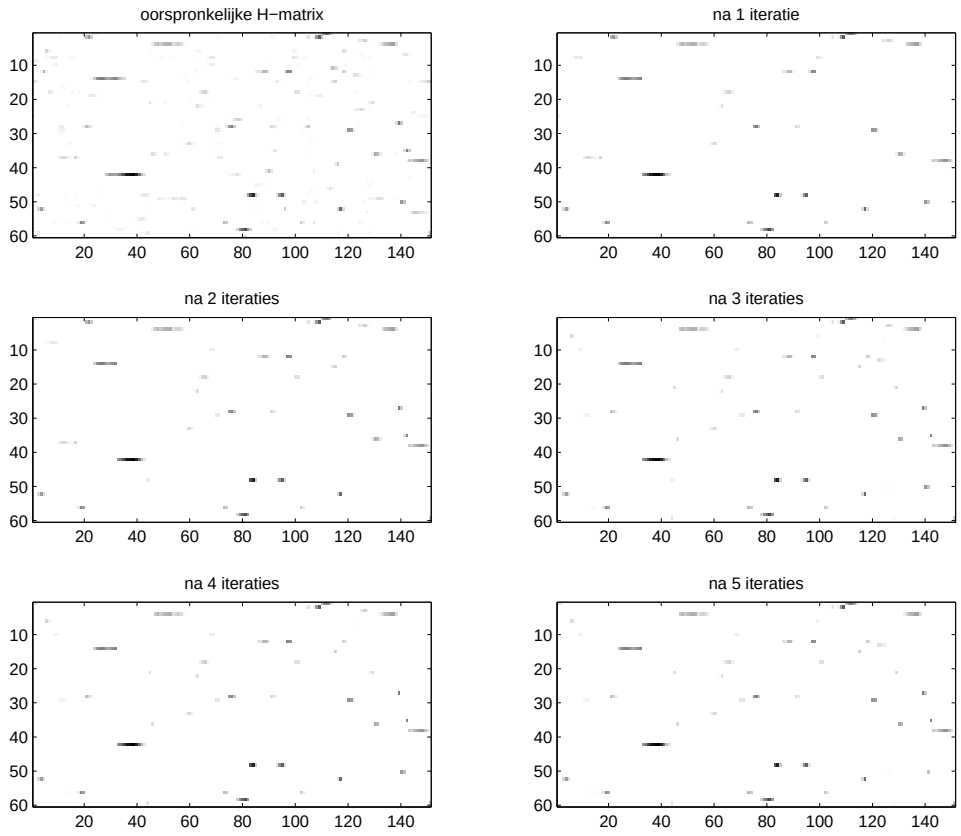
Het resultaat van de Viterbi-oplijning is dat er een segmentatie verkregen wordt. Er is eenduidig bepaald welke akoestische eenheid in dat frame aanwezig is. Een bijkomend voordeel is dat deze segmentatie veel robuuster zal zijn tegen ruis.

Om een stabiele oplossing te verkrijgen wordt dit algoritme enkele keren herhaald. Hierbij worden de transitiekansen geüpdatet aan de hand van de bekomen segmentatie in de vorige iteratie. Stel $n(i)$ het aantal keer dat akoestische eenheid i voorkomt in het optimale pad, en $n(i,j)$ het aantal keer dat akoestische eenheid i overgaat naar akoestische eenheid j op het optimale pad. De nieuwe transitiekansen worden dan als volgt berekend:

$$a_{ij} = \frac{n(i,j)}{n(i)} \tag{3.13}$$

Voor de eerste iteratie moet een initialisatie gevonden worden voor deze transitiekansen. Als model wordt gebruikt dat een akoestische eenheid een duur heeft van b. De initiële kansen worden dan

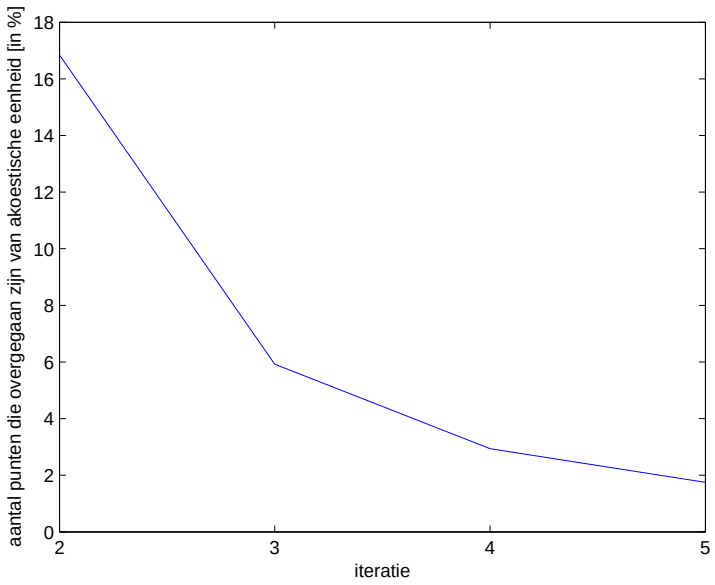
$$a_{ij} = \begin{cases} \frac{b}{b+1} & \text{als } i = j \\ \frac{1}{(R-1)(b+1)} & \text{als } i \neq j \end{cases} \tag{3.14}$$



Figuur 3.1: Resultaten van de viterbi-oplijning in verschillende iteratiestappen

In de uitgevoerde experimenten is de parameter b gelijk aan vijf. Aangezien nog niets gekend is over de akoestische eenheden, kan deze parameter in feite niet geschat worden. Dit is echter enkel een initialisatie en zal na enkele iteraties minder effect hebben op de verkregen oplijning, aangezien de transitiekansen steeds herschat worden.

Het resultaat van 5 iteraties van dit algoritme is te zien in figuur 3.1. Hier wordt een deel van de matrix H weergegeven, waarbij alle elementen die niet op het optimale pad liggen op nul worden gezet. Het effect van de Viterbi-oplijning is duidelijk merkbaar: kortstondige toewijzingen aan akoestische eenheden met een lage probabiliteit worden weggefilterd. Dit maakt het resultaat meer bestand tegen ruis. Deze filtering vindt vooral plaats in de eerste iteratie. Daarna worden bepaalde korte toewijzigen aan akoestische eenheden opnieuw ingevoerd, dit komt omdat de invloed van de initialisatie van de transitiekansen vermindert. Het resultaat wordt betrouwbaarder omdat de transitiekansen op de data worden geschat i.p.v op een veronderstelling. Na de derde iteratie is het algoritme kennelijk geconvergeerd. In figuur 3.2.2 is weergegeven hoeveel punten overgaan van akoestische eenheid voor elke iteratiestap. Na enkele iteraties blijft de segmentatie in akoestische eenheden zo goed als gelijk: deze convergentie toont aan dat een stabiele oplossing gevonden wordt.



Figuur 3.2: Convergentie van de viterbi-oplijning

3.3 Vergelijking van de gevonden akoestische eenheden met fonemen

3.3.1 Opstelling van het experiment

Het volledige algoritme om de spraakuitingen te segmenteren is nu bekend en kan gebruikt worden om de akoestische eenheden te zoeken in spraakuitingen. De segmentaties die in de experimenten berekend zijn, worden in een segmentatiebestand neergeschreven. Een voorbeeld hiervan is te vinden in bijlage A.1.

De uitingen in de WSJ-database zijn door menselijke hand getranscribeerd: van elke uiting is bepaald uit welke woorden en fonemen die bestaat. Met behulp van de ESAT-spraakherkenner kan voor de testset een segmentatie aangemaakt worden, die voor elk frame aangeeft wat het foneem is dat op het tijdstip van dat frame wordt uitgesproken. Een voorbeeld van deze segmentatie is te vinden in bijlage A.2.

De segmentatie die via het clusteralgoritme is berekend, wordt nu vergeleken met de segmentatie in fonemen. Op basis hiervan wordt nagegaan in welke mate de gevonden akoestische eenheden overeenkomen met de fonemen. Daarvoor wordt een coöccurence-matrix opgesteld, die voor elke combinatie van foneem en akoestische eenheid aangeeft in hoeveel frames deze combinatie voorkomt. De formule voor de coöccurence-matrix is:

$$C(i,j) = \sum_t \Delta_{ij}^t \tag{3.15}$$

$$\Delta_{ij}^t = \begin{cases} 1 & \text{als } P_t = i \text{ en } A_t = j \\ 0 & \text{in het andere geval} \end{cases} \tag{3.16}$$

Nr.	Symbool	Voorbeeldwoord	Nr.	Symbool	Voorbeeldwoord
1	#	[stilte]	23	r	radio
2	i	see	24	s	six
3	I	increases	25	z	zero
4	I!	fifteen	26	S	show
5	E	any	27	Z	usual
6	e+I	day	28	t+S	much
7	j	yesterday	29	d+Z	general
8	a+I	why	30	t	two
9	A	market	31	d	did
10	O	cross	32	T	three
11	O+I	point	33	D	the
12	{	has	34	k	calls
13	@	about	35	g	go
14	@!	double	36	p	point
15	u	union	37	b	buying
16	U	push	38	m	mister
17	a+U	house	39	n	next
18	@+U	over	40	N	think
19	w	we	41	f	five
20	l	last	42	v	vote
21	}	mister	43	h	help
22	}!	purchase			

Tabel 3.1: Fonemen met nummer en symbool

Hier zijn P_t en A_t respectievelijk het foneem en de akoestische eenheid in frame t. De kolomindex is het nummer van de akoestische eenheid, de rij-index is het nummer het foneem. Tabel 3.1 geeft weer welk foneem met dit nummer overeenkomt. Deze coöccurence-matrix kan nog genormaliseerd worden volgens de rijen of de kolommen. Bij een rijnormalisatie – de rijsom wordt 1 gemaakt – kan duidelijk gezien worden hoe de fonemen gemapt worden op de akoestische eenheden. Een kolomnormalisatie verduidelijkt de mapping van akoestische eenheden op fonemen.

3.3.2 Algemene bespreking van de resultaten

De bespreking van de experimenten verloopt a.d.h.v. een referentieresultaat. Het resultaat is weergegeven in figuur 3.3. Het is berekend met een trainingsset van 495.913 frames, er werden in de eerste stap van het algoritme 255 buren geselecteerd, en de gebruikte symmetrie-operatie is het gemiddelde. Er werden 60 akoestische eenheden gezocht ($R = 60$).

De figuur laat zien dat er overeenkomsten bestaan tussen de gevonden akoestische eenheden en de fonemen. Indien de coöccurencematrix een diagonaalmatrix was, dan bestond er een pefecte 1-op-1 mapping tussen akoestische eenheden en fonemen. Het verkregen resultaat is geen diagonaalmatrix, maar vertoont wel in zekere mate een diagonaalstructuur, wat erop wijst

dat er sterke correlaties zijn tussen de akoestische eenheden en fonemen. Daarnaast zijn er vele afwijkingen.

Een eerste afwijking is dat vele fonemen op meerdere akoestische eenheden worden gemapt. Wanneer de rijgenormaliseerde voorstelling vergeleken wordt met de kolomgenormaliseerde voorstelling, dan blijkt de kolomgenormaliseerde voorstelling een meer spaarse structuur te hebben en dus betere resultaten te geven. Het mappen van akoestische eenheden op fonemen gaat beter dan het mappen van de fonemen op akoestische eenheden, of met andere woorden: voor een akoestische eenheid kan beter gezegd worden welk foneem het betreft, dan dat van een foneem kan gezegd worden door welke akoestische eenheid het wordt gemodelleerd. Het feit dat meerdere akoestische eenheden op hetzelfde foneem mappen heeft meerdere oorzaken.

- Er zijn meer akoestische eenheden dan fonemen. Bijgevolg worden clusters die overeenkomen met één bepaald foneem gesplitst. Het clusteralgoritme splitst vooral de grootste clusters, aangezien het algoritme zoekt naar clusters met ongeveer dezelfde grootte. Het typevoorbeeld hiervan is de cluster die de stilte ('#') modelleert. Er zijn 129180 stilteframes, dit is een factor 10 groter dan de ideale clustergroootte van 11533 frames. Dit laatste is de grootte die de clusters zouden hebben indien ze allemaal even veel elemeten bevatten (volledig gebalanceerd zijn). Het stilte-foneem wordt gemapt op 5 akoestische eenheden (1, 36, 43, 49 en 54), met groottes variërend van 21564 tot 38811 frames.
- Een tweede oorzaak is dat het foneem uit meerdere delen bestaat, die elk afzonderlijk gemapt worden op een akoestische eenheid. Een voorbeeld hiervan is het foneem 't+S'. Dit wordt vooral gemapt op de akoestische eenheden 26 en 30, en in mindere mate op 28 en 34. Het eerste deel van dit foneem is de /t/-klank, dat onmiddelijk gevolgd wordt door de /S/-klank (in het Engels geschreven als 'sh'). Dit stemt overeen met de akoestische eenheden waarop het mapt: akoestische eenheid 26 modelleert de /S/-klank, eenheid 30 de /t/-klank. Dit foneem wordt in de fonetiek omschreven als een affricaat: het begint als een plosief en eindigt als een fricatief. De releasefase van het plosief wordt vervangen door het fricatief. Het affricaat heeft dus een eigen karakteristiek en is meer dan een combinatie van twee fonemen. Dit is de verklaring waarom het foneem /t+S/ ook een aparte akoestische eenheid heeft (nr. 28) naast de eenheden die de twee delen van het foneem modelleren.
- Of een foneem beklemtoond is of niet kan grote verchillen veroorzaken in de akoestische eigenschappen van het foneem, maar toch als hetzelfde foneem bestempeld worden. Een foneem kan ook licht afwijkende uitspraakvarianten hebben, die wel hoorbaar zijn, maar waar de luisteraar onmiddellijk herkent dat het om hetzelfde foneem gaat. Deze varianten worden *allofonen* genoemt. Een voobeeld hiervan is de aspiratie van plosieven (d.i het uitstoten van lucht onmiddelijk volgend op het plosief) in het Engels. In het Engels worden stemloze plosieven geaspireerd wanneer ze aan het begin van een bekemtoonde lettergreep staan. In het woord 'paper' wordt enkel de eerste p geaspireerd. Dit verschil is duidelijk hoorbaar, maar wordt als hetzelfde foneem aangenomen. Deze twee effecten kunnen ervoor zorgen dat verschillende akoestische eenheden de varianten apart benaderen.
- Wat ook niet mag vergeten worden, is dat de WSJ-database uitingen bevat van veel verschillende sprekers. Het is bekend dat de verschillen in akoestische eigenschappen van de uitingen door verschillende personen heel groot kan zijn, ook al is dit voor de mens niet duidelijk waar te nemen. Deze akoestische verscheidenheid resulteert opnieuw in een sterke (ver)spreiding van de data horend bij hetzelfde foneem, zodat de clusteralgoritmes deze data niet of moeilijk aan dezelfde cluster kunnen toewijzen.

Een tweede afwijking is dat akoestische eenheden vaak ook fonemen mappen die sterke gelijkenissen vertonen met het foneem waarop het normaal mapt. In de figuur is dit te zien als kleine blokstructuren. Ook hier zijn er meerdere oorzaken. In de volgende sectie worden daar enkele oorzaken voor aangegeven.

Naast deze afwijkingen die hun oorsprong vinden in het domein van de akoestische eigenschappen van fonemen, zullen er ook fouten optreden die te maken hebben met de clustering zelf. Indien de clusters ongebalanceerd zijn, zal het clusteralgoritme kleine clusters samenvoegen en grote clusters splitsen. De gebalanceerdheid is ingebouwd in het model om te clusteren. Toch is dit niet altijd een juiste veronderstelling. Zo zal de akoestische eenheid die stilte modelleert gesplitst moeten worden, omdat het aantal stilte-frames heel groot is in vergelijking met het aantal frames van andere fonemen. Een tweede probleem, dat in het vorig hoofdstuk werd besproken, is dat het algoritme naar een lokaal minimum convergeert. Vaak berekent het algoritme een suboptimale oplossing, terwijl mogelijk een betere clustering bestaat. Een derde probleem is dat het NMF-algoritme ook beperkingen heeft. Het houdt enkel rekening met de dichtste-buren relaties. Om akoestische patronen te herkennen, kan het bijvoorbeeld ook belangrijk zijn om informatie te hebben over welke de voorgaande geuitte fonemen waren. In dit algoritme houdt NMF hier geen rekening mee.

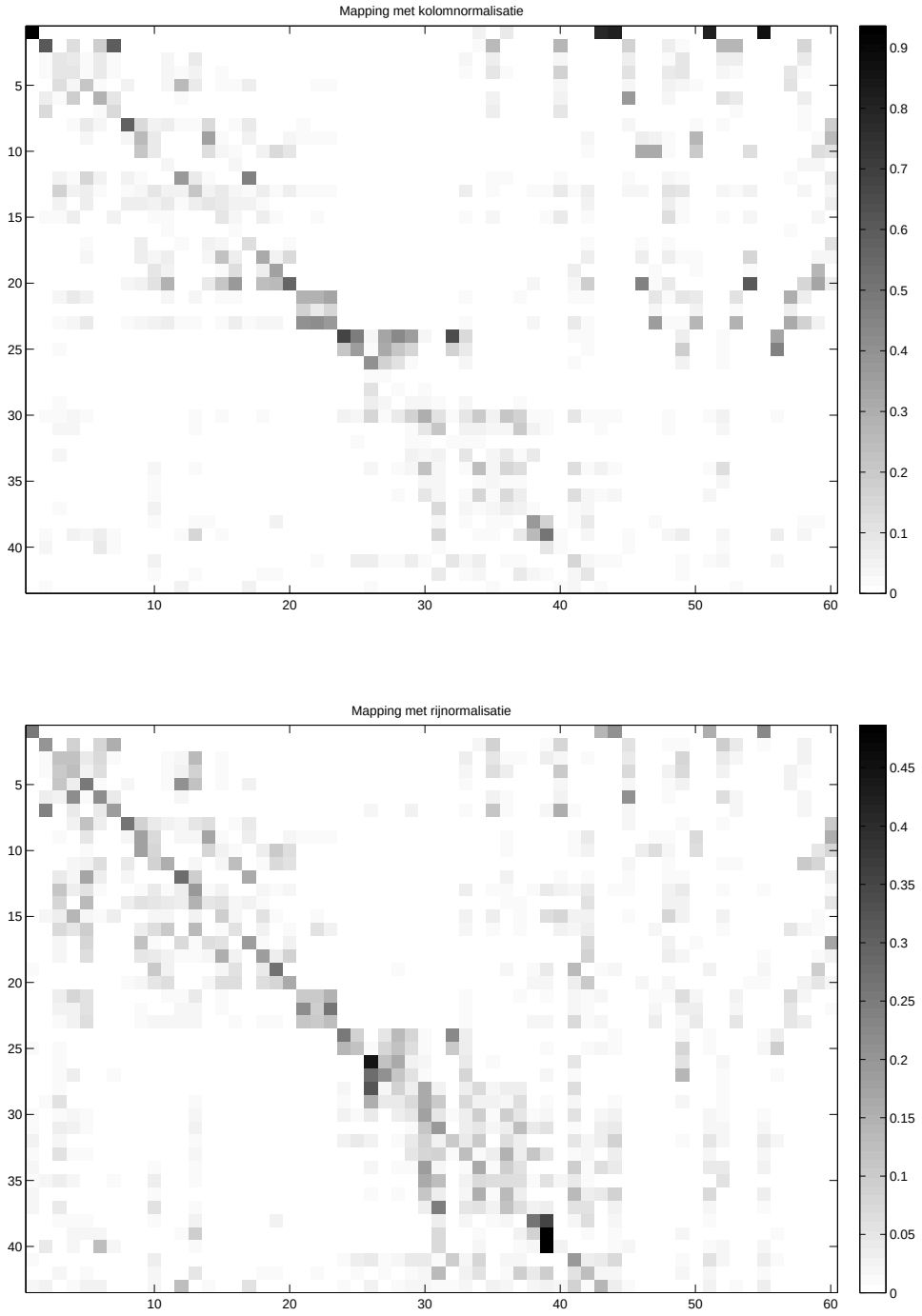
3.3.3 Invloed van parameters op de mapping

Om na te gaan wat de invloed is van de parameters van de gebruikte clustertechniek op het vinden van de akoestische eenheden, werd deze mapping berekend voor een aantal verschillende parameters. De resultaten van deze mappingen zijn te vinden in bijlage B. Bij deze resulaten is echter heel weinig verschil te merken. De invloed van de AND-symmetrie (zie vergelijking 2.30) is wel goed merkbaar. Wanneer er te weinig buren genomen worden, vertoont het resultaat vele kleine clusters van bijv. 2 punten. Zo'n resultaat is te zien in tabel 3.3.3. De oorzaak hiervan is dat de AND-bewerking vele connecties zal verwijderen, vooral veraf gelegen punten. Deze vormen dan kleine groepjes die door de clustering allemaal aan een aparte cluster toegewezen worden. De oplossing voor dit probleem is een andere symmetrie-bewerking te gebruiken.

3.3.4 Onderzoek van de eigenvectoren en eigenwaarden van de laplaciaan

Hiërarchische clustering

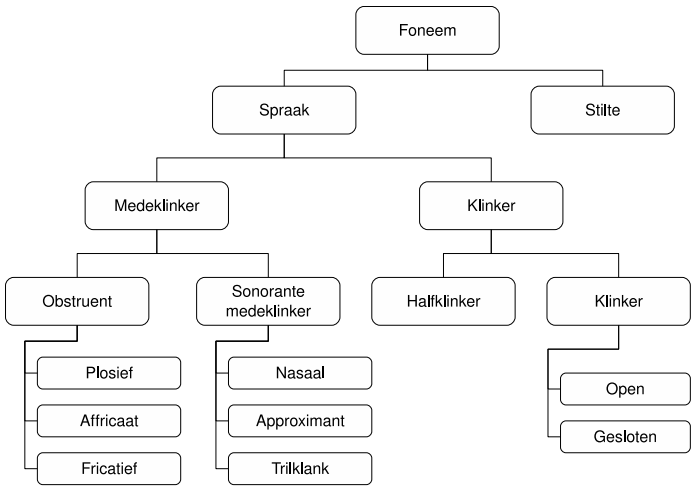
Bij het nader bekijken van figuur 3.3 is het opmerkelijk dat er een horizontale grens aanwezig is midden de figuur, nl. tussen de fonemen 23 en 24. Ook verticaal kan een grens gezien worden tussen de akoestische eenheden 23 en 24. De coöccurence-matrix vertoont in zekere mate een blokstructuur. De akoestische eenheid en het foneem die stilte modelleren kunnen beschouwd worden als een apart blok. De blokstructuur duidt aan dat er zijn vooral overlappingen zijn binnen de groep van fonemen en akoestische eenheden met nummers 2 tot 23 en binnen de groep van fonemen en akoestische eenheden met nummers 24 tot 43. Dit gegeven wijst erop dat de fonemen en akoestische eenheden opgedeeld kunnen worden in twee categorieën. De groep 2-23 komen ruwweg overeen met de klinkers, de groep 24-43 met medeklinkers. In de vorige paragraaf werd beschreven dat bepaalde fonemen en akoestische eenheden nog sterkere gelijkenissen vertonen en dus verward worden bij de herkenning. Dit is te zien in de figuur als



Figuur 3.3: Mapping van gevonden akoestische eenheden op fonemen. Parameters: 60 akoestische eenheden, OR-symmetrie, 255 buren, 495.913 frames

akoestische eenheid	aantal buren	akoestische eenheid	aantal buren	akoestische eenheid	aantal buren
S17	85712	S52	1679	S44	17
S37	39712	S58	261	S31	17
S18	28967	S48	241	S35	17
S50	27533	S63	146	S54	16
S46	27403	S67	58	S25	14
S19	23304	S38	53	S12	14
S42	22969	S73	53	S71	13
S65	21190	S45	44	S13	13
S56	19985	S30	32	S26	13
S36	19497	S5	30	S22	11
S34	19075	S41	27	S47	11
S21	18397	S43	27	S6	9
S33	15259	S55	26	S29	9
S27	15225	S15	25	S4	8
S72	13760	S10	24	S39	8
S68	12963	S53	24	S60	7
S51	12680	S2	23	S7	6
S75	11247	S24	22	S9	5
S20	10702	S69	21	S11	5
S70	9738	S14	21	S8	4
S40	9593	S16	20	S74	2
S57	8611	S1	20	S61	2
S62	7770	S59	20	S49	2
S64	6635	S3	19	S32	2
S28	4826	S66	17	S23	2

Tabel 3.2: Akoestische eenheden met lengte in aantal frames. Resultaat bekomen met parameters: AND-symmetrie en 255 buren



Figuur 3.4: Hiërarchie van fonemen

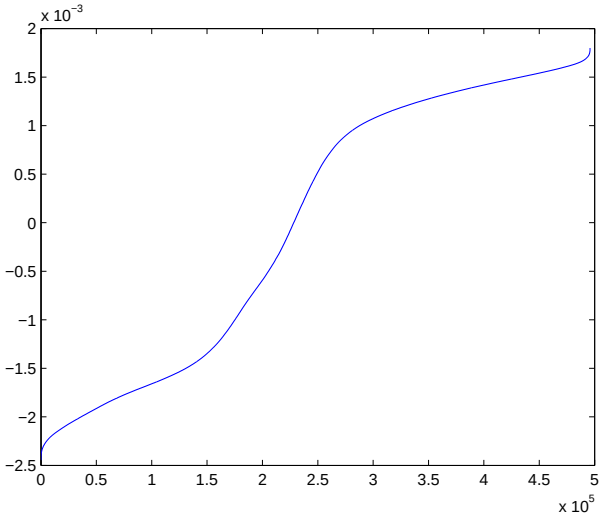
de kleinere blokstructuren binnen de 2 grote blokken. Een voorbeeld hiervan is het blok met fonemen/akoestische eenheden 2 tot en met 7.

Deze structuur doet het vermoeden rijzen dat er een hiërarchie bestaat tussen de clusters die de fonemen/akoestische eenheden modelleren. In de taalwetenschap bestaan er dergelijke hiërarchische structuren van de fonemen. Vooreerst is er het onderscheid tussen stilte en spraak. Spraak wordt dan verder opgedeeld in klinkers en medeklinkers. In figuur 3.4 is de hiërarchie gegeven van de fonemen. Bemerk echter dat dit niet de enige mogelijke hiërarchie is. Zo kunnen sonorante medeklinkers en klinkers samengenomen worden en een aparte categorie vormen t.o.v de obstruenten, de klinkers kunnen worden ingedeeld volgens hun articulatieplaats (voor of achter) i.p.v het onderscheid tussen open en gesloten.

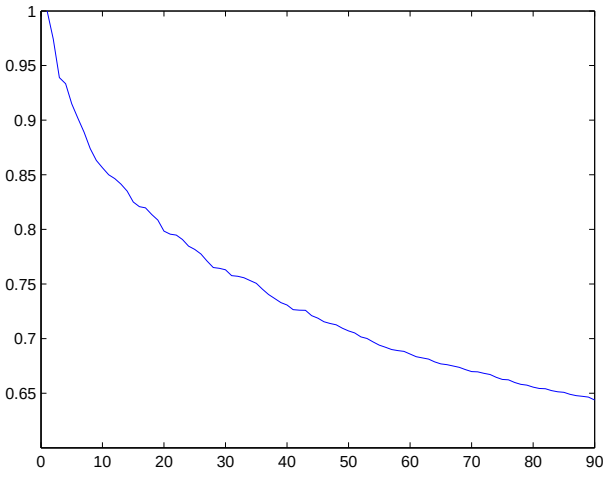
Indien er een hiërarchie aanwezig is in de data, kan de techniek van hiërarchisch clusteren gebruikt worden. Die mogelijkheid kan getest worden aan de hand van de tweede grootste eigenvector van de laplaciaan. Indien deze een grote stap vertoont tussen de hoge en de lage waardes, is het probleem wellicht geschikt om in 2 clusters onderverdeeld te worden. In figuur 3.5 zijn de waardes in deze eigenvector gesorteerd weergegeven. Hoewel er een onderscheid lijkt te bestaan tussen hoge en lage waardes, is het verloop van de scheidingslijn veel te zacht om een correcte partitie te kunnen maken.

Optimaal aantal clusters

Uit de eigenwaardenontbinding kan nog een andere interessante eigenschap worden gehaald, nl. wat het optimale aantal clusters is om de datapunten in te verdelen. Deze is te zien als de ‘eigengap’, d.i. een sprong tussen opeenvolgende eigenwaarden. In figuur 3.6 zijn de eigenwaarden van de laplaciaan weergegeven. Er is echter geen eigengap op te zien. Dit wijst erop dat de clusters niet zeer sterk gescheiden liggen, maar dat er veel overgangsgebieden zijn. In de spraak is dit het geval omdat de articulatoren tijd nodig hebben om te verwissen van stand. Daarom zal het vaak voorkomen dat een punt tussen 2 fonemen of akoestische eenheden



Figuur 3.5: Verdeling van de waarden in de eigenvector horend bij de tweede grootste eigenwaarden van de laplaciaan



Figuur 3.6: Eigenwaarden van de laplaciaanmatrix

in ligt. Door het ontbreken van de eigengap is het moeilijk een optimaal aantal clusters te berekenen.

3.4 Case-studie van ‘because’, ‘nineteen’ en ‘double’

Om een beter zicht te krijgen op de karakteristieken van de gevonden akoestische eenheden, wordt hier de decompositie van 2 woorden meer in detail bekeken. De woorden zijn ‘because’, ‘nineteen’ en ‘double’. Ze zijn gekozen omdat ze bestaan uit fonemen die behoren tot een

brede waaier aan foneemklassen en omdat ze vaak herhaald worden in de uitingen. Een goed middel om de karakteristieken van de gesproken uiting te bestuderen is het spectrogram. Het spectrogram geeft immers de energie weer op verschillende tijdstippen en voor verschillende frequenties. De tijds- en frequentieresolutie moet tegen elkaar afgewogen worden, aangezien een verfijning in het ene domein een smoothing van het andere domein impliceert.

3.4.1 Because

Het Engelse woord 'because' heeft als fonetische transcriptie / b I k @! z / en komt 15 maal voor in de testdata. Op basis van de fonetische transcriptie kunnen de segmenten waarin dit woord geuit wordt geselecteerd worden. Van één zo'n uiting is het spectrogram berekend, en is te vinden in figuur 3.9. In deze figuur staan ook de segmentatie in fonemen, het overeenkomstige deel van de H-matrix en de segmentatie in akoestische eenheden. De verschillende grafieken zijn onder elkaar uitgelijnd, zodat op elk tijdstip kan vergeleken worden wat er in deze 4 grafieken gebeurt. Met de hulp van de segmentatie kan het spectrogram nu geanalyseerd worden. Voor elk foneem worden nu de eigenschappen geanalyseerd aan de hand het spectrogram en wordt de gelijktijdige akoestische eenheid besproken.

- De uiting begint met een pauze. Het 'foneem' dat hiermee overeenkomt is stilte (#). De kenmerken van stilte zijn nogal triviaal: er is geen spraak, dus is er weinig energie in het signaal. De energie die nog voorkomt is het signaal is afkomstig van ruis. Het spectrogram bevat op die plaats dus hele lage waarden omdat er weinig energie is in het signaal. Toch kunnen in het spectrogram bepaalde frequentiebanden met relatief hogere energie onderscheiden worden. Er zijn m.a.w. formanten aanwezig in het spectrum. Deze formanten wijzen erop dat de spreker niet helemaal stil is, maar aanzet geeft voor de volgende klank.

  In de matrix H is dit gedeelte van de matrix vrijwel volledig spaars: enkel de akoestische eenheid 43 heeft significante waarden in de matrix. Dit wijst erop dat het algoritme met grotere zekerheid bepaalt dat dit de aanwezige akoestische eenheid is. Dit is dan ook de akoestische eenheid die bekomen wordt bij de segmentatie. Het is dan ook logisch te besluiten dat deze akoestische eenheid stilte modelleert. De mapping van figuur 3.3 bevestigt deze conclusie: eenheid 43 is een van de akoestische eenheden die mapt op het stiltefoneem. De conclusie is dat de gevonden akoestische eenheden goed het onderscheid maken tussen spraakfragmenten en pauzes tussen de woorden in.

- Het volgende foneem is de /b/-klank. Deze klank wordt gecatalogeerd onder de plosieven. De productie van plosieven verloopt in 2 fasen: in een eerste fase wordt het vocaal kanaal volledig geblokkeerd, zodat er geen lucht meer kan ontsnappen. Bij een /b/-klank zijn de articulatoren de lippen: deze versperren de luchtstroom. In de tweede fase bewegen deze articulatoren snel uit elkaar weg. De lucht ontsnapt nu met een explosieve burst door de ontstane opening. Deze fase wordt de release genoemd. Deze heeft veel energie in vrijwel alle frequenties. In het spectrogram kan dit heel goed worden waargenomen: op tijdstip 0.2sec vindt de stilte-fase plaats, op tijdstip 0.3sec worden de lippen geopend en ontstaat de burst. Deze burstfase is van heel korte duur: onmiddellijk erna worden weer formanten in het spectrogram waargenomen, wat wijst op een klinker of sonorante

medeklinker. De duur van deze fase wordt aangeduid als de voice onset time (VOT).

Deze karakteristiek van de plosieven betekent een inherente moeilijkheid voor de clustering in akoestische eenheden. Hoe kunnen de twee fasen die zo sterk verschillen (nagenoeg geen energie tegenover een explosie van energie) door één en dezelfde akoestische eenheid verklaard worden? Het antwoord daarop is dat dit niet gebeurt. Er worden verschillende akoestische eenheden toegekend voor elke fase. De stilte-fase wordt bij de akoetische eenheden ook daadwerkelijk gezien als stilte: de eenheden die in segmentatie gevonden worden zijn 51 en 44, twee eenheden die op stilte mappen. Aan burst-fase worden eenheden 36 en 33 toegekend. Eenheid 36 modelleert inderdaad een /b/, maar ook een /p/ en soms ook andere plosieven. De reden hiervoor is dat het verschil in akoestische karakteristieken tussen plosieven heel klein is. Informatie over de plaats van articulatie (/t/ vs. /p/ vs. /k/ enz.) moet gehaald worden uit de de overgang van en naar de stilte-fase. Het verschil tussen een stemloze plosief en zijn stemhebbende variant is zelfs nog moeilijker te detecteren. Dit verschil hangt onder meer af van de voice onset time en de duur van de stilte-fase. Aangezien de clustering geen informatie bevat over naburige frames (afgezien van de afgeleiden bij het berekenen van de features), is het heel moeilijk om de plosieven op een correcte manier in verschillende clusters op te delen.

- Deze formanten zijn afkomstig van het volgende foneem: de /I/. Klinkers en sonorante medeklinkers (nasalen en approximanten) hebben als kenmerkende eigenschap dat ze in grote mate periodisch zijn. De oorzaak hiervan is de bron van het spraaksignaal: de stembanden. De stembanden vibreren en moduleren zo de luchtstroom komende uit de longen. Deze klanken bevatten steeds een grondfrequentie (de toonhoogte waarop de spreker praat), en de harmonischen hiervan. Maar luisteraars nemen niet zozeer deze frequenties waar, maar vooral de filtering die dit geluidssignaal ondergaat door de passage in het stemkanaal. Klinkers worden bepaald door de stand van de tong en andere articulatoren, wat zich in het frequentiedomein laat merken door resonantiefrequenties. Deze worden *formanten* genoemd. Vooral de twee laagste formanten (F1 en F2) bepalen welke klinker waargenomen wordt. De relatie tussen F1, F2 en de waargenomen klinker is weergegeven in figuur 3.8

  De gedetecteerde akoestische eenheden zijn hier 3 en 6. Deze eenheden mappen op de fonemen /I/,/I!/,/E/ en /e+I/. De oorzaak waarom verwarring kan ontstaan is hierboven gegeven: deze fonemen liggen dicht bij elkaar in de akoestische ruimte. Op figuur 3.8 ziet men dat de formanten voor deze fonemen ongeveer gelijk zijn. De clustering heeft een onderscheid kunnen maken tussen de klinkers die ver uit elkaar liggen: /i/ vs /u/ en /@/, maar vindt moeilijk een onderscheid tussen klinkers die dicht bij elkaar liggen, zoals hier het geval is. Deze worden in een cluster ondergebracht, die dan gesplitst wordt (er zijn meerdere akoestische eenheden voor deze groep fonemen), maar deze splitsing komt niet overeen met de opdeling in fonemen.
  Een vraag die dit oproept is: heeft het clusteralgoritme dan een verkeerde splitsing berekend, of is de gevonden oplossing wel degelijk degene die het beste clustert, maar liggen de fonemen verspreid? Een aanwijzing naar het antwoord wordt gegeven door te kijken naar welke de dichtste buren zijn die in dat frame berekend werden. Deze zijn weergegeven in tabel 3.3. Het is alleszins merkwaardig dat de dichtste buren van de punten vooral van het foneem /i/ blijken te zijn, terwijl deze punten zelf fonemen /I/

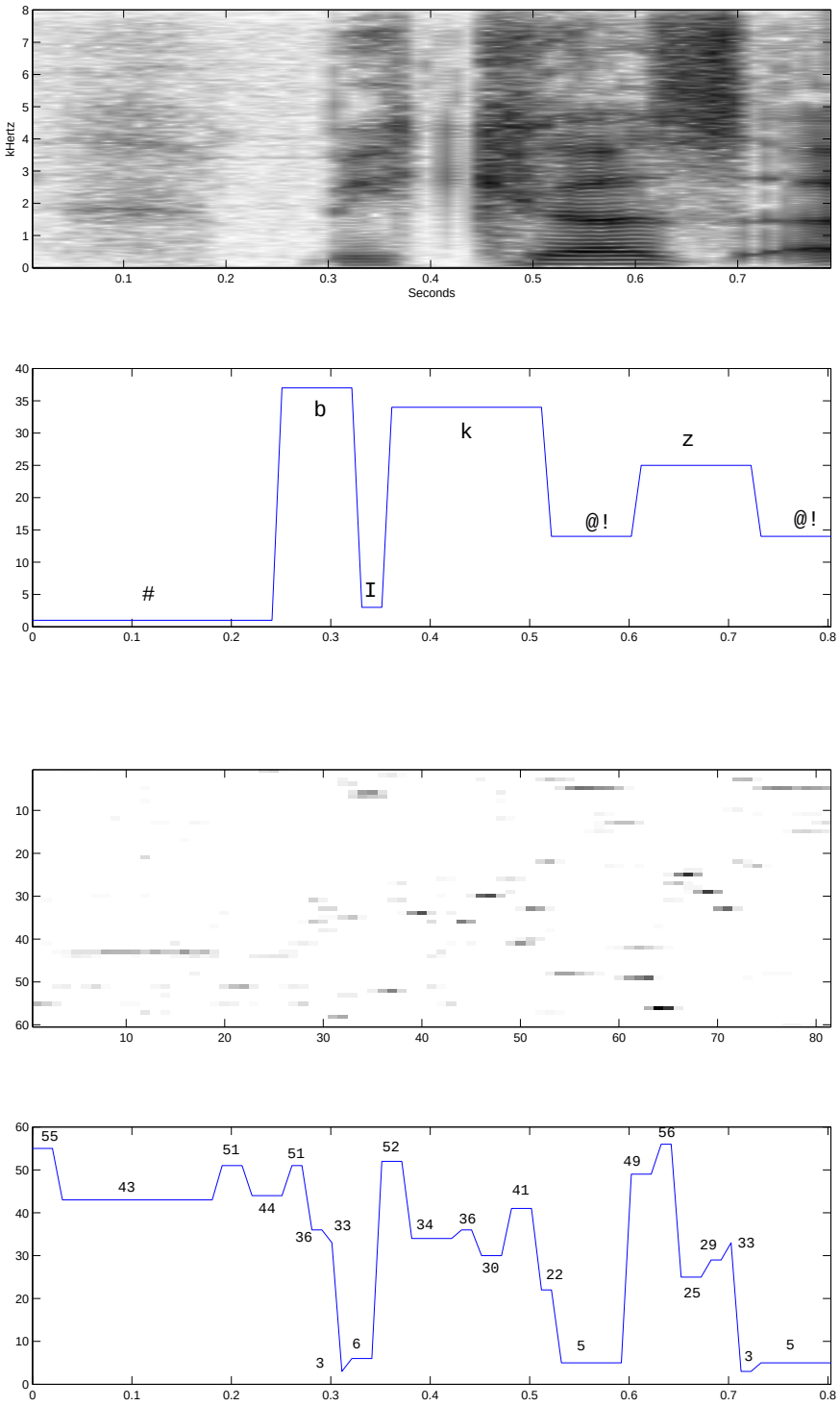
zijn. De punten hebben veel minder buren van hun eigen categorie dan dat ze buren hebben van /i/. Dit kan twee dingen betekenen: ofwel is de segmentatie hier verkeerd, en werd er in de uiting een /i/ uitgesproken, ofwel liggen de fonemen /i/ en /I/ door elkaar in de featureruimte. Een ander opmerkelijk resultaat is het volgende. Het eerste punt heeft 42 buren die een /b/ zijn, de volgende punten hebben steeds minder buren /b/. Dit komt omdat /b/ het foneem is dat net voor het eerste punt uitgesproken werd, en de release-fase is nog bezig. Het punt in kwestie is dus nog iets tussen een /b/ en en /i/ in. De spraakproductie is een continu proces dat geleidelijk overgaat van de ene stand naar de andere. Er zijn dus altijd frames die tussen 2 fonemen in zitten zoals hier.

- Het volgende foneem is opnieuw een plosief, nl. de /k/. In het spectrogram is opnieuw heel duidelijk zichtbaar wanneer de stilte-fase plaatsvindt (0.39sec tot 0.44sec) en wanneer de burst plaatsvindt (0.44sec). Hier is er echter nog een derde fase: de aspiratiefase. Net na de burst wordt er een ademstoot gegeven. In het Engels is dit heel gebruikelijk wanneer plosief stemloos is en gevolgd wordt door een beklemtoonde klinker, wat hier het geval is. Bij de segmentatie met akoestische eenheden is het opmerkelijk dat de stilte-fase hier een akoestische eenheid heeft die overeenkomt met het foneem k, en niet met een stiltefoneem.
- De beklemtoonde klinker waarvan sprake is, is het foneem /@!/. In het spectrogram zijn heel duidelijk de verschillende formanten waar te nemen. De akoestische eenheid is 5, wat gemapt wordt op een /E/ i.p.v /@!/ De oorzaak is opnieuw dat de fonemen dicht in elkaars buurt liggen, wat blijkt uit figuur 3.8.
- Het laatste foneem is de /z/. Deze medeklinker wordt gecatalogeerd in een foneemcategorie die nog niet besproken is: de fricatieven. Bij fricatieven (of wijfklanken) wordt de klank geproduceerd door het stemkanaal zodanig te vervormen dat er ruis ontstaat. Dit ruis wordt ofwel veroorzaakt omdat lucht tegen een obstakel geblazen wordt, ofwel omdat het kanaal zodanig wordt vernauwd, dat er ruis ontstaat wanneer de lucht dit nauwe kanaal verlaat (vergelijk dit met het lossen van een fietsband). De /z/-klank is een voorbeeld van de eerste soort: de lucht wordt tegen de tanden gestuwd. Wat de bron van de ruis ook is, er zal altijd turbulentie optreden van de lucht in het kanaal. Fricatieven worden daarom in het spectrogram waargenomen als hoog-frequent ruis. In dit voorbeeld is dit heel duidelijk merkbaar in het spectrogram. Merk op dat /z/ een stemhebbend fricatief is, dit is te zien in het spectrogram aangezien er in de lagere frequenties vage formanten voorkomen.
  De akoestische eenheden zijn hier 49, 56, 25, 29 en 33. Enkel eenheid 25 wordt op z gemapt. Toch mappen de andere op gelijkaardige klanken: /s/, /S/ en /d+Z/.Dit wijst erop dat de clustering wel sisklanken kan herkennen, maar niet het verschil tussen deze vier fonemen.

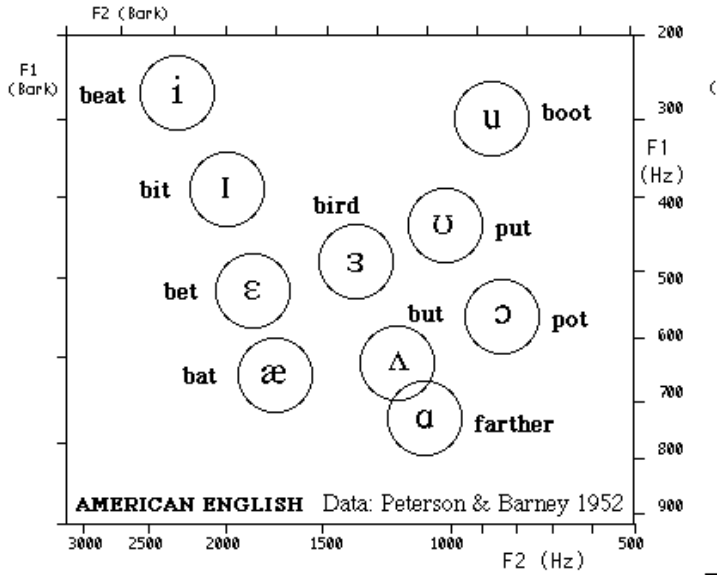
3.4.2 Nineteen

Het Engelse woord ‘nineteen’ heeft als fonetische transcriptie /n a+I n t i n /. Het woord komt 36 keer voor in de testdata. Hier wordt enkel het foneem /n/ geanalyseerd aangezien de andere fonemen tot categorieën behoren die reeds besproken zijn.

Het foneem /n/ is een nasale medeklinker. Voor het produceren van nasale medeklinkers wordt er een opening gecreëerd naar de neusholte. De mondholte wordt afgesloten door de



Figuur 3.7: Spectrogram en segmentaties van ’because’



Figuur 3.8: Gemiddelde formantwaarden van 10 Amerikaanse klinkers voor 33 mannelijke sprekers. Bron: `http://www.helsinki.fi/speechsciences/projects/vowelcharts`. Data van G. E. Peterson & H.L.Barney (1952) Control methods used in a study of the vowels. Journal of the Acoustical Society of America, 24, 175-184

.

foneem	punt 1	punt 2	punt 3	punt 4
b	42	8	0	0
I!	5	11	9	7
I	20	44	35	32
i	77	140	141	121
j	2	14	21	10
E	2	1	1	1
e+I	2	11	17	27
O+I	0	1	2	1
a+I	0	0	5	6
n	8	2	4	14
m	15	6	4	2
N	0	0	3	29
r	18	1	1	0
@	10	6	1	0
D	15	1	0	0
rest	40	10	12	6

Tabel 3.3: Fonemen van de 256-dichtste buren van de punten met akoestische eenheid 3 en 6 in figuur 3.7

tong, zodat de lucht hier niet kan ontsnappen, maar zijn weg zoekt langs de neusholte. De mondholte gedraagt zich als een zijdelingse resonantieruimte. Deze resonantie verbruikt energie en introduceert bijgevolg nulpunten in het frequentiedomein. Aangezien de lucht nu door de neusholte stroomt, is er een grotere oppervlakte die het geluid dempt. Deze twee karakteristieken maken dat het frequentiespectrum heel lage amplitudes vertoont. Aangezien er vrije uitstroom is van lucht (via de neus), zijn er ook formanten aanwezig in het frequentiespectrum. In het spectrogram kan duidelijk gezien worden dat de /n/ optreedt tussen 0.05 en en 0.1 seconden: er is een sterke vermindering van de amplitude, maar de formanten blijven aanwezig.

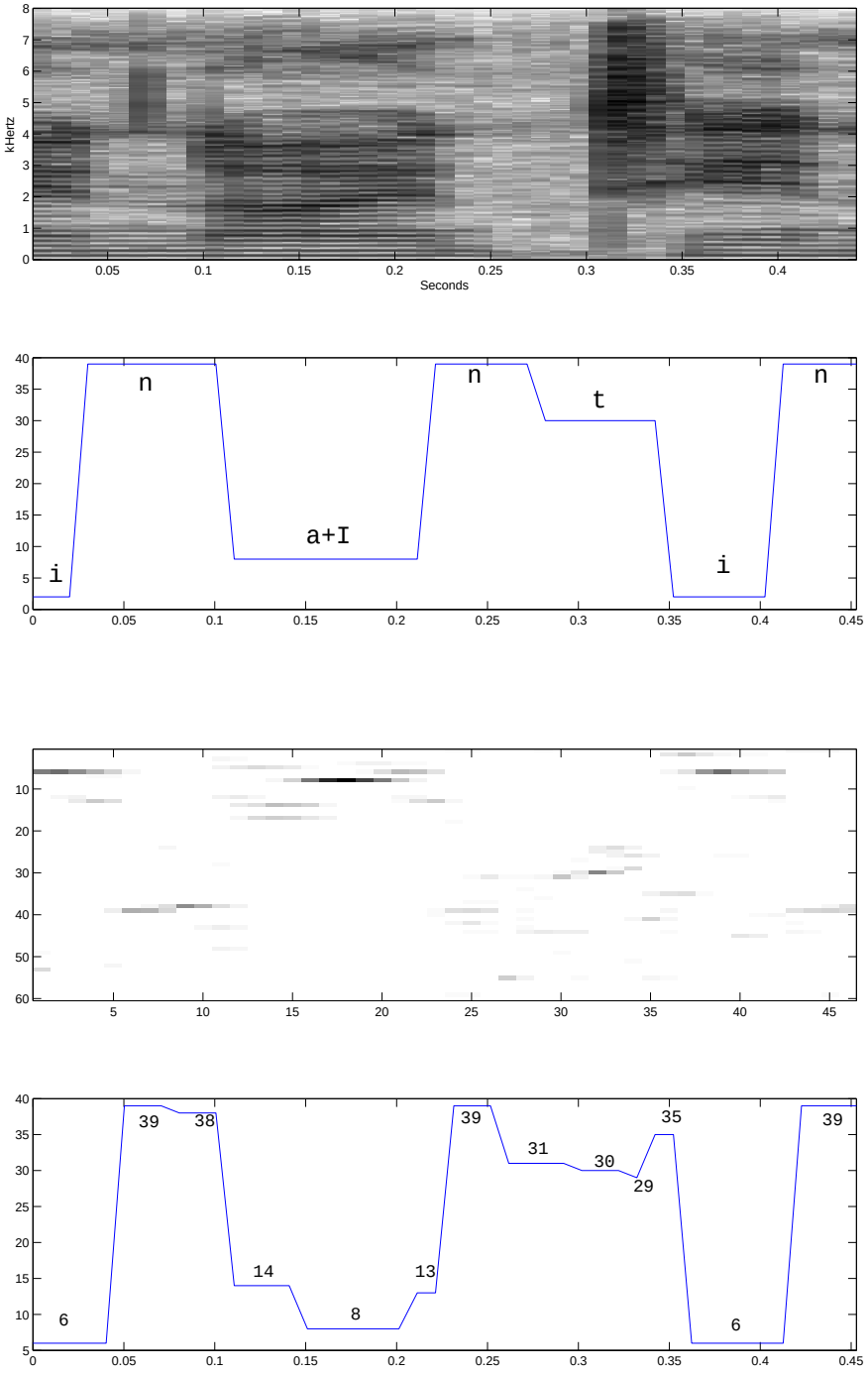
Het deel van de matrix H dat met het foneem /n/ overeenkomt is spaars, wat erop wijst dat foneem duidelijk onderscheiden wordt van de rest. De akoestische eenheden zijn 38 en 3, wat resp. gemapt wordt op /m/ en /n/. Ook hier is dit hetzelfde verhaal: de clustering kan nasale medeklinkers scheiden van de rest, maar binnen de groep wordt weining verschil gezien.

Het tweede deel van de case-studie van het woord nineteen is een globale analyse van de verschillende akoestische eenheden die gedetecteerd worden in de uitingen van dit woord. De voorkomende akoestische eenheden zijn weergegeven in tabel 3.4. Er is ook aangegeven hoe vaak - gemeten in aantal frames - deze akoestische eenheden gedetecteerd zijn. Daarnaast wordt een indicatie gegeven van de correlatie van deze akoestische eenheden met de fonemen van het woord nineteen: /n/, /a+I/, /t/ en /i/. Dit laatste wordt gerealiseerd door de de co-occurences te tellen van de verschillende akoestische eenheden met de verschillende fonemen.

Het blijkt dat sommige akoestische eenheden grote correlaties vertonen met bepaalde fonemen. Zo komt akoestische eenheid 39 in 97% van de frames overeen met het foneem /n/. Dit is een heel sterke aanwijzing dat akoestische eenheid 39 in deze context het foneem /n/ modelleert. Andere sterke correlaties zijn: eenheid 8 komt in 99% van de frames overeen met foneem /a+I/, eenheid 38 heeft 95% overeenkomst met foneem /n/, en tenslotte eenheid 30 heeft volledige overeenkomst met foneem /t/. Er is geen één-op-één-mapping: het foneem /n/ wordt bijvoorbeeld verklaard door zowel eenheid 38 als 39.

Een ander opvallend feit is dat sommige akoestische eenheden sterke correlaties vertonen met twee fonemen. Een goed voorbeeld hiervan is de akoestische eenheid 31: deze komt voor 60% overeen met het foneem /n/, en voor 40% met het foneem /t/. De oorzaak hiervan is dat deze eenheid een overgangseenheid is tussen fonemen /n/ en /t/. Bij de articulatie van deze twee fonemen moet de mond immers overgaan van een (stemhebbende) nasale klank naar een (stemloze) plosieve klank. De akoestische eenheid 31 heeft karakteristieken die liggen tussen deze van fonemen /n/ en /t/. Een ander voorbeeld is akoestische eenheid 35, deze vormt een overgangseenheid tussen de plosief /t/ en de klinker /i/.

Een derde observatie is dat er heel wat voorkomende akoestische eenheden slechts zelden worden gedetecteerd. Hiervoor zijn twee verklaringen. Een eerste verklaring is dat er ruis is op de data of afwijkingen optreden in de clustering. Dit laatste kan voorkomen aangezien de clustering geen garantie biedt op de optimale oplossing. De tweede verklaring ligt bij de spreker zelf. Als deze het foneem op een licht afwijkende wijze heeft uitgesproken, zullen de akoestische karakteristieken op het tijdstip van het foneem afwijken van de normale (verwachte) karakteristieken. Hierdoor zullen deze frames een akoestische eenheid toegewezen krijgen, die wel overeenkomt met de akoestische karakteristieken, maar niet met het bedoelde foneem. Een menselijke waarnemer hoort meestal wel het bedoelde foneem. Dit is een effect dat gecreëerd wordt door de hersenen en die een betere verstaanbaarheid tot gevolg heeft. Bij de clustering is



Figuur 3.9: Spectrogram en segmentaties van 'nineteen'

akoestische eenheid	totaal aantal vookomens	co-occurrence met /n/	co-occurrence met /a+I/	co-occurrence met /i/	co-occurrence met /t/
39	216	209	0	6	1
8	167	1	166	0	0
6	159	70	44	40	5
38	133	127	6	0	0
30	116	0	0	0	116
2	110	21	3	80	6
35	102	3	0	84	15
31	92	55	0	0	37
12	50	17	33	0	0
29	45	2	0	1	42
26	37	0	0	1	36
45	36	13	18	5	0
52	33	31	2	0	0
34	33	17	0	0	16
5	31	5	26	0	0
4	27	11	13	3	0
13	25	18	7	0	0
7	21	5	0	16	0
37	21	21	0	0	0
14	18	3	15	0	0
40	18	8	0	10	0
17	17	0	17	0	0
36	16	4	0	0	12
24	12	0	0	0	12
41	10	0	0	1	9
42	9	9	0	0	0
3	8	7	1	0	0
11	7	0	7	0	0
51	7	3	0	0	4
28	6	0	0	0	6
33	6	3	0	2	1
55	4	1	0	0	3
25	3	0	0	0	3
58	3	0	3	0	0
1	2	0	0	0	2
10	2	2	0	0	0
21	1	1	0	0	0
44	1	1	0	0	0

Tabel 3.4: Voorkomende akoestische eenheden in de uitingen van het woord ‘nineteen’. De tweede kolom bevat het totaal aantal frames waarin de akoestische eenheid voorkomt. De volgende kolommen bevatten de co-occurences (gemeten in aantal frames) van de voorkomende akoestische eenheden met fonemen.

geen informatie ingebouwd over het verwachte foneem, en zal dus een afwijkende akoestische eenheid genereren.

3.4.3 Double

Het woord double bestaat uit vijf fonemen: / d @! b @ l /. Het woord komt 32 keer voor in de testdata. Op dit woord wordt een globale analyse uitgevoerd van de akoestische eenheden. Hier focust de studie niet op de gedetecteerde eenheden afzonderlijk, maar wel op de gedetecteerde sequenties van akoestische eenheden. Het resultaat van deze studie is dat er heel wat overeenkomsten zijn tussen de verschillende sequenties. Slechts zelden komen volledig gelijke sequenties voor, maar heel vaak komen delen van de sequentie overeen. Hierdoor is het mogelijk een model op te stellen van de opeenvolging van akoestische eenheden in het woord double. Dit is weergegeven in figuur 3.10. In dit grafenmodel stellen de knopen een gedetecteerde akoestische eenheid voor en de bogen geven de opeenvolging tussen deze eenheden weer. Figuur 3.11 geeft een vereenvoudigde versie van dit model weer. Dit laatste is berekend door het weglaten van de verbindingen die slechts éénmaal voorkomen in de gesegmenteerde uitingen en van de knopen die na deze bewerking losgekoppeld zijn van het netwerk. Het aantal voorkomens van de opeenvolgingen van twee akoestische eenheden is weergegeven bij de corresponderende boog in het vereenvoudigde model.

Dit model geeft nieuwe inzichten in de segmentatie in akoestische eenheden. Ten eerste is het heel opvallend dat het berekende model zo compact is. Indien de segmentatie van woorden in akoestische eenheden totaal willekeurig zou zijn, dus statistisch onafhankelijk van het woord in kwestie, zou een model voor deze segmentaties veel complexer zijn. Elke uiting van het woord in de dataset zou immers een eigen pad van segmentaties vormen binnen het model. Hierbij zouden sommige knopen gemeenschappelijk zijn, maar dit zou enkel op toeval berusten. Het is heel duidelijk dat het model dat hier bekomen is veel compacter is. Dit wijst er dus op dat de segmentatie van een woord in akoestische eenheden niet willekeurig is, maar een bepaald patroon volgt. Dit patroon kan voorgesteld worden door een model. Het model van figuur 3.10 is een schatting voor het werkelijke model van het woord double in de ruimte van akoestische eenheden.

Een tweede opmerking is dat de uitgevoerde vereenvoudiging op het model een heel groot effect heeft. Het vereenvoudigde model is nog veel compacter dan het oorspronkelijke model. Dit wijst erop dat vele akoestische eenheden uitschieters zijn, die wellicht foutief gegenereerd zijn. Deze fouten kunnen veroorzaakt zijn door ruis op de data of omdat de clustering niet het globaal beste resultaat bereikt.

Een derde opmerking is dat er een pad bestaat in het model dat de hoogste waarschijnlijkheid heeft. Dit pad is in figuur 3.11 aangeduid met de pijlen in het vet. Het blijkt dat de meeste delen van dit pad een veel grotere waarschijnlijkheid hebben dan andere verbindingen in het netwerk. Dit wijst erop dat de woorden een vaste sequentie hebben van akoestische eenheden, maar dat bepaalde alternatieven kunnen voorkomen. Redenen voor alternatieve paden van akoestische eenheden zijn een licht verschillende uitspraak van bepaalde fonemen, een snellere uitspraak van het woord, zodat bepaalde overgangseenheden niet gedetecteerd worden, gebruik van allofonen, enz...

Deze analyse toont aan dat woorden een bepaald patroon vormen in de ruimte van akoestische

eenheden. Het beschreven algoritme vindt voor elke uiting van een woord een instantiatie van het patroon horend bij dat woord. Een model voor het patroon van een woord kan geschat worden op basis van een reeks segmentaties van de uitingen van dat woord. Een voorbeeld van zo'n geschat model is voorgesteld in figuur 3.11. Dit model kan dan gebruikt worden om woorden te herkennen op basis van de segmentatie in akoestische eenheden.

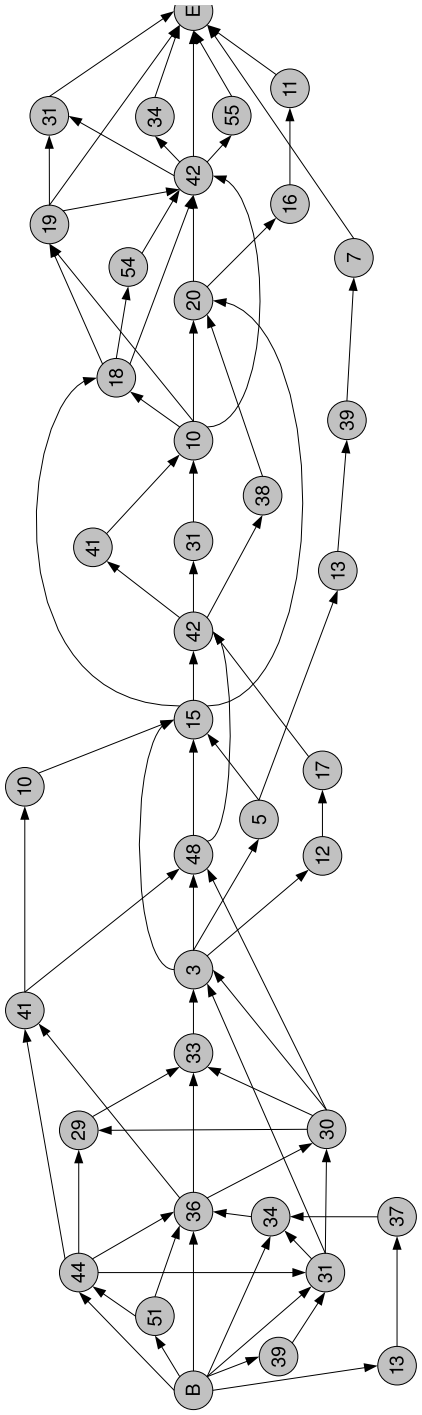
3.5 Besluit

In dit hoofdstuk werd nagagaan of een computer in staat is de akoestische basiseenheden te vinden van een taal, zonder daarbij gebruik te maken van expertkennis. Uit de experimenten bleek dat dit inderdaad bereikt werd. Het clusteralgoritme beschreven in sectie 2.3 werd uitgebreid met een Viterbi-oplijning. Beide algoritmes maken geen gebruik van kennis uit de fonologie of fonetiek, noch van enige andere kennis over de taal. Beide algoritmes maken ook geen gebruik van enige vorm van inbreng van de gebruiker, zodat het geheel unsupervised is. Het resultaat is een segmentatie van de uitingen in akoestische eenheden. Deze akoestische eenheden vormen de basiseenheden waaruit de uitingen in de bepaalde taal (in dit geval het Engels) zijn opgebouwd. Deze basiseenheden worden gezocht en gevonden in de akoestische karakteristieken van de uitingen.

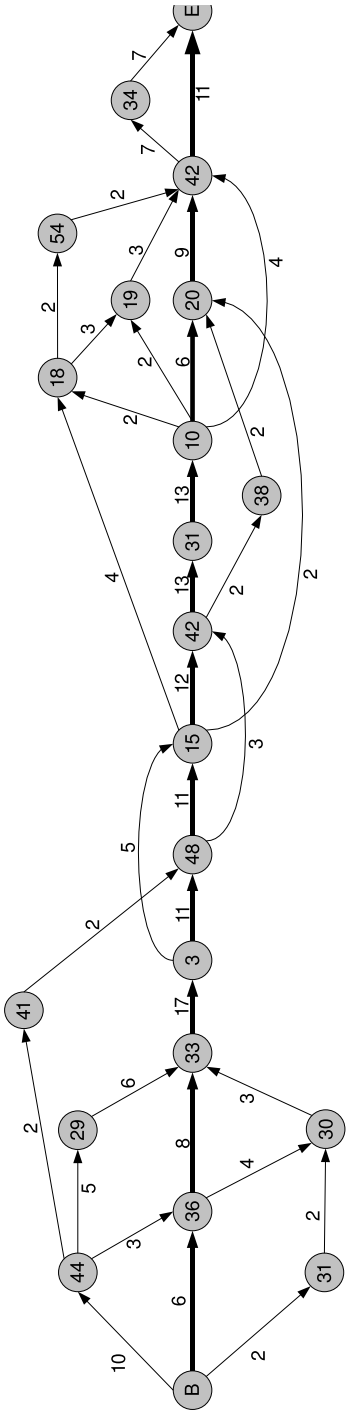
In een volgende stap werd nagegaan wat het verband is tussen deze gevonden basiseenheden en de taalkundige basiseenheden, nl. de fonemen. Uit de berekende mappingen blijkt dat er sterke verbanden bestaan tussen de gevonden akoestische eenheden en de fonemen. Er zijn echter ook een aantal verschillen. Deze hebben drie oorzaken. De eerste oorzaak is te vinden in het model van het clusteralgoritme: dit gaat op zoek naar gebalanceerde clusters, terwijl fonemen vaak ongebalanceerd zijn. Het gevolg is dat bepaalde fonemen op meerdere akoestische eenheden mappen en omgekeerd. Een voorbeeld hiervan is het stiltefoneem, dat in veel meer frames voorkomt dan een om het even welk ander foneem. Dit wordt bijgevolg op meerdere akoestische eenheden gemapt. Een tweede oorzaak is dat het clusteralgoritme niet perfect is, en dat er ruis aanwezig is op de data, zodat de bekomen segmentatie niet de optimale segmentatie is in akoestische eenheden. Hierdoor zullen bepaalde frames een akoestische eenheid toegewezen krijgen die afwijkt van de best passende akoestische eenheid. Een derde oorzaak is dat de overeenkomsten tussen fonemen en akoestische eenheden zich eerder bevinden op een niveau hoger. Vaak vertoont een bepaalde foneemcategorie (bijv. sisklanken, plosieven, voorklinkers, achterklinkers, nasalen, ...) een sterke correlatie met één bepaalde set van akoestische eenheden. De fonemen binnen deze categorie zullen echter correlaties vertonen met meerdere akoestische eenheden binnen de set. Dit wordt veroorzaakt omdat de akoestische karakteristieken van fonemen binnen een categorie zeer sterk gelijkend zijn. Het onderscheid tussen deze fonemen is dan eerder op basis van eigenschappen die niet rechtstreeks af te leiden zijn uit de akoestische karakteristieken. Akoestische eenheden maken het onderscheid wel enkel en alleen op basis van deze akoestische karakteristieken. De opdeling van akoestische eenheden van een set die overeenkomt met een bepaalde foneemcategorie zal bijgevolg anders zijn dan de opdeling in fonemen binnen die categorie.

Hoewel deze akoestische eenheden niet volledig overeenkomen met fonemen, en dus geen vervanging kunnen vormen voor foneemherkenning, zijn akoestische eenheden wel nuttig voor verdere woordherkenning. Uit de casestudie bleek immers dat woorden een bepaald patroon

vormen in de ruimte van akoestische eenheden. Voor elke uiting van een woord wordt een instantiatie van dat woordpatroon gedetecteerd. Een model voor het woordpatroon kan geschat worden op basis van een reeks segmentaties van de uitingen van dat woord. Dit model kan dan gebruikt worden om woorden te herkennen op basis van de segmentatie in akoestische eenheden. De akoestische eenheden vormen dus bouwblokken waarmee verdere woordherkenning kan uitgevoerd worden.



Figuur 3.10: Model van de opeenvolging van akoestische eenheden in het woord double



Figuur 3.11: Vereenvoudigd model van de opeenvolging van akoestische eenheden in het woord double

Hoofdstuk 4

Woordmodellen en woordherkenning

4.1 Inleiding

De kleinste eenheden in een taal die een betekenis hebben, zijn de woorden. Om spraakuitingen te begrijpen is het dan ook heel belangrijk de woorden te detecteren en te herkennen. De mens heeft bij zijn geboorte nog geen kennis van deze woorden, maar bouwt gedurende zijn leven een woordenschat op van om en bij de 50.0000 woorden. De meeste woorden leert hij in zijn kinderjaren: hij analyseert de geluidsdata die op hem af komt en haalt er de terugkerende patronen uit. In dit hoofdstuk wordt dit proces nagebootst. Dit verloopt in 2 fases. In de trainingfase worden woorden aangeleerd aan de herkenner. Het doel is hier nieuwe patronen te zoeken die overeenkomen met woorden. De tweede fase is de test-fase. Hier maakt de herkenner gebruik van de woordmodellen die in de trainingsfase geleerd zijn. Met deze kennis moet de herkenner in staat zijn te detecteren of een gekend woord voorkomt in een uiting ja dan neen. Deze woordherkenning bouwt voort op de gevonden akoestische eenheden. Zo kan de woordherkenning volledig zelflerend worden uitgevoerd.

4.2 Ontdekken van woordmodellen

4.2.1 Probleemstelling

Wanneer baby's en peuters hun eerste woordjes leren, komt er veel non-verbale communicatie aan te pas. Een moeder neemt de bal wanneer ze vraagt: "Spelen met de bal?". Deze aanwijzingen helpen het kind te begrijpen wat er gezegd wordt. Het kind associeert hetgeen aangewezen wordt met het terugkerende akoestische patroon van de uiting. Dit proces wordt in deze experimenten nagebootst. Het algoritme moet een beperkt aantal sleutelwoorden leren. De input van het algoritme zijn uitingen die bestaan uit één van deze sleutelwoorden en daarnaast nog andere woorden, die niet geleerd of herkend moeten worden. Daarbij wordt telkens aangegegeven welk sleutelwoord in de uiting voorkomt. Het doel van het algoritme is dan het zoeken van bepaalde patronen die bij de sleutelwoorden horen.

4.2.2 Voorstelling van de spraakuitingen

Dit algoritme maakt – net zoals het algoritme om akoestische eenheden te ontdekken – gebruik van een niet-negatieve matrixfactorisatie. Het is daarom nodig een matrix V op te stellen met enkel niet-negatieve elementen en die voor elke uiting twee componenten bevat: een voorstelling van de uiting zelf en een aanwijzing van welk sleutelwoord in de uiting voorkomt.

De eerste component maakt gebruik van de segmentatie in akoestische eenheden die in het vorige hoofdstuk beschreven zijn. Met deze segmentatie is het immers mogelijk om een nieuw akoestisch model te trainen. Dit akoestisch model kan dan gebruikt worden om te berekenen wat de kans is op een akoestische eenheid, gegeven de featurevector. De opbouw van de matrix maakt gebruik van de Flavor architectuur [6]. De uitingen worden voorgesteld in de kolommen van V. Voor elke uiting wordt een lattice opgesteld. Voor elke combinatie (ϕ, ψ) van akoestische eenheden wordt een coëfficiënt berekend als volgt. Elke keer dat akoestische eenheid ψ volgt op ϕ, worden de posteriore probabiliteiten van deze twee akoestische eenheden vermenigvuldigd en gewogen met de probabiliteit van de gemeenschappelijke knoop. De coëfficiënt is dan de som van deze producten.

$$c(\phi, \psi) = \sum_{\{\alpha \in E : h(\alpha) = \phi\}} \sum_{\{\beta \in E : h(\beta) = \psi\}} P(\alpha) P(\beta) \Delta_{\alpha\beta} \tag{4.1}$$

$$\Delta_{\alpha\beta} = \begin{cases} \frac{1}{P(e(\alpha))} & \text{als } e(\alpha) = b(\beta) \\ 0 & \text{in het andere geval} \end{cases} \tag{4.2}$$

De gebruikte symbolen zijn hier: E voor de verzameling bogen in de lattice, $h()$, $b()$ en $e()$ geven respectievelijk de akoestische eenheid, de beginknoop en de eindknoop horend bij een boog. Deze coëfficiënten vormen de elementen van de kolom in V die overeenkomt met de uiting in kwestie. De dimensie van de matrix V is dus $R^2 \times U$ met R het aantal akoestische eenheden en U het aantal uitingen.

De matrix V bevat nu de voorstellingen van de uitingen op basis van de gegenereerde lattices. Naast deze component moet de matrix ook nog de aanwijzingen bevatten over welke sleutelwoorden aanwezig zijn in de uiting. Daarom worden er aan de matrix S rijen toegevoegd. S is het aantal sleutelwoorden dat geleerd moet worden. De dimensie van de matrix V is nu $(R^2 + S) \times U$. Het toegevoegde deel van de matrix heet de *tag-matrix*. Elke rij van deze deelmatrix komt overeen met een sleutelwoord, en elke kolom komt overeen met een uiting. De tag-matrix bevat een 1 indien de respectievelijke uiting het respectievelijke sleutelwoord bevat, en een nul indien dit niet het geval is.

4.2.3 NMF-factorisatie

Met deze matrix V kan nu de NMF-factorisatie berekend worden. Zoals beschreven in hoofdstuk 2 is het belangrijk een goede initialisatie te voorzien voor de W- en/of H-matrix. De W-matrix wordt hier geïnitialiseerd als volgt: het onderste gedeelte van de matrix dat overeenkomt met het tag-gedeelte van de V-matrix, wordt geïnitialiseerd als een eenheidsmatrix, waarbij een klein random getal wordt opgeteld. Bij een goede ontbinding zal elke kolom van V immers

overeenkomen met de voorstelling van één sleutelwoord. Dit sleutelwoord heeft in het tag-gedeelte een 1 in de rij corresponderend met zichzelf en een nul bij de andere sleutelwoorden. Via deze initialisatie van de W-matrix wordt dus specifiek gezocht naar basisvectoren die de sleutelwoorden modelleren.

Het resultaat van deze niet-negatieve matrixfactorisatie bevat twee onderdelen. De matrix W bevat de basisvectoren van de nieuwe vectorruimte. In deze context zou dit moeten overeenkomen met woordmodellen voor de sleutelwoorden. De matrix H bevat de indicatoren: het geeft voor elke uiting aan welke basisvectoren van W aanwezig zijn, wat bij een goede oplossing hetzelfde is als het aangeven van welk sleutelwoord voorkomt in de uiting. Indien een echte indicatie nodig is, is het echter beter om de H-matrix eerst links te vermenigvuldigen met het geüpdatete tag-deel van de W-matrix. Zo wordt een betere indicatie bekomen als de matrix W afwijkt van de correcte oplossing.

4.3 Woordherkenning met de geleerde woordmodellen

Ook de woordherkenning gebruikt het NMF-algoritme om de taak te voltooien. De woordherkenning in deze stap maak gebruik van de geleerde woordmodellen uit de trainingsfase. De matrixfactorisatie verloopt als volgt. De V matrix wordt op dezelfde manier berekend als voor de training, maar ditmaal op de testdata. De W-matrix is het resultaat van de trainingfase. Deze matrix wordt niet geüpdatet tijdens het algoritme. De H-matrix is hetgeen gezocht wordt en wordt berekend via het normale NMF-algoritme.

De performantie van de woordherkenning kan nu eenvoudig worden berekend. Eerst wordt het gedeelte van de W-matrix dat overeenkomt met het tag-gedeelte van de V-matrix vermenigvuldigd met de H-matrix. Nu bevatten de kolommen van H een indicatie van welke sleutelwoorden er in de testuitingen voorkomen. Uit elke kolom van H wordt het grootste element geselecteerd. Indien het werkelijke sleutelwoord overeenkomt met de rij-index van dat geselecteerde element, dan is de herkenning juist gebeurd. Indien niet, dan faalt de woordherkenning. De performantie van de woordherkenning is de som van alle juiste herkenningen voor elke uiting gedeeld door het aantal uitingen. De performantie geeft dus aan in hoeveel % van de uitingen een juist woord gedetecteerd werd.

De resultaten van de experimenten zijn te vinden in tabel 4.3. In de tabel staan de foutpercentages $1-performantie$ van de woordherkennig. Voor elke woordbatch wordt de NMF-factorisatie vijfmaal berekend met telkens verschillende initialisaties. Deze random initialisatie leidt tot verschillende performanties.

4.4 Resultaten

Het is duidelijk dat de woordherkenningsexperimenten via akoestische eenheden heel hoge foutpercentages halen. Wanneer deze pecentages vergeleken worden met de woordherkenningsexperimenten via fonemen, dan ziet men dat de herkenning op basis van fonemen heel wat kleinere foutpercentages hebben. Het blijkt ook dat een groter aantal buren een postief effect

aantal ak. eenheden	50	50	100	50	50
aantal buren	255	255	255	511	511
Testbatch	01	02	00	00	01
	27.586207%	29.032258%	28.947368%	18.421053%	17.241379%
	24.137931%	35.483871%	26.315789%	13.157895%	17.241379%
	24.137931%	32.258065%		28.947368%	20.689655%
	20.689655%	35.483871%		31.578947%	13.793103%
	27.586207%	35.483871%		28.947368%	10.344828%

Tabel 4.1: Foutenpercentages van de woordherkenning getraind met akoestische eenheden

Testbatch	00	01	02
	5.263158%	10.344828%	6.451613%
	5.263158%	10.344828%	3.225806%
	5.263158%	6.896552%	
	5.263158%	10.344828%	
	2.631579%	10.344828%	

Tabel 4.2: Foutenpercentages van de woordherkenning getraind met fonemen

heeft op de performantie. Deze experimenten zijn nog maar in het beginstadium van het onderzoek. Er is nog heel wat ruimte voor verbetering.

Hoofdstuk 5

Algemeen besluit

Het doel van dit eindwerk bestond voornamelijk uit het toepassen van clustertechnieken voor het opsporen van akoestische eenheden in spraakuitingen. De gevonden akoestische eenheden kunnen dan aangewend worden voor het herkennen van woorden, zonder dat er kennis in het model wordt ingebouwd over de taal.

In het eerste deel van de thesis werd een geschikte clustertechniek gezocht. Daarbij werd vooral onderzoek gedaan naar clustering met niet-negatieve matrixfactorisatie en spectraal clusteren. Niet-negatieve matrixfactorisatie heeft het nadeel dat het geen globaal optimum vindt. Het is daarom sterk afhankelijk van de initialisatie. Spectraal clusteren begint met een eigenwaarde-probleem. Hier wordt wel geconvergeerd naar een globaal optimimum. Voor K-way clustering moet hierna echter K-means toegepast worden, dat op zich naar een lokaal minimum convergeert en dus ook afhankelijk is van de initialisatie. Er werd een techniek voorgesteld die het globale minimum van spectraal clusteren aangrijpt om een betere initialisatie te hebben voor NMF. In de berekening van de transformatie van de eigenwaarde-matrix naar de initialisatiematrix wordt echter gebruik gemaakt van een algoritme dat geen garantie biedt dat het globale optimum dat met de eigenwaardenontbinding bereikt werd, behouden blijft. De drie technieken NMF, spectraal clusteren en NMF met initialisatie via spectraal clusteren werden getest op een artificieel probleem. Daaruit blijkt dat NMF met random initialisatie slechtere performanties haalt dan de andere technieken, inclusief K-means. Wanneer het NMF-algoritme beter wordt geïnitialiseerd stijgt de performantie spectaculair. Het voorgestelde algoritme, dat NMF initialiseert met spectraal clusteren, bereikt de beste performantie van de onderzochte technieken. In het artificiële probleem werd gemiddeld gezien 98% van de data juist geclusterd.

In het tweede deel was het doel het vinden van akoestische eenheden in spraakuitingen, en na te gaan of er verbanden bestonden tussen deze eenheden en de fonemen. Via clustering met de techniek uit sectie 2.3, werden er eenheden gevonden. Er werden duidelijke overeenkomsten gevonden tussen de akoestische eenheden en de fonemen. Toch kunnen deze eenheden niet eenduidig gemapt worden op de fonemen. Het clusteralgoritme wijst wel aparte akoestische eenheden toe aan de verschillende foneemcategorieën, maar vindt weinig of geen onderscheid binnen zo'n categorie. Oorzaak hiervan ligt meestal in de sterke gelijkenissen tussen verschillende fonemen, zodat het akoestische spectrum op zich niet genoeg is om te weten te komen welk foneem geüit wordt. Hoewel geen eenduidig verband bestaat tussen fonemen en akoestische eenheden, zijn deze laatste wel nuttig aangezien ze bepaalde kenmerken in het spectrum gaan

modelleren. Woorden zijn daarom vaak opgebouwd uit een vast patroon van akoestische eenheden. Akoestische eenheden kunnen daarom beschouwd worden als bouwblokken waarop verdere spraakherkenning kan plaatsvinden.

In het derde deel werden woordherkenningsexperimenten uitgevoerd die de gevonden akoestische eenheden gebruiken als akoestisch model. Met behulp van niet-negatieve matrixfactorisatie werden woordmodellen geleerd voor een beperkte set van sleutelwoorden. Het doel was om de geleerde modellen voor deze sleutelwoorden te gebruiken om te detecteren of een sleutelwoord voorkomt in een uiting. De eerste resultaten van deze experimenten bleken nog heel slecht te zijn. De resultaten tonen echter ook aan dat het veranderen van parameters een duidelijk effect kan hebben op de performantie van de woordherkenning. Dit biedt perspectief voor verbetering. Indien dit in de toekomst verbeterd wordt, kan een volledig zelflerend systeem akoestische eenheden ontdekken en met dit als basis woordmodellen leren, zodat een volwaardig spraakherkenningssysteem bekomen wordt dat net zoals bij kinderen niet afhankelijk is van expertkennis.

Bibliografie

[1] D. Lee en Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, 1999.

[2] Ding *e. a.*, "Orthogonal nonnegative matrix tri-factorizations for clustering," *proceedings of the Twelfth ACM SIGKDD international conference on knowledge discovery and data mining, August 20-23, 2006, Philadelphia, PA, USA*, 2006.

[3] Ding, He, en Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," *Proc. SIAM Data Mining Conf.*, 2005.

[4] J. Shi en J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[5] A. Y. Ng, M. I. Jordan, en Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems 14*, 2002.

[6] K. Demuynck, T. Laureys, D. Van Compernolle, en H. Van hamme, "Flavor: a flexible architecture for lvcsr," *Proc. EUROSPEECH*, 2003.

[7] D. Lee en H. Seung, "Algorithms for non-negaticev matrix factorization," *Advances in Neural Information Processing Systems*, 2001.

[8] A. Betrand, "Zelflerende spraakherkenning d.m.v. matrixfactorisatie," *Katholieke Unversiteit Leuven*, Juni 2007.

Bijlage A

Voorbeelden van segmentaties

A.1 Segmentatie door ESAT-spraakherkenner

```
DATATYPE SEG
SEGTYPE VITERBI
MODEL_FILE wsj_train/modWSJ1_CMUv2.4/acmod.hmm
NENTRY 646
NSEG 143547
FSHIFT 0.010
TIMEBASE DISCRETE

ENTRY UNIT F1 NFR
##################################################
wsj0/si_tr_s/011/011o0308 ##0 1 1
-##1 2 62
-##2 64 1
-m#0 65 6
-m#1 71 1
-m#2 72 2
-I!#0 74 2
-I!#1 76 1
-I!#2 77 2
-s#0 79 3
-s#1 82 5
-s#2 87 1
-t#0 88 2
-t#1 90 2
-t#2 92 2
-}#0 94 3
-}#1 97 4
-}#2 101 2
-k#0 103 6
```

```
-k#1 109 4
-k#2 113 5
-r#0 118 3
-r#1 121 1
-r#2 122 2
-O#0 124 3
-O#1 127 10
-O#2 137 6
-s#0 143 4
-s#1 147 8
-s#2 155 8
-##0 163 1
-##1 164 1
-##2 165 1
-k#0 166 2
-k#1 168 4
-k#2 172 2
-O#0 174 4
-O#1 178 1
-O#2 179 1
-l#0 180 1
-l#1 181 1
-l#2 182 3
-d#0 185 1
-d#1 186 1
-d#2 187 1
-D#0 188 1
-D#1 189 1
-D#2 190 1
-@!#0 191 1
-@!#1 192 1
-@!#2 193 1
-p#0 194 5
-p#1 199 2
-p#2 201 5
-}!#0 206 4
-}!#1 210 1
-}!#2 211 5
-t+S#0 216 4
-t+S#1 220 4
-t+S#2 224 1
-@#0 225 2
-@#1 227 1
-@#2 228 1
-s#0 229 3
-s#1 232 5
-s#2 237 4
```

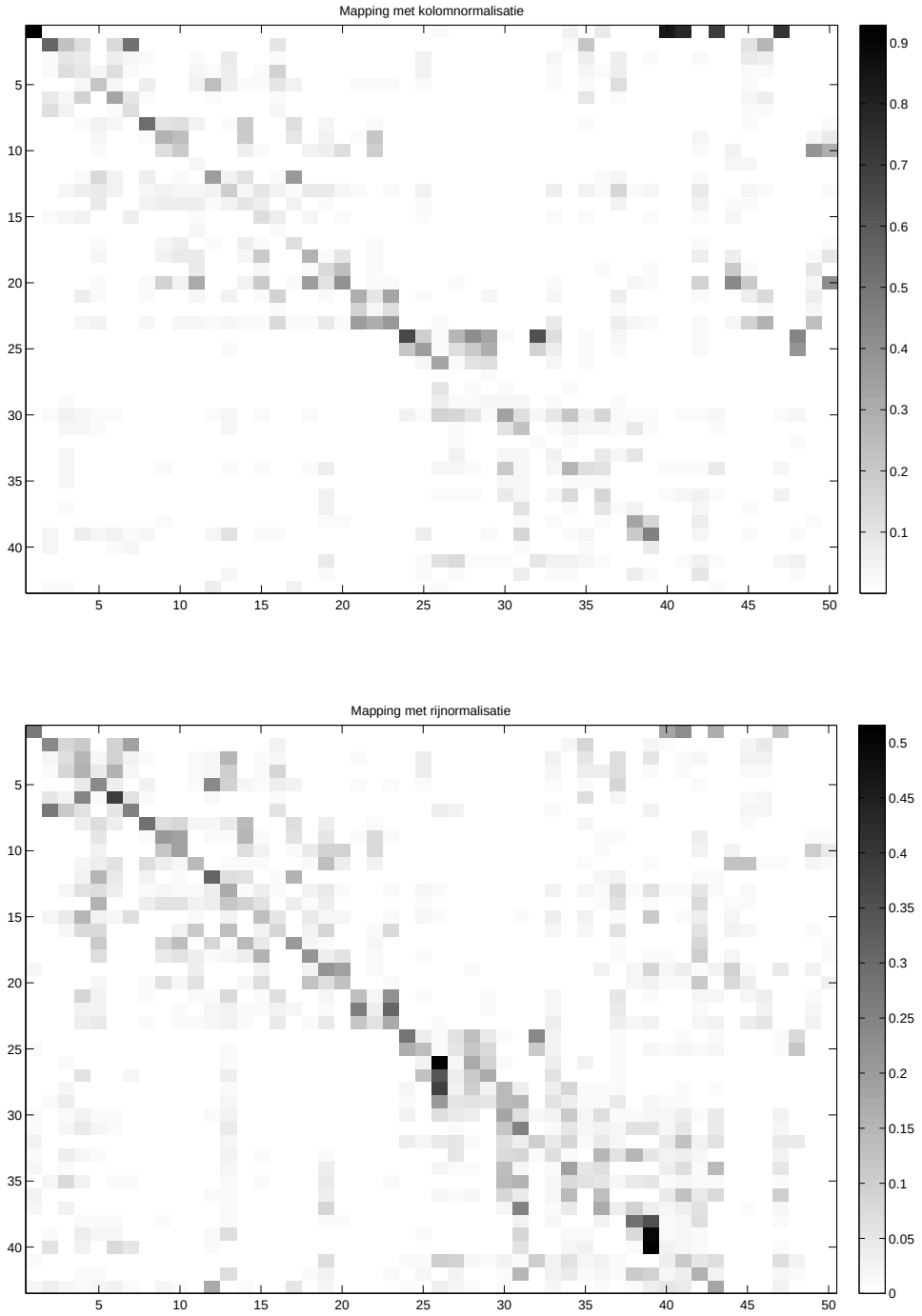
A.2 Segmentatie via clustering

```
DATATYPE SEG
SEGTYPE VITERBI
NENTRY 646
NSEG 131498
FSHIFT 0.010
TIMEBASE DISCRETE
#
wsj0/si_tr_s/011/011o0308 S8 1 46
-S18 47 2
-S8 49 12
-S22 61 2
-S18 63 2
-S8 65 3
-S50 68 2
-S45 70 2
-S41 72 2
-S38 74 3
-S59 77 1
-S21 78 2
-S36 80 2
-S49 82 9
-S27 91 2
-S50 93 1
-S35 94 2
-S25 96 4
-S1 100 2
-S52 102 3
-S6 105 3
-S10 108 2
-S29 110 3
-S56 113 7
-S28 120 3
-S14 123 9
-S42 132 11
-S48 143 2
-S36 145 1
-S4 146 12
-S33 158 3
-S22 161 2
-S18 163 4
-S10 167 2
-S29 169 2
-S56 171 3
-S28 174 3
```

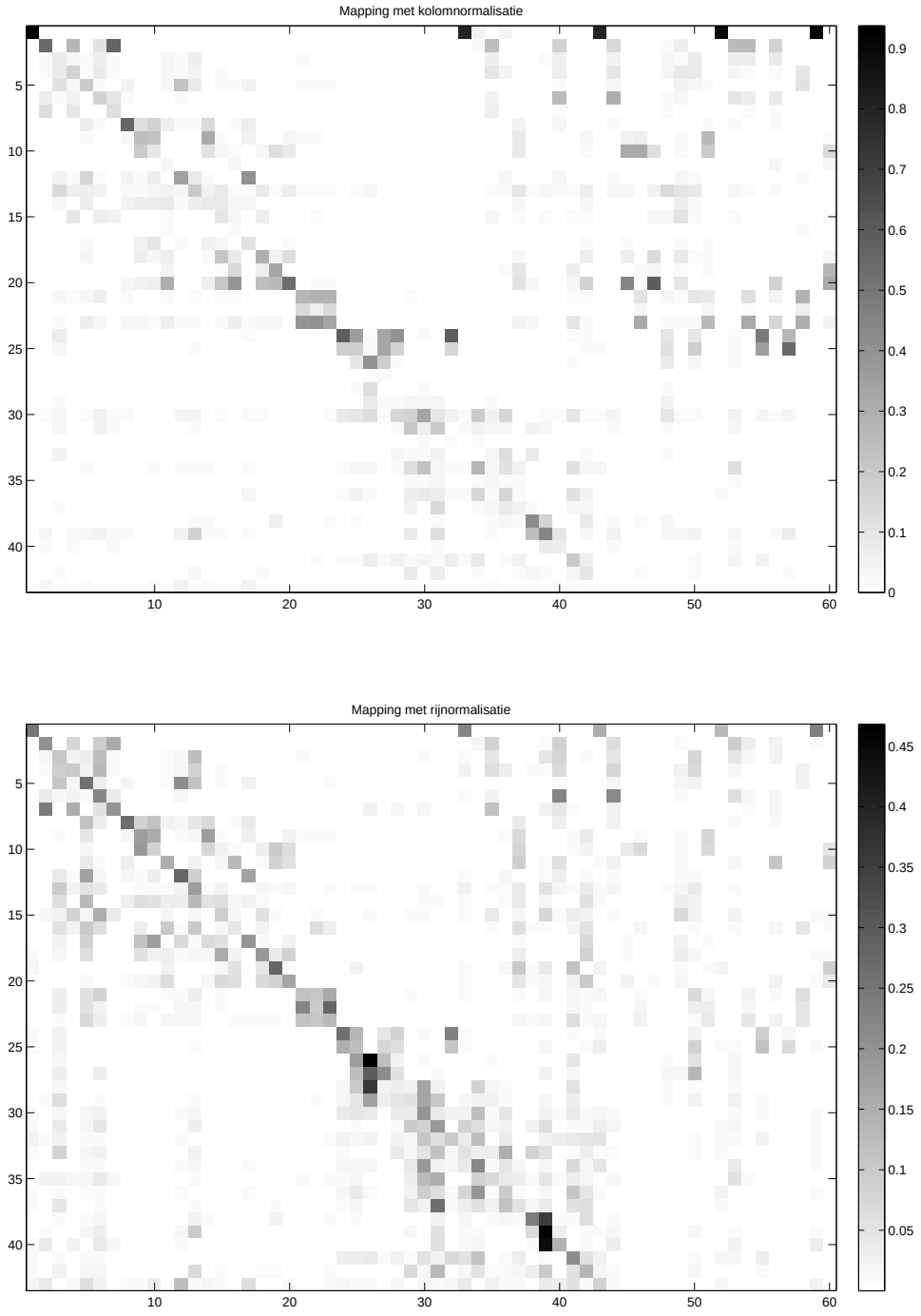
```
-S58 177 5
-S48 182 3
-S12 185 5
-S28 190 2
-S48 192 4
-S12 196 3
-S18 199 2
-S56 201 2
-S2 203 6
-S1 209 5
-S15 214 1
-S52 215 2
-S12 217 2
-S29 219 4
-S7 223 1
-S3 224 3
-S21 227 2
-S36 229 2
-S4 231 7
-S27 238 1
-S50 239 2
-S35 241 1
-S38 242 8
-S59 250 2
-S21 252 2
-S36 254 1
-S27 255 2
-S50 257 1
-S35 258 2
-S47 260 16
-S15 276 3
-S45 279 4
-S41 283 3
-S15 286 3
-S45 289 4
-S41 293 4
-S47 297 6
-S15 303 3
-S21 306 1
-S36 307 1
-S4 308 7
-S10 315 3
-S29 318 3
-S15 321 2
-S45 323 3
-S41 326 3
-S47 329 5
```

Bijlage B

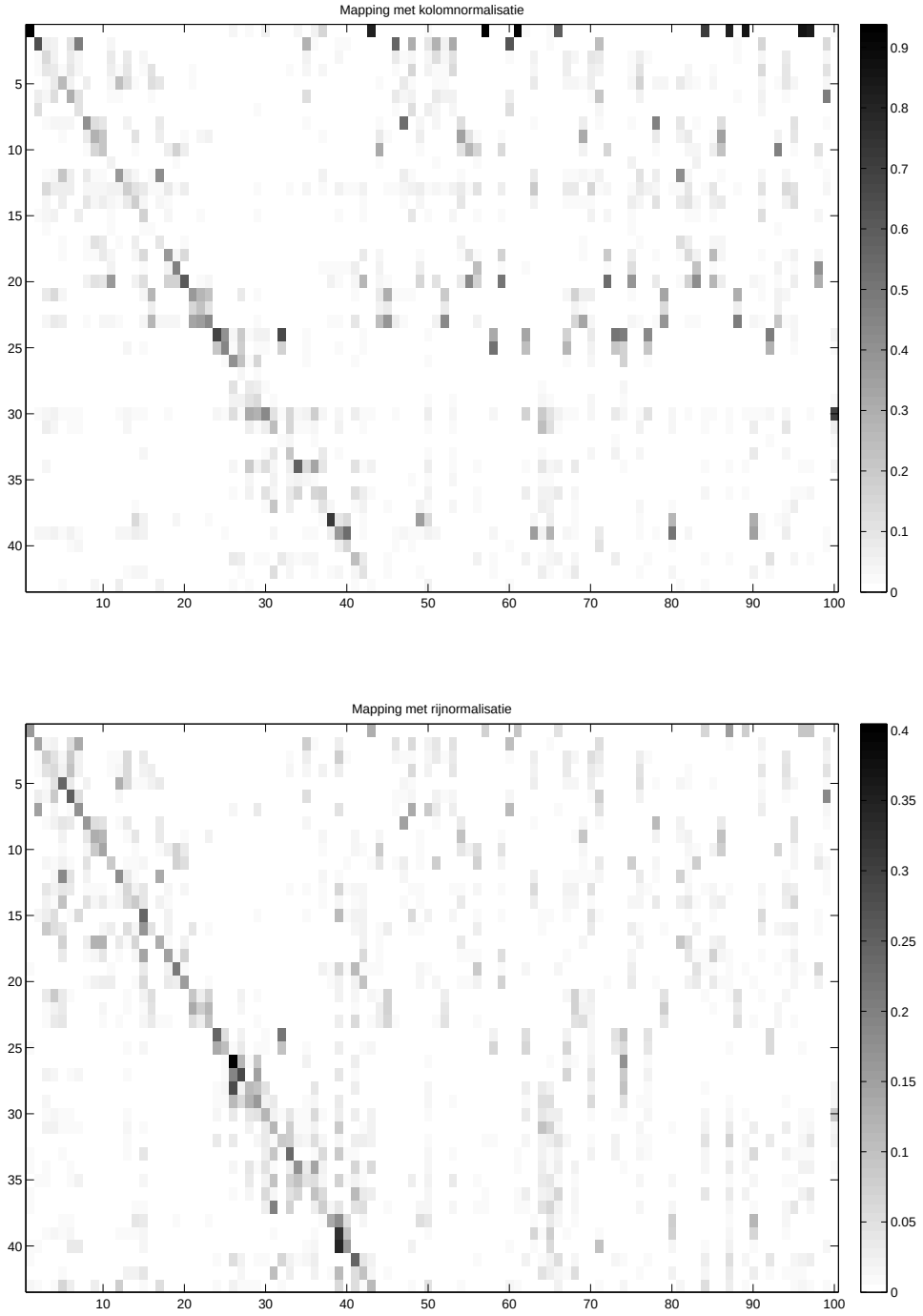
Mappingen tussen fonemen en akoestische eenheden



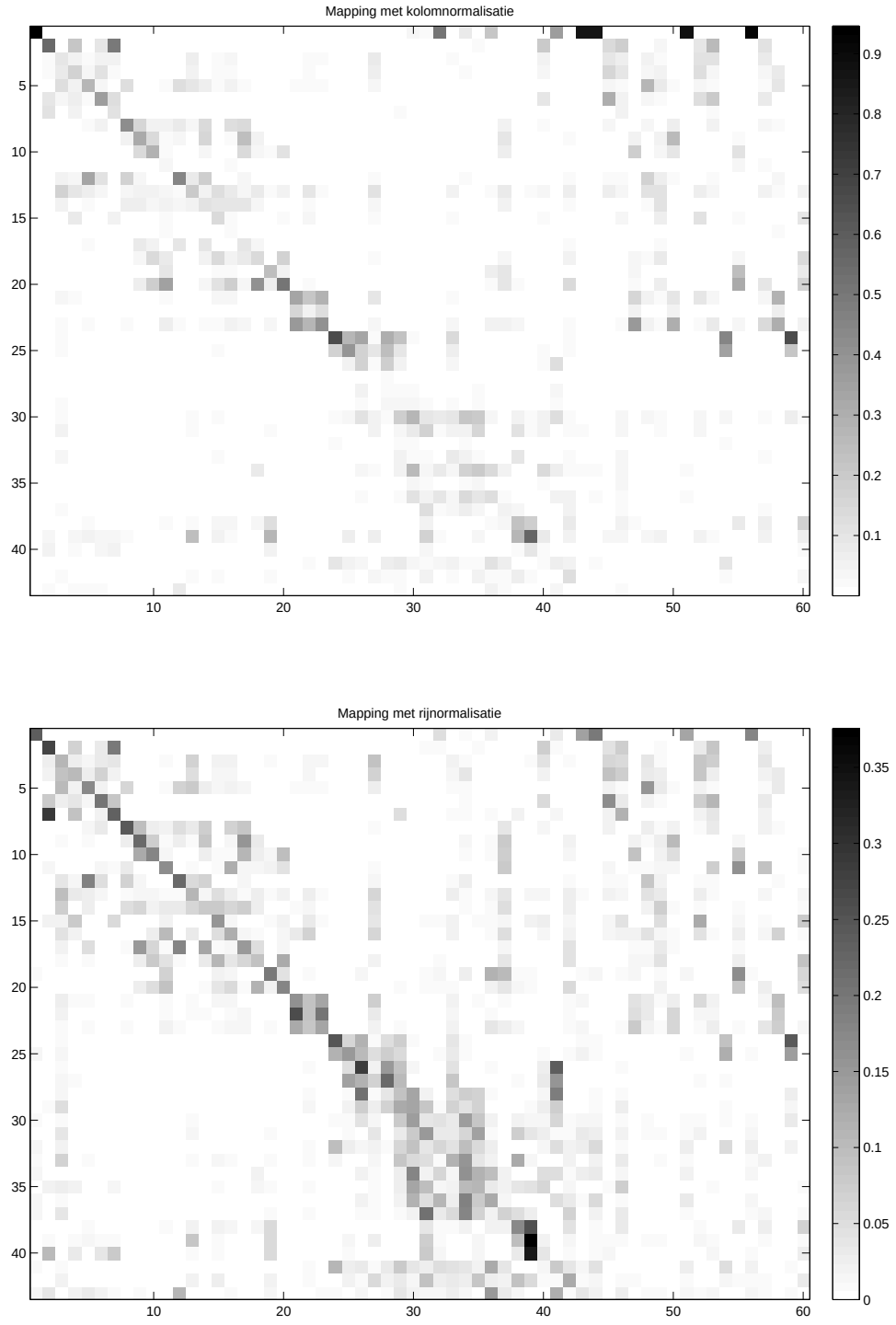
Figuur B.1: Mapping van gevonden akoestische eenheden op fonemen. Parameters: 50 akoestische eenheden, OR-symmetrie, 255 buren, 495.913 frames



Figuur B.2: Mapping van gevonden akoestische eenheden op fonemen. Parameters: 60 akoestische eenheden, OR-symmetrie, 255 buren, 495.913 frames; geen normalisatie van de H-matrix bij het bereken van de viterbi-oplijning



Figuur B.3: Mapping van gevonden akoestische eenheden op fonemen. Parameters: 100 akoestische eenheden, OR-symmetrie, 255 buren, 495.913 frames



Figuur B.4: Mapping van gevonden akoestische eenheden op fonemen. Parameters: 60 akoestische eenheden, OR-symmetrie, 255 buren, 495.913 frames; NMF-berekening met KL-divergentiecriterium