NTNU
Det skapende universitet

Analyse av Ulike Akustiske Taleegenskaper for Deteksjon av Artikulatoriske Attributter

Claus Fasseland
18. juni 2007

Institutt for Elektronikk og Telekommunikasjon
ved
Norges Teknisk-Naturvitenskapelige Universitet

Forord

Denne rapporten er utarbeidet i forbindelse med avsluttende masteroppgave ved Institutt for elektronikk og telekommunikasjon ved Norges Teknisk-Naturvitenskapelige Universitet.

Rapporten tar for seg analyse av akustiske taleegenskaper for bruk i gjenkjenning av talenheter. Gjenkjenneren baserer seg på rammedeteksjon av artikulatoriske taleattributter.

Jeg vil rette en stor takk til veileder og faglærer, Magne H. Johnsen.

Claus Fasseland
Trondheim, 18. juni 2007

Sammendrag

Arbeidet bak denne rapporten har omhandlet analysering av egendefinerte taleegenskapers potensial ved detektering av lydenheter (attributter). Et HTK-basert simuleringsverktøy har blitt utviklet for å kunne kartlegge egenskapenes evne til å frembringe gode gjenkjenningsresultater. En har undersøkt muligheten for forbedring av feilraten ved å representere talesignalet vha egenskapsvektorer som er tilpasset hver attributtdetektor. Utvelgelsen av vektorene er gjort ved subjektiv inspeksjon av attributtenes fordelingshistogram og kan dermed medføre menneskelige feilantagelser.

Undersøkelsene som er blitt utført viser, som forventet, underlegne resultater sammenlignet med referansen, MFCC. En beskjeden forbedring oppnås ved tilegnelse av utvalgte subsett av egenskaper til hver attributtdeteksjon. Nærmere tester bekrefter usikkerheten rundt potensialet til de egendefinerte egenskapsvektorene. Det er dermed anbefalt å redefinere egenskapene og eventuelt velge ut et nytt sett med attributter. En bør i tillegg vurdere å utvikle en bedre teknikk for utvelgelse av egenskapene for å optimalisere gjenkjenningen.

Innhold

1 Innledning **1**

2 Teori og Bakgrunnstoff **5**

2.1 Fonetikk og Fonologi 6

2.1.1 Fonemer 7

2.1.2 Taleattributter 8

2.2 Egenskapsuttrekning 10

2.3 Perseptuelle Egenskaper 10

2.4 Artikulatoriske Egenskaper 12

2.4.1 Initielle Akustiske Egenskaper 12

2.5 Statistiske Modeller 17

2.5.1 HMM vs GMM 18

2.5.2 Gaussiske Blandingsfordelinger 19

2.5.3 Estimering av Modellparametere 21

2.5.4 Skjulte Markovmodeller 21

2.6 Deteksjon 22

2.6.1 Metoder for Beslutningsrate 23

3 Hjelpeverktøy **25**

3.1 Taledatabase: TIMIT 25

3.2 Programvare: HTK 27

4 Utvikling av Simuleringsverktøy **31**

4.1 Trening 32

4.2 Testing/Gjenkjenning 35

5 Analyse av Egenskapsvektorer **37**
5.1 Initielle Egenskaper . . . 37
5.2 Utvalg av Potensielle Egenskaper . . . 41
5.3 MFCC . . . 41

6 Resultater **43**

7 Evaluering og Diskusjon **47**
7.1 Akustiske Egenskapsvektorer . . . 47
7.2 Deteksjonsrate: Trening vs Test . . . 50
7.3 Feildetektering av Transisjonsrammer . . . 51
7.4 Korrelasjon . . . 52

8 Konklusjon og Videre Arbeid **55**

A Egenskapsanalyse **63**
A.1 Utvelgelse av Akustiske Taleegenskaper . . . 63
A.2 Korrelasjon i Egenskapsvektorer . . . 65

B Fordelingsdiagram for Egenskaper **67**

C Kildekode **73**
C.1 Matlab . . . 74
C.1.1 preprFE2htk.m . . . 74
C.1.2 trainMatlab.m . . . 81
C.1.3 recMatlab.m . . . 84
C.1.4 read_mfc_label2.m . . . 88
C.1.5 read_models.m . . . 92
C.1.6 pdfCalc2.m . . . 96
C.1.7 optThresEERMER.m . . . 98
C.1.8 testData2.m . . . 101
C.1.9 featureSelect.m . . . 104
C.1.10 testTrainData.m . . . 107
C.1.11 displayMixInc.m . . . 109
C.2 Perl . . . 115
C.2.1 quickstartup_trainScript . . . 115
C.2.2 trainScript . . . 117
C.2.3 recScript . . . 127

Figurer

2.1 Triangulære filtre for beregning av MFCC. 11
2.2 Formantdiagram for vokaler 14
2.3 Formantdiagram for frikativer 14
2.4 Illustrasjon av ulikhet mellom stemte og ustemte lyder 15
2.5 GMM (to blandingskomponenter) for stillhet og parameteren energi (F3-Nyq) 20
2.6 GMM (fire blandingskomponenter) for stillhet og parameteren energi (F3-Nyq) 20
2.7 GMM (seks blandingskomponenter) for vokaler og parameteren nullkryssingsrate 21
2.8 Tre-tilstands HMM 22
2.9 Illustrasjons av deteksjon 23
2.10 Terskel for EER mhp FAR og FRR. 24

4.1 Blokkskjema for treningsdelen. 32
4.2 Flytskjema for treningsdelen. 33
4.3 Blokkskjema for test-/gjenkjenningsdelen. 35
4.4 Flytskjema for test-/gjenkjenningsdelen. 36

5.1 Histogram for fordelingen av F4 for approksimanter 38
5.2 Histogram for fordelingen av båndbredden til F4 for approksimanter 39
5.3 Histogram for fordelingen av *stemthet* i alveolarklassen 39
5.4 Histogram for fordelingen av energiparameter for attributten *stillhet* 40

5.5 Histogram for fordelingen av energiparameter for *Dental* . . . 40
5.6 Histogram for fordelingen av kepstralkoeffisient 11 for *stillhet* 42
5.7 Histogram for fordelingen av kepstralkoeffisient 4 for *stillhet* . 42

7.1 Korrelasjon mellom de 13 koeffisientene i MFCC for attributten *Vokal* 53
7.2 Korrelasjon mellom de 20 initielle akustiske egenskapene for attributten *Vokal* 53

A.1 Korrelasjon mellom de 13 koeffisientene i MFCC for attributten *Dental* 65
A.2 Korrelasjon mellom de 20 initielle akustiske egenskapene for attributten *Dental* 66

C.1 Funksjonsdiagram for kildekoden. 74

Tabeller

2.1 Artikulasjonsmåtene og deres tilhørende fonemer. 8
2.2 Artikulasjonsstedene og deres tilhørende fonemer. 9
2.3 Oversikt over de 20 initielle akustiske egenskapene. 13

3.1 Oversiktstabell over viktige funksjoner fra HTK. 28

6.1 Resultat i feilprosent for de fire ulike eksperimentene 44
6.2 Resultat presentert med FAR og FRR 45

7.1 Trening- vs Testresultater - egendefinerte egenskaper 50
7.2 Trening- vs Testresultater - MFCC 51

A.1 Utvelgelsestabell for egenskaper: sikre kombinasjoner 63
A.2 Utvelgelsestabell for egenskaper: sikre og mindre sikre kombinasjoner 64
A.3 Resultat for egenskapsvektor basert på korrelasjon 65
A.4 Korrelasjonsmatrise 66

B.1 Nummerering av egenskaper 67
B.2 Feilrate i prosent for hver attributtdetektor fra ulike simuleringer 68

Forkortelser

Forkortelse	*Engelsk*	*Norsk*
DCT	Discrete Cosine Transform	Diskret Cosinustransform
MFCC	Mel Frequency Cepstral Coeffisient	Mel-frekvens Kepstralkoeffisient
HMM	Hidden Markov Model	Skjulte Markovmodeller
HTK	HMM Tool Kit	Verktøy for Skjulte Markovmodeller
GMM	Gaussian Mixture Models	Gaussiske Blandingsfordelinger
PDF	Probability Density Function	Sannsynlighets-tetthetsfunksjon
MLE	Maximum Likelihood Estimation	Sannsynlighets-maksimeringsestimering
EER	Equal Error Rate	Lik Feilrate
MER	Minimum Error Rate	Minimum Feilrate
FRR	False Rejection Rate	Falsk Forkastningsrate
FAR	False Acceptance Rate	Falsk Akseptrate
LPC	Linear Predictive Coding	Lineær Prediksjonskoding
SHR	Subharmonic-to-Harmonic Ratio	Underharmonisk-til-harmoniskrate
ZCR	Zero Crossing Rate	Nullkryssingsrate

Kapittel 1

Innledning

Gjenkjenning av tale krever kompliserte og intelligente systemer. Dette skyldes at menneskets måte å oppfatte og forstå tale på er svært vanskelig å overføre til datadreven gjenkjenning. Kompleksiteten skyldes blant annet av stor variasjon i talens karakteristikk; ulike dialekter, spontan tale osv.

Talens komplekse natur fikk forskere til å utvikle statistiske modeller for å kunne klassifisere taleenheter. Skjulte Markovmodeller, et rammeverk for statistiske modeller, ble dominerende og har vært sterkt medvirkende til at fagfeltet, automatisk talegjenkjenning, oppnådde gode forbedringer frem til slutten av forrige århundret. Dessverre nådde man et metningspunkt og utviklingen har mer eller mindre blitt stående på stedet hvil de siste 5 årene. Forskere innså etterhvert at et internasjonalt samarbeid var veien man burde gå for å utvikle fagområdet videre og gi forskningsmiljøene ny motivasjon. I 2004 ble samarbeidsprosjektet ASAT [1] initiert med støtte fra National Science Foundation (NSF). Prosjektets mål var å utvikle en felles plattform for forbedret samarbeid på tvers av forskningsmiljøer. Fokuset deres ble lagt på kunnskap innen taleanalyse og bruk av artikulatoriske attributter i et statistisk rammeverk. Flere utviklingsmiljøer rundt om i verden har siden da blitt motivert mot en lignende visjon. NTNU-prosjektet SIRKUS [1] har sterke bånd til ASAT. Arbeidet i denne rapporten er utført som et ledd av forskningen til SIRKUS.

[1] SIRKUS, *Spoken Information Retrieval by Knowledge Utilization in Statistical Speech Processing*

Tradisjonelt har man bygd gjenkjennere for klassifisering av taleenhetene fonemer, ord eller setninger. Fonemer er den minste enheten man til nå har brukt for inndeling av talelyder. De er definert ved at betydningen av et ord endres om et fonem byttes eller fjernes. De er ikke basert på *fysiske* egenskaper. For å oppnå høyest mulig korrelasjon mellom de akustiske (fysiske) egenskapsvektorene og de karakteristiske egenskapene knyttet til en taleenhet, vil alternative taleenheter brukes. Disse vil heretter omtales som *attributter*. Attributtene er basert på artikulasjonsmåte eller artikulasjonssted og vil erstatte mer tradisjonelle taleenheter. Eksempler på taleenheter er stemt/ustemt, vokaler/konsonanter osv.

Oppgavens gjenkjenningsdel vil benytte seg av deteksjon til fordel for klassifisering. Ved klassifisering definerer man et sett med klasser og forsøker å klassifisere et lite utsnitt av talen som en av disse klassene. I deteksjon bestemmes først hvilken taleattributt som skal detekteres. Deretter defineres to klasser, en for den valgte attributten og en for *alle* andre. Ved å sammenligne talens tilhørighet til de to klassene kan man vha en bestemt terskelverdi *detektere* talen som den valgte attributten eller ikke.

Detektorene vil bli implementert for å detektere enkeltrammer, dvs små tidsrammer på 10 ms. Alternativet ville vært hele segmenter (f.eks fonem eller attributt). Anvendelse av rammer kontra segmenter gir mulighet for mindre kompleks gjenkjenner/detektor ved at man ikke behøver å inkorporere variasjon over tid. Ulempen er at rammer i grenseintervallet mellom påfølgende fonemer kan inneholde svært begrenset karakteristisk informasjon. Ettersom hovedfokuset for oppgaven er analyse og testing av akustiske egenskaper og ikke selve gjenkjenningen, vil kun rammer benyttes.

Arbeidet knyttet til oppgaven består i hovedsak av å teste et sett ulike kunnskapsbaserte akustiske taleegenskaper mhp potensialet i deteksjon av ulike artikulatoriske klasser av tale. Som en referanse skal en i tillegg bruke tradisjonelle egenskapsvektorer (sett med flere egenskaper) basert på persepsjon som modell, slik som mel-frekvens kepstralkoeffisienter (MFCCs). Kunnskap knyttet til menneskets oppfattelse (persepsjon) av tale er svært begrenset pga hjernens særdeles komplekse natur. Menneskets måte å produsere (artikulere) tale er derimot nøye kartlagt ved hjelp av akustiske egenskaper. Man kan derfor lettere analysere de ulike egenskapene og deres potensial i gjenkjenningssammenheng.

Hovemålet med oppgaven er å kartlegge *hvilke* akustiske egenskaper som best tjener deteksjon av *hvilke* attributter. Ved utnyttelse av denne kunnskapen skal en da kunne designe detektorene individuelt med hensyn på de egenskapene som er best egnet. En håper da at dette skal resultere i detektorer med høyere gjenkjenningsrate.

Hovedoppgaven er en fortsettelse på et tidligere prosjekt hvor lignende egenskapsvektorer og rammedeteksjon ble testet. Da ble simuleringsverktøyet implementert i Matlab. Funksjoner for å beregne parametere til de statistiske modellene ble her hentet fra voicebox-prosjektet [2]. Deres algoritme for estimering av disse parameterne (expectation-maximization (EM) algorithm) mistenkes å være ugunstig ved økning av antall gaussiske blandings-fordelinger, og av den grunn har Hidden Markov Model Tool Kit (HTK) blitt tatt i bruk. HTK er ansett som det beste verktøyet innen talegjenkjenning og skal ha en velfungerende implementering av EM-algoritmen med støtte for multidimensjonale fordelinger.

Gjennom eksperimenter og simuleringer ønsker man gjennom dette arbeidet å bygge opp kunnskap rundt akustiske egenskapsvektorer basert på artikulasjon. Senere i rapporten vil det bli lagt frem resultater for akustiske og MFCC-baserte vektorer. Mulige løsninger for forbedring vil også bli presentert.

Valg av akustiske egenskaper, attributter og hjelpeverktøy er på forhånd bestemt av faglærer som en del av oppgaven.

Rapporten er videre strukturert som følger: Neste kapittel forklarer teoretiske begreper rundt de metoder og teknikker som anvendes i implementasjonen og ellers bakgrunnstoff for forståelse av arbeidet. Kapittel 3 er et kort kapittel for beskrivelse av nødvendige eksterne hjelpemidler. Implementasjonsdelen i kapittel 4 tar for seg oppbygging av trenings- og gjenkjenningsdelen. Kapittel 5 vil omhandle analyse av egenskapsvektorene, mens kapittel 6 vil presentere resultater knyttet til simuleringene. Kapitlene 7 og 8 er satt av til diskusjon og konklusjon. Bakerst i rapporten finner en et tillegg med egenskapsanalyse, fordelingsdiagram for egenskaper og kildekode.

Kapittel 2

Teori og Bakgrunnstoff

Fagfeltet automatisk talegjenkjenning oppstod allerede på 1940-tallet [3], da AT&T Bell Laboratories utviklet den første talegjenkjenneren, dog primitiv i forhold til hva man har i dag. Siden den tid har fagmiljøer fra hele verden forsket mot samme mål, nemlig å utvikle systemer som forstår menneskelig tale. Menneskets metoder for å forstå tale er svært komplekse og det er derfor veldig vanskelig med kunstig etterlignelse vha datamaskiner. Øret og hjernen utfører et unikt arbeid sammen. Metodene de bruker for å oppfatte tale fra bakgrunnstøy vha filtrering og til klassifisering av lyder er så komplekst at ingen systemer til nå har klart kopiere dem.

Menneskelig tale er et resultat av lufttrykk i munn og nese. Ved kontrollert påvirkning fra organer som stemmebånd, talerøret og leppene gjør at man har mulighet til å lage forskjellige lyder, som andre kan forstå. Hvordan lydene faktisk blir produsert kan avvike fra person til person og derfor vil presentasjonen av talen være forskjellig (folk snakker forskjellig). Dette fører til stor usikkerhet når man skal gjenkjenne tale både for et menneske og da spesielt for en maskin.

Dette kapittelet vil blant annet omhandle hvordan man kan klassifisere talen, hvordan man best kan representere talen for maskinen og hvilke metoder man bruker for å gjenkjenne talen på best mulig måte. Taleteknologi er et utstrakt fagfelt, derfor vil de påfølgende seksjonene kun ta for seg det mest relevante for forståelse av oppgaven. Teori utover oppgavens omfang er mindre utdypet

eller utelatt.

2.1 Fonetikk og Fonologi

Fonetikk og fonologi er de to studiene som omhandler språket vårt. Førstnevnte betegnes som læren om språklydene og hvordan de oppstår, mens fonologi tar for seg hvordan små språkenheter skiller seg fra hverandre og hvordan de passer inn i språket vårt. Fonetikk kan deles inn i to hoveddeler: *Artikulatorisk* og *akustisk* fonetikk. I tillegg definerer enkelte også *auditiv* fonetikk som en likeverdig del [4]. Artikulatorisk fonetikk omhandler de anatomiske og fysiologiske egenskaper til taleapparatet, akustisk fonetikk tar for seg fysiske egenskaper til lydbølger og auditiv fonetikk beskriver de perseptuelle egenskapene til øret og hjernen.

Artikulatorisk fonetikk er begrenset til det fysiske aspektet ved produksjon av tale. Taleapparatet er en fellesbetegnelse for de organene som bidrar til utformingen av det et menneske oppfatter som tale. Menneskelig tale blir produsert ved at luft gjennom åndedrettet blir påvirket på vei gjennom luftrøret, strupen og til slutt leppene og tennene eller nesehulen. I strupen sitter stemmebåndene, som er to folder som sitter festet på hver sin side. Når de er foldet mot hverandre vil de skape en hindring i luftpassasjen og danne et lufttrykk. Dette fører til vibrasjon i stemmebåndene og resulterer i såkalte *stemte* lyder. Vokaler er normalt stemte, mens konsonanter både har tilfeller av stemte (*p, d, g*) og ustemte (*p, t, k*). Videre oppover kommer så ansatsrøret som består av svelget, nesehulen og munnhulen. Det vil her skapes resonans etter formen på organet når luft passerer.

Akustisk fonetikk omhandler de fysiske (akustiske) egenskaper i lydbølger som dannes ved hjelp av taleapparatet. Artikulatorisk fonetikk forteller hvilke organer som er involvert og hvordan de påvirker lyden, mens akustisk fonetikk beskriver egenskapene i talen som er dannet som et resultat av artikulatorenes innvirkning. Slike egenskaper kan for eksempel være talens grunnfrekvens, amplitude og stemhet. Flere akustiske egenskaper vil få en nærmere utredning i kapittel 2.4.1.

Auditiv og perseptorisk fonetikk er læren om den prosessen der mennesket oppfatter og forstår ytret tale. Øret fanger opp og oppfatter talen, mens hjernen står for forståelsen, dvs skille mellom forskjellige ordlyder. I den indre delen av øret befinner det seg hårceller. Når lydbølger beveger seg inn til det indre øret, oppstår vibrasjon i hårcellene. Denne mekaniske tilstanden blir da videre omdannet til nerveimpulser og sendes til hjernen. Hjernen er bygd opp av et nett med nevroner (nerveceller), som påvirkes av nerveimpulsene fra øret. Den klarer da å trekke ut informasjon nok til å forstå lyder. Det er delte meninger om persepsjon skal inngå i fonetikk. Det er likevel tatt med for å gi en bredere bakgrunnsforståelse for rapporten.

(*Occupational Exposure to Noise: Evaluation, Prevention and Control* [5], *Acoustic and Auditory Phonetics* [6], *A course in phonetics* [7])

2.1.1 Fonemer

Fonemer er den minste enheten man til nå har brukt for inndeling av talelyder. De er den minste ordlyden som kan skille betydningen til et ord fra et annet [8]. De vil variere fra person til person, men vil alltid ha funksjonalitet som betydningsskillere mellom ord.

Fonemer er dannet for å kunne beskrive språket vårt i form av lydskrift. Et fullstendig sett av fonemer kan ses på som et alfabet for språklyder. Det mest kjente og brukte er *Speech Assignment Methods Phonetics Alphabet (SAMPA)*. Alfabetet er basert på *International Phonetic Alphabet (IPA)*. SAMPA omfatter rundt 25 forskjellige språk, deriblant engelsk, fransk, norsk og thailandsk.

Ved hjelp av et fullstendig sett med fonemer for et gitt språk, kan man beskrive *alle* ord og setninger i språket. SAMPAs amerikansk-engelske alfabet består av omtrent femti fonemer. Til sammenligning kreves det enormt mange flere *ord* til å beskrive samme vokabular.

I talegjenkjenning kreves det naturligvis mindre komplekse klassifiserere for femti fonemer enn for mange tusen ord. Derfor anvendes små enheter som fonemer for talegjenkjenning av kontinuerlig tale med stort vokabular.

Fonemer kan deles inn i flere grupper basert på *hvilke* og *hvordan* organer medvirker til språklydene og *hvor* påvirkningene oppstår. Ofte deler man

dem inn i disse hoveddelene:

- Vokaler
- Konsonanter

Deres mest markante forskjell er hvordan luften blir påvirket gjennom ansatsrøret. Vokallyder passerer forholdsvis upåvirket, mens konsonanter blir delvis hindret og luften blir turbulent.

2.1.2 Taleattributter

Oppgavens hovedfokus er bruk av akustiske egenskapsvektorer for detektering av attributter. En antar at de artikulatoriske attributtene har ulike akustiske egenskaper, men med klar karakteristikk innad i klassen. Attributtenes akustiske karakteristikk er grunnlag for bruken av dem til fordel for fonemer. Ettersom standard taledata er merket som fonemer, må en transformere dem over til attributter. Attributter har en grovere inndeling enn fonemer og det vil da si at hver attributtklasse inneholder flere fonemer. Klassene vil i hovedsak baseres på produksjonssted (artikulasjonssted) og produksjonsmåte (artikulasjonsmåte), henholdsvis 8 og 6 klasser. Attributtene som anvendes her er ikke globalt definert, men er spesielt bestemt for denne oppgaven. De er derimot ofte brukt i talegjenkjenning. For eksempel har man i [9] og [10] anvendt lignende attributter for artikulasjonsmåte og i [11] tilsvarende attributter for sted.

Nr.	*Artikulasjonsmåte*	*Fonem*
1	Frikativer	ch, jh, z, s, dh, th, f, v, hh, sh, zh
2	Nasaler	en, n, nx, em, m, eng, ng
3	Stopp	t, dx, d, p, b, k, g
4	Vokaler	er, axr, ix, ih, iy, eh, ey, ae ay, aw, aa, ao, oy, ow, uh, uw
5	Approksimanter	w, y, l, el, r, hv
6	Stillhet	pau, epi, #h, h#

Tabell 2.1: Artikulasjonsmåtene og deres tilhørende fonemer.

Ytring av attributtene i Tabell 2.1 oppstår etter forskjellig påvirkning fra artikulatorene i ansatsrøret. Ved artikulasjon av *vokaler* skapes det ikke turbulens i luften slik som ved konsonanter. De forskjellige språklydene i

denne klassen blir dannet ved form og posisjon av tunge, tenner og lepper. *Frikativer*, *nasaler*, *stopp* og *approksimanter* tilhører konsonantklassen. Førstnevnte dannes ved delvis hindring i ansatsrøret slik at det oppstår turbulent luftstrøm, og resultatet er da en høyfrekvent vislende lyd. Stopp-konsonanter er egentlig en fellesbetegnelse på lyder hvor luften enten stoppes i munnen, *oral stop*, eller gjennom nesa, *nasal stop*. Fonetikere betegner likevel *oral stop* som stopp og *nasal stop* som nasal. Den siste klassen av konsonanter, approksimanter, er ofte ansett som en litt vrien attributt for gjenkjenning. De plasserer seg nemlig mellom konsonanter og vokaler. Artikulatorene i ansatsrøret er plassert nærme hverandre, men *uten* at luftstrømmen blir turbulent i like stor grad som for konsonanter.

Nr.	*Artikulasjonssted*	*Fonem*
7	Alveolar	ch, jh, z, s, en, n, nx, t, dx, d
8	Dental	dh, th
9	Lab-dent	f, v
10	Labial	hh, em, m, p, b
11	Velar	sh, zh, eng, ng, k, g
12	Bak	w, r, er, axr, ao, oy, ow, uh, uw
13	Senter	l, el, hv, aw, aa
14	Front	y, ix, ih, iy, eh, ey, ae, ay

Tabell 2.2: Artikulasjonsstedene og deres tilhørende fonemer.

Taleattributtene i Tabell 2.2 er basert på *hvor* de dannes i taleapparatet. *Alveolarklassen* består av de ordlydene som oppstår ved bruk av tuppen av tunga eller tungebladet mot alveolarkanten (tannkjøttet). For å uttale lyder fra *dentalklassen* må man presse tungetuppen eller tungebladet mot de øvre fortennene. Her kan det være litt variasjon mellom de forskjellige amerikanske aksentene. Noen plassere tunga mellom øvre og nedre fortenner og kalles da *interdental*. De lydene som dannes ved bruk av begge leppene, kalles *labial*, mens *lab-dent* (labiodental) er når underleppa og de nedre fronttennene føres sammen. *Velar* dannes i bakre del av ganen (*velum*) ved at tungas bakre del presses opp. *Bak*, *senter* og *front* er vokaler som klassifiseres av posisjonen av tungas høyeste punkt [7].

2.2 Egenskapsuttrekning

Hvordan man representerer tale er svært avgjørende for et gjenkjennings-system. Man ønsker å trekke ut egenskaper i talen som gjør det lettere å skille forskjellige taleenheter fra hverandre og dermed øke gjenkjenningsraten. Det er i hovedsak to metoder å representere tale på, basert på enten gjenkjenning av taleproduksjon eller persepsjon (oppfattelse) av tale. Tradisjonelt i tale-gjenkjenning har man brukt mel-frekvens kepstralkoeffisienter, ettersom de innehar gode egenskaper for tilnærming av menneskelig taleoppfattelse. I motsetning til disse, er artikulatorisk gjenkjenning basert på kunnskap til talens akustiske egenskaper.

Valg av representasjon er ikke entydig. Man skulle kanskje tro at jo flere egenskaper vi benytter jo tydeligere skille skapes mellom enhetene. Slik er det nødvendigvis ikke. Et slikt problem har fått navnet *"The curse of dimensionality"*. utdypet utdypt i [12]. Teoretisk skal økning av egenskapsvektorer bedre resultatet, men i praksis har det vist seg å ikke stemme. Problemet er trenbarheten til tilgjengelig data, dvs i hvilken tilfredsstiller treningsdataene parameterne i klassifisereren. Om de ikke er tilfredsstillende nok, kan en økning i dimensjon resultere i lavere gjenkjenningsrate.

Det finnes mange forskjellige teknikker uttrekning av egenskaper, både for artikulasjon og persepsjon. Noen av dem er kommentert i henholdsvis kapittel 2.4 og 2.3. De fleste teknikker er utarbeidet under forutsetning av at tale-dataene er statistiske stasjonære. Dessverre er tale vanligvis ikke-stasjonært. Innenfor et kort tidsrom (10 - 20 ms) kan man likevel anta at tale (spesielt vokaler) er *tilnærmet* stasjonært. Problemet kan da løses ved å inndele talen i mange tilsvarende korte tidsrammer, for så å beregne egenskaper ramme for ramme.

2.3 Perseptuelle Egenskaper

Kunnskapen rundt menneskets persepsjon er, som tidligere nevnt, svært begrenset. En mulighet er å beregne talens Mel-frekvens kepstralkoeffisienter (eng: Mel Frequency Cepstral Coefficients - MFCC). Egenskapsvektorer av

MFCCs er det mest anvendte i moderne talegjenkjenning. Deres styrke er bruken av mel-frekvensskalaen som er tilnærmet lineær under 1 KHz og tilnærmet logaritmisk over 1 KHz. Dette er nemlig veldig likt den følsomheten vår hørsel har for frekvenser. I tillegg har koeffisientene gode korrelasjonsegenskaper.

Fremgangsmåten for å beregne MFCCs er som følger:

Først transformerer man inngangssignalet til Diskret Fourier Transform (DFT),

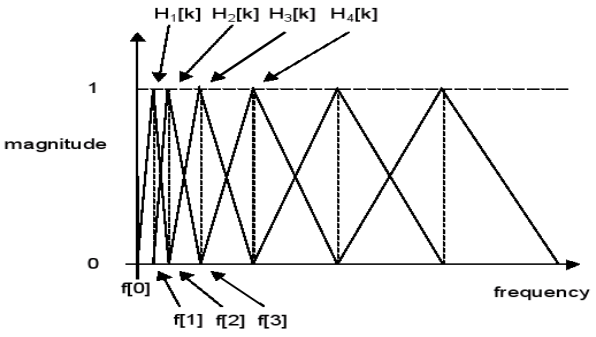
$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N} \tag{2.1}$$

, hvor $0 \leq \mathrm{k} < \mathrm{N}$.

Deretter defineres en filterbank med M triangulære filtre, hvor filter m er:

$$H_m[k] = \begin{cases} 0 & k < f[m-1] \\ \frac{2(k-f[m-1])}{(f[m+1]-f[m-1])(f[m]-f[m-1])} & f[m-1] \leq k \leq f[m] \\ \frac{2(f[m+1]-k)}{(f[m+1]-f[m-1])(f[m+1]-f[m])} & f[m] \leq k \leq f[m+1] \\ 0 & k > f[m-1] \end{cases} \tag{2.2}$$

For hvert filter m beregnes en gjennomsnittsverdi rundt senterfrekvensen som vist i Figur 2.1.



Figur 2.1: Triangulære filtre fra formel 2.2 for beregning av mel-kepstrum.

Koeffisientene kan da beregnes ved dekorrelasjon av de M filtrenes utgangsverdier. Dette gjøres ved bruk av diskrete cosinus transform (eng: Discrete

Cosine Transform - DCT):

$$c[n] = \sum_{m=0}^{M-1} S[m] \cos \pi n(m - 1/2)/M, \qquad 0 \leq n < M \tag{2.3}$$

, hvor

$$S[m] = \sum_{k=0}^{N-1} \left[\ln(|X[k]|^2 H_m[k]) \right], \qquad 0 < m \leq M \tag{2.4}$$

MFCCs inneholder kun statiske egenskaper for det gitte vinduet, men ved å legge inn tidsdynamikk øker man den lingvistiske informasjonmengden. Dette gjøres ved å utvide egenskapsvektorene med deltakoeffisienter, som er den tidsderiverte rundt en koeffisient.

2.4 Artikulatoriske Egenskaper

Akustiske egenskapsvektorer representerer kunnskap om artikulasjon av tale. Motivasjonen for slik representasjon er at man anser de fysiske egenskapene til å være karakteristiske nok til å kunne basere gjenkjenning på. Man ønsker å ekstrahere distinktive egenskaper for de forskjellige språkenhetene man skal klassifisere. Valg av godt egnede egenskaper resulterer i større karakteristisk avvik mellom klasser og kan øke sannsynligheten for korrekt gjenkjenning. Ved bruk av slike egenskapsvektorer er det viktig å velge dem ut med omhu. Egenskapene bør i størst mulig grad innholde karakteristisk informasjon knyttet til attributtene som gjenkjennes.

Arbeidet med oppgaven ble initiert med 20 ulike akustiske egenskaper, som på forhånd ble valgt ut av faglærer. Tabell 2.3 gir en komplett liste med korte beskrivelser. Resten av kapittelet er tilrettelagt for nærmere redegjørelse av egenskapene.

2.4.1 Initielle Akustiske Egenskaper

Formanter besitter viktig fonetisk informasjon og er dermed ofte brukt i analyse av tale. Likevel er de sjelden brukt i talegjenkjenning. Det skyldes

Nr.	*Egenskapsbeskrivelse*
1-4	Frekvens for formant F1,F2,F3 og F4
5-8	Amplitudeverdi for formant F1,F2,F3 og F4
9-12	Båndbredde for formant F1,F2,F3 og F4
13	Grunnfrekvens (F0)
14	Grad av stemthet
15	Spektralflathet
16	Midlet svingning i frekvensspekteret
17	Rate for hvor ofte signalet har negativ verdi
18	Rate for hvor ofte høy-passdelen av signalet har negativ verdi
19	Spektralenergi fra frekvens 0 til tredje formant (F3)
20	Spektralenergi fra F3 til Nyquist-frekvensen (samplingfrekvens/2)

Tabell 2.3: Oversikt over de 20 initielle akustiske egenskapene.

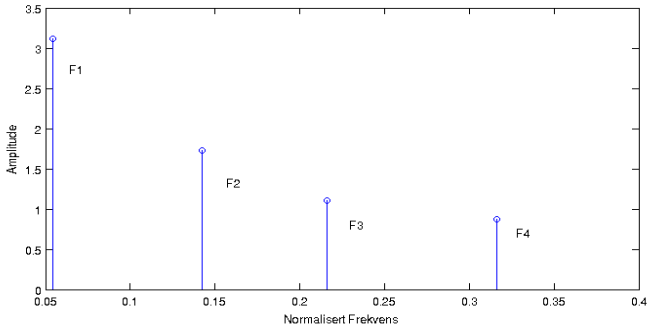
blant annet at formantfrekvenser for enkelte lyder er svært dårlig definerte og er dermed problematisk å bestemme [13], [8].

Formantfrekvensene er høyereordens harmoniske av grunnfrekvensen og opptrer som topper i spektrumet. Forskere har brukte flere metoder for å analyse formanter, blant annet lineær prediksjonsanalyse og utrekning av topper i et kepstralglattet spekter. Feil tolkning av formanter kan ha store negative effekter på gjenkjenningen [13]. I [13] har forfatterne oppnådd gode resultater ved å kombinere formanter og MFCCs i en egenskapsvektor.

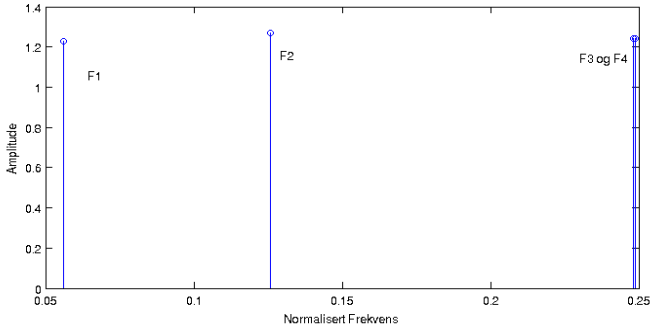
Analyse av Lineær Prediksjonskoding (eng: Linear Predictive Coding - LPC) er ofte brukt i beregning av formantfrekvenser. Metoden som er benyttet i denne oppgaven er hentet fra voicebox-prosjektet [14]. Formantkandidatene blir sporet ved å bestemme røttene i et LPC-polynom av orden 12.

Figur 2.2 og 2.3 gir en illustrasjon av to forskjellige formantmønstrer. Formantene skal ideelt ha god seperasjon i frekvens og mindre amplitude for høyere frekvenser. Figur 2.2 er et eksempel på *gode* formanter, mens figur 2.3 viser heller dårlig definerte formanter.

Grad av stemthet er et meget brukbart mål for talekvalitet i talegjenkjenning. Den mest fundamentale distinksjonen mellom lyder i tale er om de er stemte eller ikke stemte [8]. Et talesegment som er *stemt* har en mer regelmessig tids- og frekvensstruktur, noe som skiller seg fra ustemte lyder. Stemte lyder er normalt mer energirike enn ustemte. Graden av stemthet er



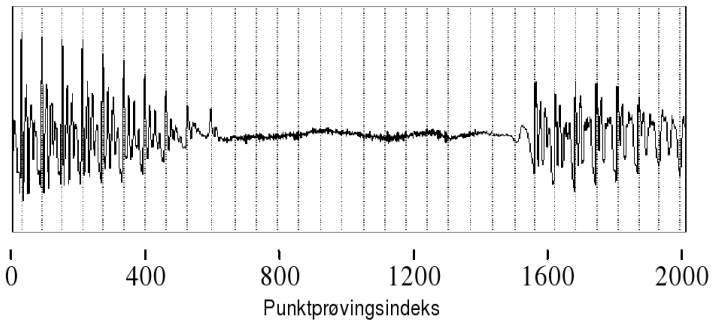
Figur 2.2: De fire første formantfrekvensene for en ramme fra vokalklasse. Figurene viser at formantene er godt definert; amplitudene er høyere for lavere frekvenser og har en tydelig avstanden i frekvens



Figur 2.3: De fire første formantfrekvensene for en ramme fra frikativklassen. Figurene viser at formantamplitudene itilfredsstillendeillende.

et mål på hvor periodisk et talesignal er innenfor et gitt tidsrom [15].

Beregningen av denne egenskapene er beskrevet i [16]. Forfatteren av dette dokumentet baserer uttrekning på analyse av underharmonisk-til-harmonisk-raten (eng: Subharmonic-to-Harmonic Ratio - SHR).



Figur 2.4: Figuren viser tre tydelig skilte deler. Den første er stemt, deretter går den over i ustemt for så å ende opp stemt igjen. Periodisiteten og energien er klart høyere i de stemte delene.

Grunnfrekvens er en ofte brukt parameter i taleanalyse. Når luften beveger seg gjennom taleapparatet vil den måtte passere stemmebåndene (stemmeleppene) i lufterøret. Hvis stemmebåndene er foldet mot hverandre vil luften likevel forsøke å presse seg gjennom. Det medfører i at leppene åpnes og lukkes (vibrerer) med en bestemt frekvens. Denne frekvensen er defineres som antall åpne-lukkesykluser per sekund og har fått navnet grunnfrekvens (F0). I mange tilfeller brukes istedet pitsj som er den inverse av F0. Pitsj er den minste repeterende enheten i talen.

F0 er den egenskapen som bidrar mest i teknikker for å skille stemte og ustemte lyder.

Metode for uttrekning av grunnfrekvens som er brukt er også hentet fra [17] og dokumentert nærmere her [16]. I denne metoden, Pitch Determination Algorithm (PDA), estimeres pitsj (og dermed F0) ved spektrumskifting på logaritmisk skala og beregning av SHR.

Spektralflathet er et mål for flathetskonturen til frekvensspekteret. Det er igjen et mål for periodisitet i talesegmentet [18].

Denne egenskapen er introdusert i [18] for detektering av pitsj og stemthet. For denne oppgaven vil en undersøke dens eget potensial i detektering av taleattributter.

Først finner en spekteret ved å beregne talesegmentets Fouriertransform:

$$X(f) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi fn/N} \tag{2.5}$$

, hvor x er det tidsdiskrete talesignalet, n er punktprøvingsnummer og f er frekvens. Deretter finner man verdien av flathet, SF:

$$SF_i = \sum(\log(X(f))) - \log(\sum(X(f))) \tag{2.6}$$

, hvor i er rammenummer.

Svingning i frekvensspekteret gir en verdi som beskriver svingninger i spekteret innen for en tidsramme. Denne egenskapsparameteren vil også gi informasjon om konturen i spekteret og periodisitet.

Midlet svingning for en ramme beregnes slik:

$$SV_i = \log(\sum(|X(f_n) - X(f_{n-1})|)) \tag{2.7}$$

, hvor n er frekvensindeksen og i er rammenummer.

Nullkryssingsrate (eng: Zero Crossing Rate - ZCR) er et mål for verdiendring i tidsfunksjonen til et talesignal. For korte tidsrammer kan den ses på som et mål for deres frekvensinnhold [19].

Relasjonen mellom ZCR og tidsfunksjonen eller autokorrelasjonsfunksjonen til tale er definert i [19] som:

$$ZCR = k_0 \sqrt{\frac{\overline{f'(t)^2}}{f(t)^2}} = k_0 \sqrt{\frac{-\theta''(0)}{\theta(0)}} \tag{2.8}$$

, hvor f er tidsfunksjonen og $\theta(0)$ er autokorrelasjonsfunksjonen $\theta(\tau)$ for $\tau = 0$.

For denne oppgaven er det trukket ut to ulike egenskaper for ZCR, én for hele spekteret og én for kun den høypassfiltrerte delen (over 3000 Hz).

Energi til talesignalet er mye brukt i analyse av tale. For eksempel ved betraktning av formanters plassering i spektrogram, vil høy energi rundt formantfrekvensene fremheve formantenes posisjon. For denne oppgaven vil det benyttes to egenskaper basert på energi. Den første er signalenergien for frekvenser mellom null og tredje formant (F3), mens den andre tar for seg frekvenser fra F3 til Nyquist-frekvensen (punktprøvingsfrekvens/2).

Energien regnes ut ved å summere alle absolutte amplitudeverdiene for det valgte tidsintervallet:

$$E = \sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \tag{2.9}$$

, hvor $x(n)$ er den diskrete tidsfunksjonen og $X(f)$ er den diskrete fourier-transform (DFT) med frekvenskomponenter f. Formelen er hentet fra [20].

2.5 Statistiske Modeller

Forskere har observert at stokastiske variabler fra fysiske eksperimenter ofte har en tilnærmet gaussiske fordeling. Dette gjelder også menneskers tale. Man kan derfor anta at egenskaper trukket ut fra tale er fordelt tilnærmet slik.

Måten en utnytter denne antagelsen er ved modellering av statistiske fordelingsmodeller på bakgrunn av kjent talematerial. *Kjent* vil her bety at man har såkalte *etikettfiler* (eng: label files) som på forhånd har kartlagt den fonetisk informasjon knyttet til talefilene (se kapittel 3.1). Modellene skal fremstille hvordan de ulike taleklasser en ønsker å gjenkjenne er statistisk fordelt. Høy distinksjon mellom modellene betyr at klassene har svært ulike egenskaper og vil dermed lettere kunne klassifiseres. Dette er dessverre sjelden tilfellet for tale, noe som øker vanskelighetsgraden for klassifisering og kompleksiteten til gjenkjennerne.

Variasjon i talen gjør at én enkel fordeling ikke klarer å modellere en

tilfredsstillende tilnærmelse av de faktiske dataene. Dette løses ved å innføre en modell bestående av flere vektede fordelingskomponenter. Denne teknikken kalles *multi-dimensjonale gaussiske blandingsfordelinger (eng: Gaussian Mixture Models - GMM).*

Ved gjenkjenning av hele segmenter (f.eks fonemer) med ulik varighet brukes ofte *skjulte markovmodeller (eng: Hidden Markov Model - HMM).* Denne metoden består av flere sammenkoblete tilstander med hver sin fordelingsmodell. Ved hjelp av en fler-tilstandsmodell vil en kunne beherske modellering av tidsdynamikken i segmentet. I tillegg unngår en ved slik modellering de usikkerhetsmomentene som opptrer i transisjonsområdet mellom segmenter. Her kan nemlig den fonetiske informasjonsmengden være svært begrenset.

For denne oppgaven er det blitt bestemt at gjenkjenneren skal jobbe med rammer på 10 ms til fordel for segmenter med ulik varighet. Slike rammer ses på som korte statiske segmenter, hvor tidsdynamikk ikke har noen funksjon. Dette er fordi en ramme i starten mot en ramme i slutten av et fonem kan ha ulik egenskapsverdier. Ved å inkludere egenskaper for tidsdynamikk vil det kunne gi en uønsket reduksjon i korrelasjon innad i en taleklasse. En rammegjenkjenner realiseres i dette arbeidet ved en degenerert HMM med kun én enkel tilstand.

2.5.1 HMM vs GMM

GMM består av et endelig antall gaussiske fordelingskomponenter som tilsammen utgjør en tilnærming av fordelingen til en taleklasse. Under trening vil hver klasse få estimert en slik modell på bakgrunn av treningsdata. En klassifiserer for talesegmenter basert på GMM har som oppgave å klassifisere et segment til en av de trente modellene.

HMM er, som nevnt ovenfor, oppbygd av flere sammenkoblete tilstander. Hver tilstand genererer en utgangsverdi basert på sannsynlighetstetthets-funksjonen til den aktuelle tilstanden. Denne funksjonen er ofte en funksjon av flere gaussiske blandingsfordeler (GMM), men kan også være basert på andre statistiske funksjoner. Ved bruk av flere enn én tilstand vil HMM klare å modellere variasjon i tidsløpet. Dette er meget effektivt ved gjenkjenning av hele segmenter da en ikke behøver å ta hensyn til eventuelle

mindre karakteristiske verdier i transisjonsområdet. Til gjengjeld oppstår en problematikk ved deteksjon av overgangene mellom segmenter.

Ved gjenkjenning på rammenivå vil en fler-tilstands HMM ikke ha noen merkbar påvirkning. I denne oppgaven har en, som nevnt, brukt en én-tilstands HMM nettopp på bakgrunn av bruk av rammer.

2.5.2 Gaussiske Blandingsfordelinger

Modellering av tale er helt essensielt for et gjenkjenningssystem. Modelleringsprosessen vil forsøke å utforme en modell av sannsynlighetsfordelinger basert på innhentet data. Gaussisk modellering motiveres av at akustiske egenskaper fra tale har vist seg å være tilnærmet gaussiske og at artikulatorisk karakteristikk i talen kan klassifiseres vha gaussiske tetthetsfunksjoner [21].

Gaussiske blandingsfordelinger er et sett med gaussiske fordelingsfunksjoner, som defineres med parameterne varians, middelverdi og fordelingsvekt. I modellering av data med høy karakteristisk variasjon, vil man med flere fordelinger oppnå modeller som i større grad oppnår en tilnærmelse av de faktiske data.

Modellens sannsynlighetstetthetsfordeling er summen av K vektede fordelinger.

$$p(\mathbf{x}) = \sum_{k=1}^{K} c_k p_k(\mathbf{x}) = \sum_{k=1}^{K} c_k N_k(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.10}$$

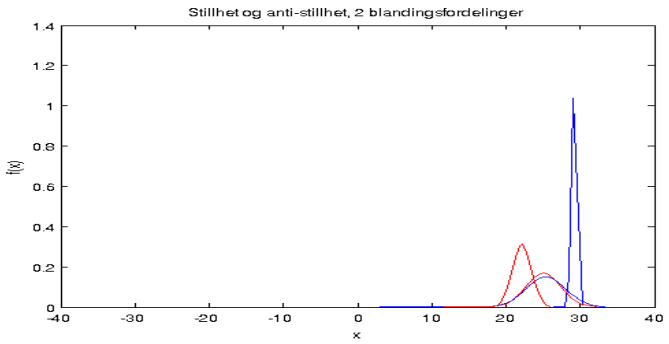
, hvor $\mathbf{x}$ er en fler-dimensjonal vektor. c_k er vekten av fordelingskomponent k, hvor $k = 1, ..., K$. $\boldsymbol{\mu}_k$ er middelverdi og $\boldsymbol{\Sigma}_k$ er kovariansmatrisen til fordeling k [8].

Fordelingsfunksjonen for hver komponent er definert

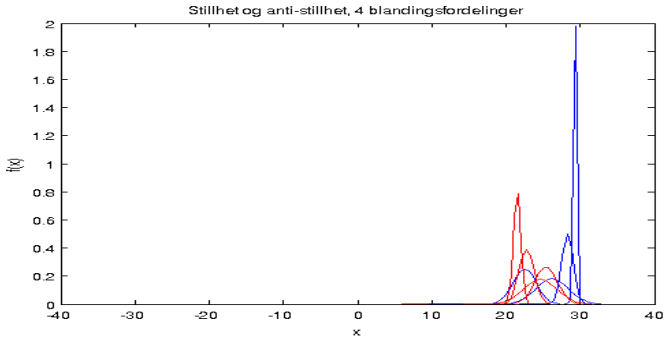
$$p(\mathbf{X} = \mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = N(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right] \tag{2.11}$$

Figurene 2.5, 2.6 og 2.7 viser ulike typer modeller. De to førstnevnte er modeller for stillhet bestående av henholdsvis to og fire blandingskomponenter. Det første diagrammet gir et godt bilde av hvordan attributtklassene kan fordele seg. Både klassen for stillhet og ikke-stillhet har én

fordeling som i stor grad skiller seg ut. I tillegg har de en tilnærmet lik fordeling. Ved en økning av antall komponenter vil en teoretisk oppnå en mer nøyaktig modell. Figur 2.6 viser hvordan fire komponenter skal gi en bedre tilnærming enn for to komponenter.

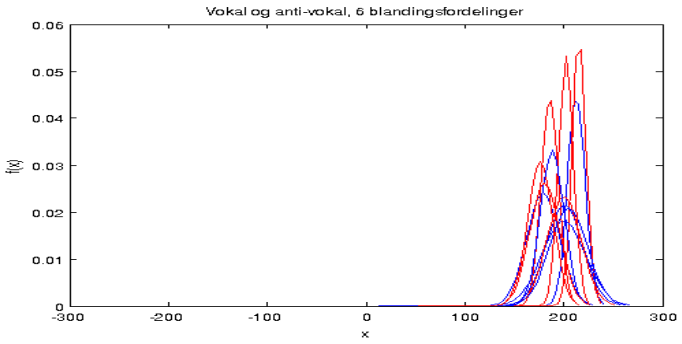


Figur 2.5: GMM med to fordelingskomponenter. Fordelingensfunksjonene er beregnet på egenskapen for energi mellom tredje formant og Nyquistfrekvensen. Blå graf viser fordelingsmodellen for stillhet, mens rød er modellert for ikke-stillhet.



Figur 2.6: GMM (fire blandingskomponenter) for stillhet og parameteren energi (F3-Nyq).

Den gaussiske modellen i figur 2.7 eksemplifiserer en modell og anti-modell som er meget lik hverandre. Dette er noe en ikke ønsker da det vanskeliggjør klassifiseringen. Ut i fra diagrammet ser nullkryssingsrate ut til å være lite egnet for å skille vokaler fra resten av attributtene.



Figur 2.7: GMM for vokal (blå) og anti-vokal (rød) med seks blandingsfordelinger. Egenskapensparameteren som er modellert er nullkryssingsrate beskrevet i 2.4.1

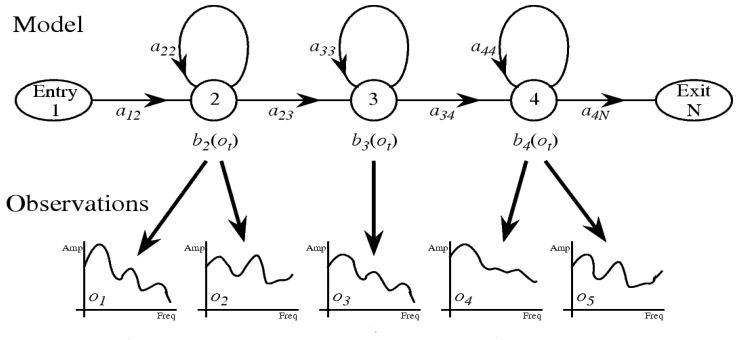
2.5.3 Estimering av Modellparametere

Det er meget viktig at parameterverdiene til en GMM blir estimert slik at de på best mulig måte svarer til fordelingen til treningsdataene. Det finnes flere teknikker for slik beregning [22], men den mest foretrukne er sannsynlighetsmaksimeringsestimering (eng: Maximum Likelihood Estimation - MLE). MLE forsøker å finne de modellparameterne som maksimerer sannsynligheten til modellen i henhold til treningsdataene. For estimeringen anvendes den iterative forventningsmaksimeringsalgoritmen (expectation-maximation - EM).

2.5.4 Skjulte Markovmodeller

En skjult markovmodell (HMM) er en stokastisk prosess hvor et tidsdiskret signal genereres vha en serie med sammenkoblet tilstander [23]. For hver tidsluke vil modellen skifte tilstand i henhold til et sett med transisjonssannsynligheter. Hver tilstand, bortsett fra de passive (første og siste), genererer da en observasjonsvektor på bakgrunn av tilstandens sannsynlighetsfordeling. Disse fordelingene kan være gaussiske blandingsfordelinger, som beskrevet i seksjon 2.5.2.

Figur 2.8 viser en 3-tilstands HMM med tilhørende genererte observasjoner. For hver av de aktive tilstandene har man transisjonssannsynligheter a_{ij} og fordelingsfunksjon for observasjonsvektorene $b_j(o_t)$. a_{ij} beskriver sannsynlig-



Figur 2.8: Eksempel på en 3-tilstands HMM [23]

heten for at modellen skal skifte tilstand fra i til j. Matematisk beskrevet slik

$$a_{ij} = P(m(t+1) = j | m(t) = i) \tag{2.12}$$

, hvor $m(t)$ er tilstanden man er i ved tid t.

Fordelingen av observasjoner produsert av tilstand j, er beskrevet med $b_j(o_t)$ og forteller hvor stor sannsynligheten er for at nettopp *denne* tilstanden genererte observasjonen o_t. Man vet nemlig ikke *hvilken* tilstand som er opphav til vektoren, derfor navnet *skjulte* markovmodeller.

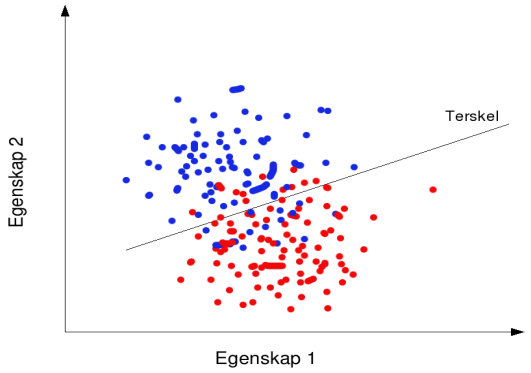
Skjulte markovmodeller er ofte favorisert fremfor alternative metoder. Dette skyldes først og fremst at en HMM klarer å modellere inn variasjon i tid, som beskrevet tidligere i kapittelet. I tillegg finnes det svært effektive algoritmer for estimering av modellparameterne, noe som effektiviserer arbeidet og senker krav til prosessorkraft.

2.6 Deteksjon

Felles for alle gjenkjennere er at de forsøker å plassere observasjonsdata i distinktive klasser (ofte referert som statistiske modeller). Normalt brukes markovmodeller til dette formålet. Tradisjonell bruk av skjulte markovmodeller innebærer å klassifisere hvert talesegment i klasser. Deteksjon forsøker å senke kompleksitet ved å begrense seg til binært utfallsrom, dvs bare to klasser. Den ene klassen er for eksempel vokaler, mens den andre er

alle andre. For hver taleramme vil man da kunne beregne sannsynligheten for tilhørigheten til hver av de to klassene. Forholdet til deres logaritmiske verdier vil da teoretisk gi en sterk indikasjon på hvilken klasse man hører til.

Figur 2.9 gir en enkel illustrasjon for deteksjon. Vi tenker oss at prikker med blå farge hører til en klasse og rød til en annen. Hvert *individ* i hver klasse har fått en verdi basert på deres egenskaper. Denne verdien tilsvarer plasseringen i figuren. Terskelen er blitt beregnet på forhånd på grunnlag av observerte data. Man antar da at alle prikker (individer) over terskellinja tilhører den røde klassen, mens alle under er fra den blåe. I figuren ser man at noen røde vil bli detektert som blå og omvendt. Dette vil da selvsagt gi feil gjenkjenning. Dette er veldig typisk for talegjenkjenning da stor variasjon og usikkerhet preger taledataene. Det er derfor veldig viktig at man klarer å lage så stor avstand mellom klassene som overhodet mulig. F.eks er valg av rette egenskapsvektorer svært avgjørende.



Figur 2.9: Illustrasjon av deteksjon av to klasser, rød og blå.

2.6.1 Metoder for Beslutningsrate

For å kunne bruke deteksjon må man først beregne terskelverdier for detektorene. Detektorene er primitive på den måten at de baserer sine beslutninger kun ut fra en terskelverdi. Det er da svært viktig at man har en terskel som på best mulig måte representerer grenseverdien mellom to klasser, slik som i Figur 2.9. To mye anvendte teknikker for å beregne denne verdien er Minimum Feilrate (eng:Minimum Error Rate - MER), og Lik Feilrate (eng:Equal Error Rate - EER).

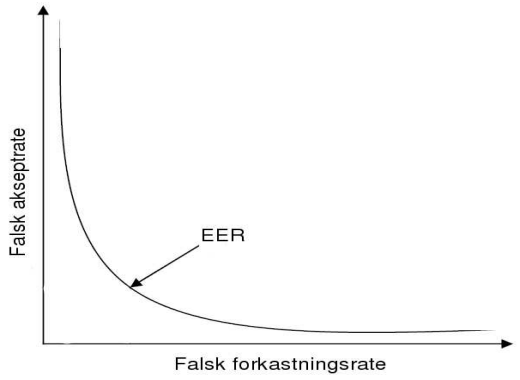
Formelen for MERs terskelverdi er

$$\Theta_{MER} = \arg\min_{\Theta}(FAR(\Theta) + FRR(\Theta)) \tag{2.13}$$

, hvor FAR (*False Acceptance Rate*) er falsk akseptrate og FRR (*False Rejection Rate*) er falsk forkastningsrate. FAR er andel observasjoner i prosent som man ikke skulle ha akseptert, mens FRR er de man burde akseptert, men som man feiltolket og forkastet. MER gir da en terskel ut fra den minst summen av de nevnte prosentverdiene. EER, i Formel 2.10, forsøker å finne den terskelen som skal gi minst avvik i prosent mellom FAR og FRR.

$$\Theta_{EER} = \arg\min_{\Theta}(|FAR(\Theta) - FRR(\Theta)|) \tag{2.14}$$

Figur 2.10 illustrerer hvor den optimale terskelverdien for EER ligger i forhold til FAR og FRR.



Figur 2.10: Terskel for EER mhp FAR og FRR.

Kapittel 3

Hjelpeverktøy

Dette kapittelet gir en kort beskrivelse av de hjelpemidlene som er brukt i arbeidet. For å kunne trene opp gjenkjenneren må en passende taledatabase velges. Ved valg av taledatabase er det svært viktig å velge en som tilbyr et rikt og bredt utvalg av karakteristiske egenskaper i henhold til det aktuelle bruksområdet. I dette arbeidet skal man jobbe med kontinuerlig amerikansk-engelsk tale og gjenkjenne korte taleenheter. TIMIT tilbyr et stort antall setninger fordelt på de største amerikanske dialekter og databasen har god fonetisk merking. Hidden Markov Model Tool Kit (HTK) er dagens mest brukte verktøy i talegjenkjenning og ble derfor et naturlig valg.

3.1 Taledatabase: TIMIT

TIMIT er en meget anerkjent taledatabase og hyppig brukt blant forsker-miljøer innenfor automatisk talegjenkjenning. Databasen er bygd opp av et rikt antall innspilte setninger med ulik dialektisk bakgrunn (amerikansk-engelsk), varierende sinnstilstander og fra begge kjønn. TIMIT-prosjektet ble opprettet på fullmakt fra *Defense Advanced Research Projects Agency* (DARPA). Initiativtakerne var Texas Instruments (TI) og Massachusetts Institute of Technology (MIT), dermed navnet (TI-MIT).

Databasen er blitt lest inn av 630 ulike personer som representerer de åtte mest utbredte dialekter i USA. Den inneholder, foruten taledata,

fonetisk beskrivelse av hver setning på fonem- og ordnivå og punktprøvingsgrenser for hver taleenhet. Totalt holder den 4620 treningsfiler og 1680 testfiler. Informasjonen som databasen tilbyr er veldig anvendelig og nyttig i gjenkjenningssammenheng. Gjenkjenning på kontinuerlig tale krever store datamengder med stort mangfold og variasjon av talens karakterstikk, slik at den totale lingvistiske informasjon blir størst mulig. TIMIT har rikt mangfold og passer da godt for slik testing.

Databasen er bygd opp av fire filer per ytret setning: én binær taledatafil (*.wav*), to etikettfiler i ASCII-format og én tekstfil for hele setningen i vanlig tekstformat. Etikettfilene beskriver talefilens fonetiske informasjon. Hvert fonem i setningen er merket av i tilhørende *.phn*-fil med fonemets første og siste punkprøvingsnummer. Tilsvarende gjøres for hvert ord i *.wrd*-fila, mens *.txt*-fila innholder hele setningen slik en ville *skrevet* den.

Nedenfor finner en et eksempel på de tre tekstfilene. *X.phn*-fila er strukturert med én linje pr fonem med tilhørende punktprøvingsnummere for start og slutt. For ord (*X.wrd*) har en samme type oppsett, men oppdelt i ord istedenfor fonemer. *X.txt*-fila presenterer til slutt setningen i sin helhet.

X.phn: start slutt fonem
0 2308 h#
2308 4469 sh
4469 5639 iy
5639 6433 hv
6433 8158 ae
8158 8754 dcl
8754 9357 jh
. . .
. . .
. . .
42722 44728 ao
44728 45730 l
45730 47501 y
47501 49339 ih
49339 50801 axr
50801 52941 h#

X.wrd:
2308 5639 she
5639 9357 had
. . .
. . .
. . .
42722 45730 all
45730 50801 year

X.txt:
0 52941 She had your dark suit in greasy wash water all year.

3.2 Programvare: HTK

HTK er et meget kraftig og anvendelig hjelpemiddel i forskning der skjulte markovmodeller skal benyttes. Det er spesielt designet for talegjenkjenning, men har i tillegg blitt brukt i talesyntese og andre typer gjenkjenning.

HTK ble utviklet på Cambridge University i 1989. Siden den tid har det byttet eiere ved et par anledninger før Microsoft kjøpte rettighetene til prosjektet i 1999 og gav tilbake lisensrettigheter til den opprinnelig avdelingen på Cambridge University. Cambridge driver nå utviklingsstøtte og distribusjon, med støtte fra Microsoft.

HTK er et rammeverk bestående av biblioteksmoduler og andre skript skrevet i C. Disse er meget profesjonelle analyse-, trening og gjenkjennings-verktøy som blir brukt av majoriteten av forskningsmiljøer innen tale-gjenkjenning.

HTK kan, som de fleste talegjenkjenningssystemer, oppdeles i to hoveddeler, trening- og test/gjenkjenningsdel. Tabell 3.1 beskriver kort noen av de viktigste funksjonene. Felles for alle funksjonene er at de har flere bruks-områder avhengig av inngangsparameterne. Beskrivelsen tar kun for seg det som har vært av nytteverdi under arbeidet for hovedoppgaven. I dette arbeidet er det bare treningsdelen av HTK som er anvendt, og derfor vil beskrivelsen av gjenkjenneren være begrenset.

Navn	*Beskrivelse*
HCopy	HCopy er en effektiv metode for konvertering av taledata til HTK-format. Den er kompatibel med flere typer inndata, mens utgangsfilene alltid vil være på HTKs eget format da de resterende funksjonene ikke er kompatible med andre format. HCopy støtter blant annet lesingen av bølgeformdata fra TIMIT. Den har også mulighet til å redigere etikettfilene til mer HTK-vennlig format.
HCompV	HCompV brukes til å beregne de globale middel- og kovariansverdiene for treningsettet. Disse verdiene blir så brukt som initialverdier til de gaussiske modellene for en HMM.
HRest	Denne funksjonen gjør nye estimeringer og erstatter parameterne funnet vha HCompV. I motsetning til HCompV som bruker *hele* treningsettet, skiller HRest mellom de valgte taleenhetene vha etikettfiler. På den måten får modellene verdier basert på taledata som tilhører den aktuelle attributten. Estimeringen gjøres av en Baum-Welch-algoritme.
HLEd	Redigeringsskript for etikettfiler. Det kan for eksempel brukes til å slå sammen et sett med etikettfiler til én stor fil, som kalles *Master Label File (.mmf).* Man redigere selve innholdet etter eget formål.
HHEd	HHEd er et redigeringsverktøy for manipulering av markovmodeller, hvor man for eksempel kan klone modelldefinisjoner eller øke antall blandingsfordelinger.
HList	HLists funksjon er å lese datafiler og liste opp egenskapene deres. Man bruker den til å undersøke taledata og for å bestemme konfigurasjonsparametere for blant annet konverteringer, men også på et senere stadie til å kontrollere at dataene er ble prosessert på korrekt vis.

Tabell 3.1: Oversiktstabell over viktige funksjoner fra HTK.

For gjenkjenningsdelen kan det nevnes at de to viktigste skriptene er HVite og HResults. HVite er en Viterbi-basert gjenkjenner, hvor testdata blir beregnet opp mot modeller fra trening og gir ut resultat i form av etikettfiler. HResults setter etikettfilene fra HVite opp mot de originale etikettfilene og presenterer gjenkjenningsresultatet.

Informasjon om HTK er hentet fra *"The HTK Book"* [24].

Kapittel 4

Utvikling av Simuleringsverktøy

Oppgaven krever at en utvikler et verktøy for simulering. I første omgang var det planlagt et perl-basert rammeverk som skulle benytte metoder fra HTK. Systemet skulle blant annet fungere med forskjellige antall blandingsfordelinger, akustiske og MFCC baserte egenskapsvektorer og støtte aktuelle feilratemetoder. Systemet skulle også baseres på deteksjon av enkeltrammer. Dette viste seg vanskelig å kombinere med HTKs verktøy for gjenkjenning. For overkomme dette ble det bestemt å bruke HTK i treningsfasen og Matlab-funksjoner i gjenkjenningsfasen. Systemet i helhet består da av et rammeverk av perl-skript som eksekverer henholdsvis HTK- og Matlabfunksjoner.

HTK har et eget format for taledata og støtter i utgangspunktet ikke egendefinerte kildefiler. Derimot kan en ved hjelp av deres funksjon, HCopy, konvertere tale på bølgeform til HTK-format. TIMITs taledatabase er på bølgeform som støttes. Ved bruk av MFCCs, har HCopy blitt anvendt direkte for konvertering til HTKs MFCC-format. I tillegg skulle oppgaven benytte egenkomponerte 20-dimensjonale egenskapsvektorer basert på artikulasjon. Dette ble gjort ved å manipulere dataene slik at HTK måtte prosessere dem som om de skulle vært på deres eget format.

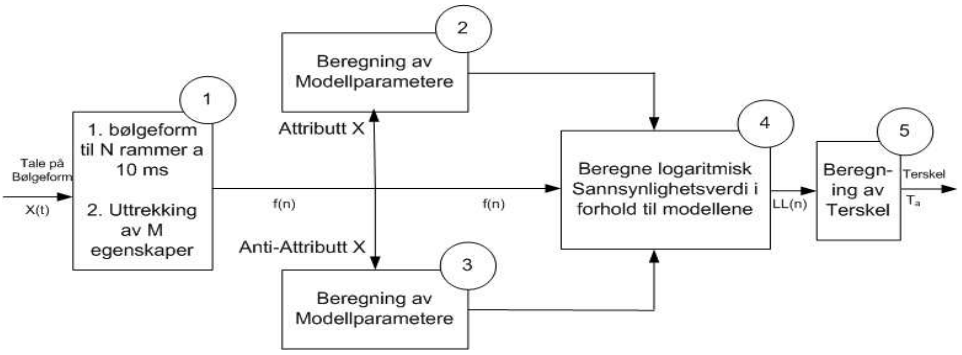
Implementasjonen er bygd opp i to blokker, treningsdel og gjenkjenningsdel. Treningsdelen består grovt sett av prosessering av treningsdata, opprette statistiske modeller, kalkulere sannsynlighetsverdier per ramme basert på

modellene og finne terskelverdier til detektorene. Gjenkjenningsdelen, eller testdelen, består av dataprosessering av testdata, kalkulering av sannsynlighetsverdier per ramme basert på modellene fra trening og beregning av gjenkjenningsraten. Systemet prosesserer hver attributt hver for seg gjennom begge blokkene.

4.1 Trening

Både trenings- og testdelen tar for seg én og én attributt av gangen. For enkelhetsskyld vil attributten som prosesseres heretter navngis attributt X. For hver attributt trenes én statistisk modell for data tilhørende den aktuelle attributten og én for alle andre (anti-attributt X).

Prosessen med å beregne modellparameterne blir utført av HTKs funksjoner HCompV og HRest, med hjelp av HLEd og HHEd. HLEd brukes i forkant for å endre etikettfilene i henhold til attributt X, mens HHEd anvendes for endring av HMM-parameterne.

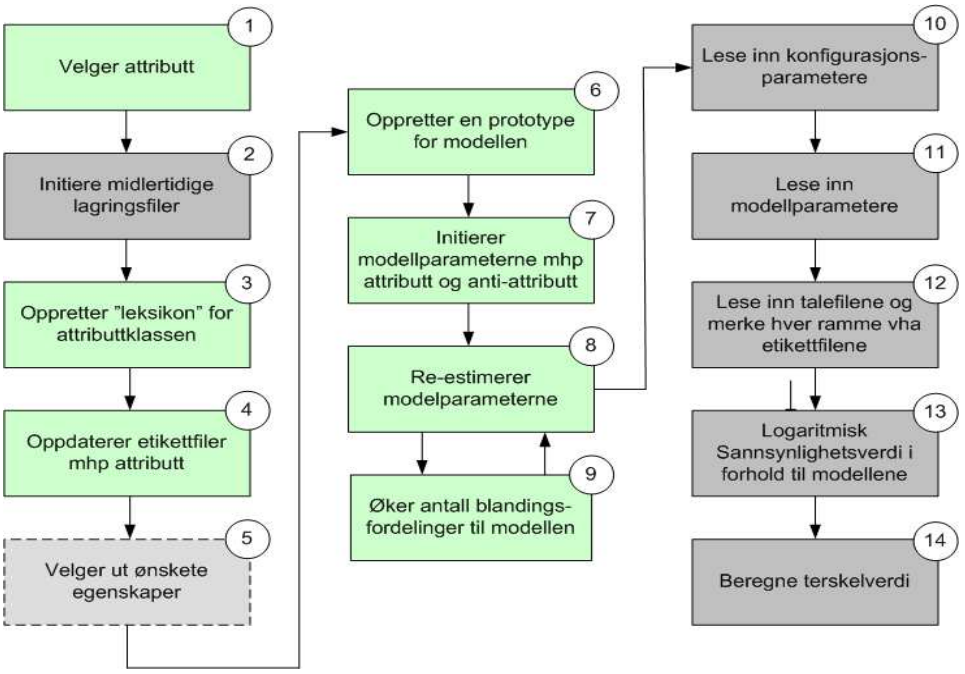


Figur 4.1: Blokkskjema for treningsdelen.

Figur 4.1 gir en overorden beskrivelse av systemets treningsdel. Blokk 1 vil transformere tale på bølgeform til N rammer av varighet 10 ms. For hver ramme beregnes verdier knyttet til hver egenskap redegjort i kapittel 2.4.1. Utgangen av blokk 1 vil derfor være N antall rammer med tilhørende M antall egenskapsparametere. Boks 2 vil så kalkulere den logaritmiske sannsynlighetsverdien til rammen vha fordelingsfunksjonen gitt av de trente modellene.Den siste blokka av treningsdelen sammenligner hver ramme fra blokk 2 mot deres etiketterverdi. På bakgrunn av en bestemt

beslutningsstrategi for feilraten velges den beste estimerte terskelverdi. Beslutningsmetoder er beskrevet i kapittel 2.6.1. I dette arbeidet har en valgt å benytte *lik feilrate (EER)*.

Figur 4.2 viser flytdiagram for trening. Diagrammet er ment for å illustrere rekkefølgen som systemet eksekverer de forskjellige modulene. De blokkene som er fylt med lysegrønn (lys) farge er prosessert av HTK, mens de gråe (mørke) blokkene er Matlab. Boks 5 er det Matlab-skriptet som velger ut egenskaper for deteksjonen og har fått stiplet kantlinje ettersom den ikke inngår i alle simuleringer.



Figur 4.2: Flytskjema for treningsdelen.

Beregning av modellparameter (HTK) *Første steg* er å opprette en prototype for modellen. Det kan for eksempel være en én-tilstands HMM med én gaussisk fordelingskomponent og vektordimensjon på 15. Vektordimensjonen er antall parametere for hver komponent. Deretter genereres to tilsvarende HMMer dedikert til attributtklassen og anti-attributtklassen. Dette steget tilsvarer boks 6 i figur 4.2.

I neste steg (boks 7) blir modellparameterne i hver HMM initiert med globale

middel- og variansverdier.

Tredje steg tar de initierte HMMene og gjør nye estimatberegninger med hensyn på hver attributt. Det vil si istedenfor global beregning av all data, trekker en ut (vha etikettfiler) de rammene som tilhører attributten og estimerer nye modellparametere. Det samme gjøres for anti-attributtklassen. Steget er plassert i boks 8 i figur 4.2.

Boks 9 vil øke antall blandingskomponenter til en GMM. For hver økning må en tilbake til boks 8 for å estimering parameterne på nytt.

Beregning av terskelverdi (Matlab) Når modellene er generert ferdig vil flere Matlab-funksjoner beregne seg frem til en terskelverdi for detektoren. Aller først leser boks 10 inn konfigurasjonsparametere bestemt i et tidligere skript. Deretter innleses modellparametere generert av HTK og hver ramme merkes vha etikettfilene. Boks 13 kalkulerer en logaritmisk verdi per ramme basert på dens tilhørighet til modell og anti-modell. Dette vil illustreres matematisk:

For en modell med K komponenter og M egenskaper beregnes den samlede sannsynlighetstettheten for rammevektor $\mathbf{x}$ slik

$$p(\mathbf{x}) = \sum_{k=1}^{K} c_k \prod_{f=1}^{F} p_{f,k}(\mathbf{x}) = \sum_{k=1}^{K} c_k \prod_{f=1}^{F} N_{f,k}(\mathbf{x}|\boldsymbol{\mu}_{f,k}, \boldsymbol{\Sigma}_{f,k}) \tag{4.1}$$

, hvor c_k er vekten til komponent k, $\boldsymbol{\mu}$ er middelverdivektor og $\boldsymbol{\Sigma}$ er vektor for modellens kovarians.

Hvis $p_a(\mathbf{x})$ er sannsynlighetstetthetsverdien beregnet på attributtmodellen, mens $p_{a-a}(\mathbf{x})$ er beregnet på anti-attributtmodellen, vil den logaritmiske differansen være

$$\mathbf{L} = \log(p_a(\mathbf{x})) - \log(p_{a-a}(\mathbf{x})) \tag{4.2}$$

, hvor L er en vektor som inneholder den logaritmiske differansen for hver ramme.

Boks 14 vil så beregne den terskelverdien som gir best rate med hensyn på beslutningsmetoden EER.

4.2 Testing/Gjenkjenning

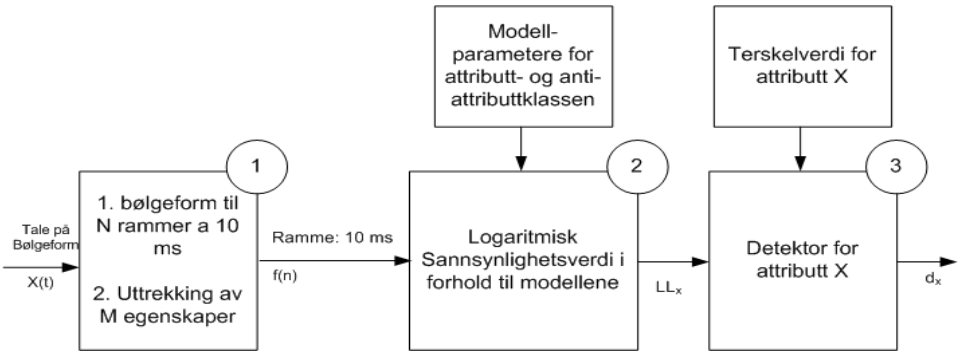
Gjenkjenningsdelen av systemet har som oppgave gjenkjenne taledata vha attributtdetektorene. En detektor besitter informasjon om attributtens trente modeller og dens tilhørende terskelverdi. For hver tidsramme vil detektoren gi en utgangsverdi som representerer dens beslutning. Enten så vil den anta at rammen tilhører den aktuelle attributten eller ikke.

Denne modulen er stort sett skrevet i matlab med unntak av redigeringen av etikettfiler (HLEd fra HTK).

Testdelen er illustrert i figur 4.3. Blokk 1 i figuren er nøyaktig den samme som blokk 1 for trening (se figur 4.1). Blokk 2 bruker modellparametere beregnet i treningsdelen og kalkulerer logaritmisk sannsynlighetsverdier for testdataene. Blokk 3 er selve detektoren. Terskelverdiene funnet i treningsprosessen blir brukt for å bestemme utgangen til hver ramme, enten attributt X eller anti-attributt X:

$$U[n] = \begin{cases} AttributtX & L[n] > Terskel \\ Anti-attributtX & L[n] < Terskel \end{cases} \tag{4.3}$$

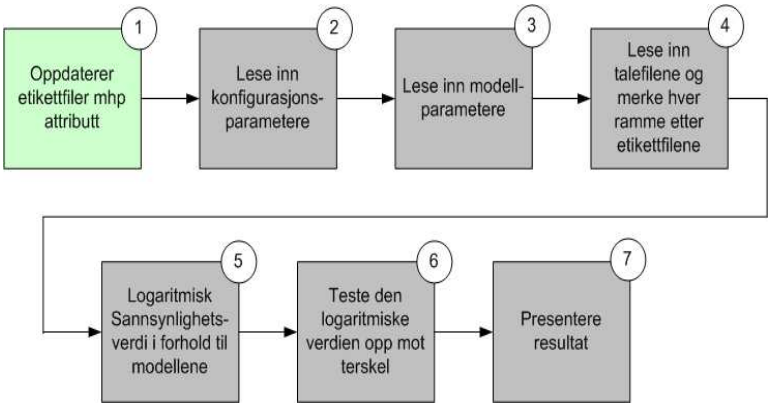
, hvor $U[n]$ er utgangen og $L[n]$ er den logaritmiske differansen for ramme n.



Figur 4.3: Blokkskjema for test-/gjenkjenningsdelen.

Figur 4.4 viser flytdiagram for gjenkjenningen. Den bruker flere av metodene fra trening og er dermed delvis forklart under kapittelet for trening (se kapittel 4.1). Blokk 6 skiller seg ut. Her sjekkes de detekterte rammene mot

deres originale etiketter. På den måten kan en enkelt finne systemet feilrate. Resultatene blir deretter presentert i den siste fasen.



Figur 4.4: Flytskjema for test-/gjenkjenningsdelen.

Kapittel 5

Analyse av Egenskapsvektorer

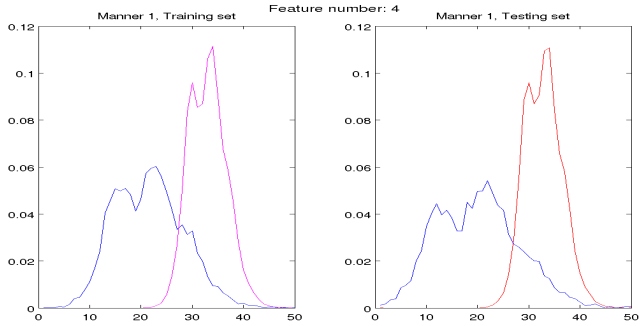
Dette kapittelet vil omhandle de ulike eksperimentene som er blitt simulert og bakgrunn for valg av disse. Det viktigste formålet var å prøve ut egendefinerte akustiske taleegenskaper og undersøke om de har potensial i gjenkjenning av artikulatoriske taleattributter.

For simuleringene er det benyttet et utvalg av de totale datafilene som TIMIT-databasen disponerer. For trening har en trukket ut 305 setninger. De er valgt ut fra ulike dialekter for å inkludere de karakteristiske variasjonene i talen. Med tilsvarende kriterier har 157 testsetninger blitt valgt. Et begrenset talematerial slik som i dette tilfellet kan ha innvirkning på prestasjonen til gjenkjenningsystemet, og er noe en må ta med i betrakning.

Det første avsnittet vil ta for seg de 20 initielle egenskapene gitt av faglærer. Deretter to ulike utvalg av de antatt best egnede for hver av de 14 attributtene. Til slutt en kort presentasjon av MFCC.

5.1 Initielle Egenskaper

Ved oppstart av denne oppgaven ble det gitt 20 egenskaper. De skulle testet og undersøkes for bruk i gjenkjenning av ulike artikulatoriske attributter. De 20 egenskapene er beregnet for hver ramme av talen. På bakgrunn av disse 20 håper en å oppnå høy separasjon mellom attributtklassene som kan resulterer i god feilrate for detektorene. Resultater etter simuleringer



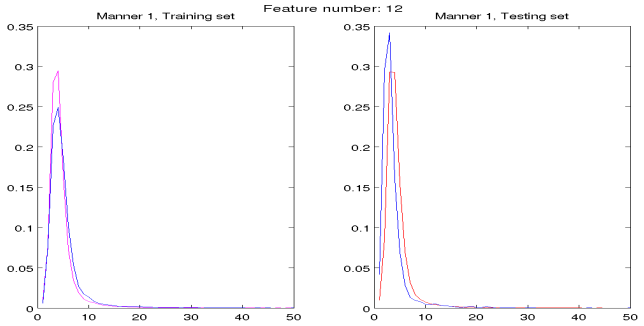
Figur 5.1: Egenskapen F4 (frekvens) er fordelt for approksimanter med blå kurve, mens resten av taleattributtene er fordelt i rødt.

ved bruk av disse egenskapene gir derimot grunn til betenkelighet for deres potensial. For å undersøke dette grundigere er det laget histogrammer som viser fordelingene til egenskapene for henholdsvis attributt og anti-attributt fra både trening- og testsettets talefiler. Analysen av egenskapene er gjort ved subjektiv inspeksjon av histogrammene og bestemmelsene kan derfor ha innslag av menneskelige feiltolkninger. Fordelingene til klassene vil kunne sjekkes manuelt, mens korrelasjon mellom klassene er heller komplisert.

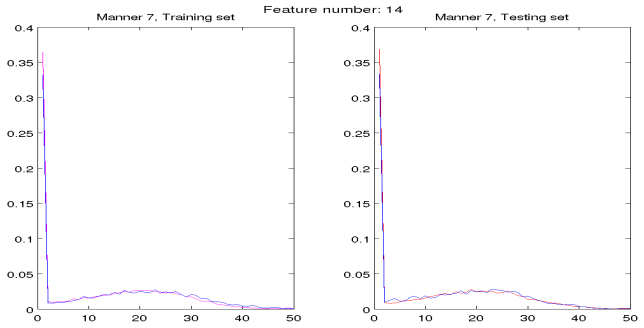
Hvert diagram tar for seg én attributt og én egenskap. Fordelingene for attributt X og anti-attributt X viser hvordan verdiene av *denne* egenskapen forekommer for rammer av attributten kontra anti-attributten. Histogrammene er normalisert i forhold til antall rammer slik at arealet under kurvene er like. Dette vil lette den subjektive betrakningen av prosentvis forhold mellom klassene.

Ved beregning av verdien til den samlede sannsynlighetstetthetsfunksjonen (eng: probability density function - pdf) til en ramme vil en multiplisere pdf-verdiene for hver egenskap og deretter summere for hver gaussiske modell-komponent (se formel 4.1). Et hvert positivt bidrag fra egenskapsvektorene vil øke separasjonen mellom klassene og øke sannsynligheten for korrekt gjenkjenning.

Figur 5.1 er tatt fra attributten *Approksimanter* og egenskapen *frekvens ved fjerde formant (F4)*. Blå kurve følger approksimantrammer, mens rød kurve er resten av rammene. Fra diagrammet ser en at klassenes fordelinger har klare ulikheter. Middelverdien til approksimanter ligger rundt 20 (ikke en



Figur 5.2: Parameteren for båndbredden til fjerde formant (F4) ser ikke ut til å skille approksimanter og resten særlig bra.



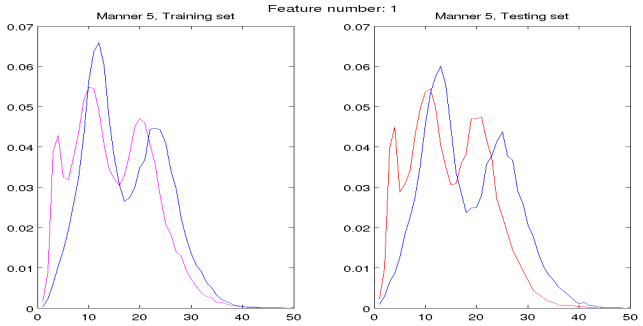
Figur 5.3: Grad av stemthet fordelt for Alveolarklassen og alle andre, henholdsvis blå og rød graf.

fysisk verdi), mens samme målverdi for alle andre ligger i overkant av 30. Fordelingene overlapper noe, men er likevel så distinkte at denne egenskapen kan antas å være egnet for gjenkjenning av approksimanter.

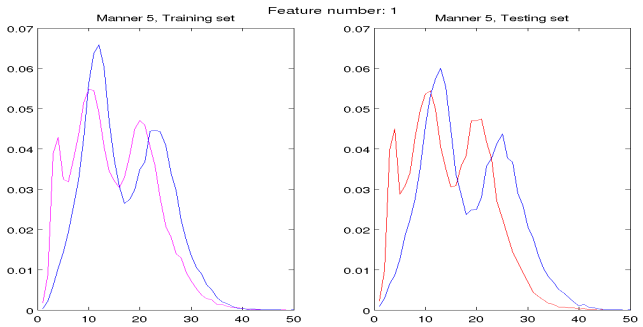
I kontrast til figur 5.1 viser figur 5.2 at ikke alle egenskaper er egnet for detektering av approksimanter. Her er det båndbreddeparameteren for F4 som avslører mindre gode kurver. Det viser seg at *flere* av egenskapene er mindre gode eller ubrukelige. Et annet eksempel er figur 5.3.

Figur 5.3 representerer en meget lite potensiell kombinasjon av attributt og egenskap. Kurvene her er tilnærmet like. Det betyr at egenskapen for stemthet *ikke* er spesielt egnet til å skille Alveolarrammer fra resten av attributtklassene.

Om en tar for seg egenskapen for energi (0-F3) ser en helt klart ulikt potensial



Figur 5.4: Stillhetklassens energimål for frekvenskomponenter mellom 0 og tredje formant (F3).



Figur 5.5: Dentalklassens energimål for frekvenskomponenter mellom 0 og tredje formant (F3).

på tvers av attributter. Histogrammet for *Stillhet* i figur 5.4 viser en god kombinasjon. Det var også forventet da stillhet *ikke* ikke er energirikt. I motsetning gir figur 5.5 et eksempel på en mindre god kombinasjon med attributten *Dental*. Det betyr da at energien til lyder fra *Dental* ikke skiller seg ut fra resten av attributtene.

Dette er bare et utsnitt av de histogrammene som er laget og undersøkt. Flere av kombinasjonene virker å være lite egnet for gjenkjenningen. En samling av histogrammer for attributten *Approksimanter* kombinert med alle egenskapene og histogrammer for egenskapen *energi (F3)* kombinert med alle attributter er lagt ved i tillegget bakerst i rapporten (se tillegg B).

5.2 Utvalg av Potensielle Egenskaper

Analysen av de 20 egenskapene viste helt klart at ikke alle er egnet i deteksjon av alle attributter. For å forbedre systemet har en da forsøkt å finne *hvilke* egenskaper som har positiv effekt ved detektering av *hvilke* attributter. Som allerede nevnt i forrige avsnitt er undersøkelsen gjort ved manuell inspeksjon av fordelingene til egenskapene. Den subjektive oppfattelsen av figurene må da være grunnlag for utvelgelsen av egenskaper.

Simuleringene er fordelt i fire ulike eksperimenter:

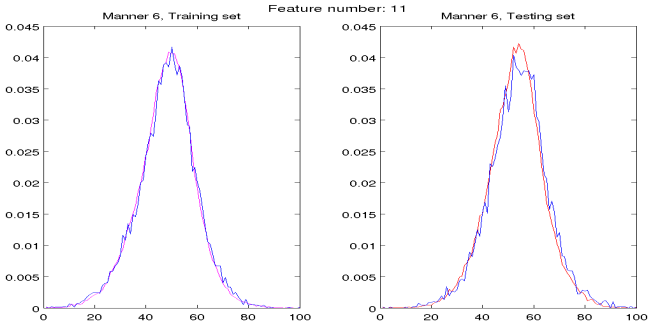
- *Eksperiment 1*: Alle 20 egenskapene brukes for deteksjon av alle typer attributter.
- *Eksperiment 2*: Kun de *høyst potensielle* egenskapene for hver attributt.
- *Eksperiment 3*: Egenskaper fra Eksperiment 2 pluss de egenskaper som er antatt delvis egnet.
- *Eksperiment 4*: MFCC.

Utvalget av egenskaper for eksperiment 2 og 3 er presentert i rapportens tillegg i henholdsvis tabell A.2 og A.1. Tabellene viser hvilke kombinasjoner av akustiske egenskaper og artikulatoriske taleattributter som er trukket ut for eksperimentet.

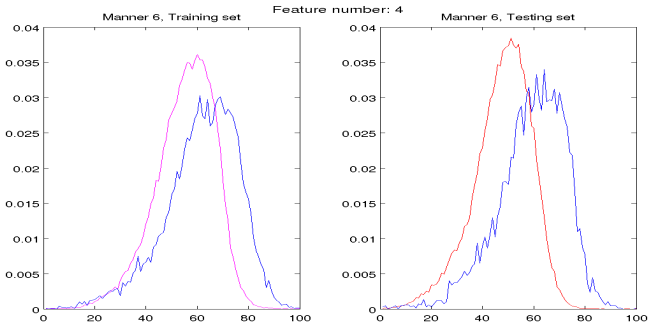
5.3 MFCC

Formålet med oppgaven var å kartlegge bruken av akustiske egenskaper for deteksjon av attributter basert på artikulasjon. I tillegg skulle en se hvordan de stiller seg i sammenligning med tradisjonelle egenskapsvektorer basert på mel-frekvens kepstralkoeffisienter (MFCC). På lik linje som for akustiske egenskaper har noen av koeffisientene blitt trukket ut og for å undersøke deres fordelinger for ulike attributter.

MFCC har vært anerkjent som meget suksessfulle egenskapsparametere gjennom mange år. En forventer dermed *ikke* å oppnå tilsvarende resultater som MFCC ved disse simuleringene.



Figur 5.6: Histogram for fordelingen av ellevte kepstralkoeffisient (MFCC) for *stillhet*. Separasjonen mellom klassene er svært liten.



Figur 5.7: Histogram for fordelingen av fjerde kepstralkoeffisient (MFCC) for *stillhet*. Koeffisienten er fordelt med fin separasjon og skiller klassene.

Feilraten til alle attributtdetektorene gir bedre resultat som forventet. Ved inspeksjon av egenskapsdiagrammene for MFCC viser derimot *ikke* noe tydelig større separasjon i forhold til de akustiske egenskapene. Dette kan skyldes flere grunner. Blant annet kan MFCCs bruk av DCT for dekorrelering være svært avgjørende. Korrelasjonen vil heller ikke kunne oppfattes ved manuell inspeksjon av diagrammene.

Et eksempel på et lite tilfredsstillende fordelingsdiagram for MFCC er gitt i figur 5.6. Fra figuren kan en se at fordelingkurven til anti-attributten følger nesten identiske med attributten. Separasjonen mellom klassene er her svært dårlig.

I motsetning viser figur 5.7 en fin fordeling som gir god separasjon mellom klassene og derfor bidrar positivt til deteksjon av *stillhet*.

Kapittel 6

Resultater

Resultatene fra simuleringene vil her presenteres for de fire eksperimentene, beskrevet i 5.2. Simuleringsverktøyet gir ut feilrate i prosent for hver attributtdetektor. Det har blitt simulert med GMMer med ulikt antall komponenter. Her vil det kun legges frem resultater etter ti blandingskomponenter. Dette skyldes at feilraten ikke er hovedfokus i oppgaven, men kun skal gi en indikasjon på forbedringer ved ulike valg av egenskapsvektorer. I tillegg har forbedringen av feilrate stort sett stabilisert seg før ti komponenter og derfor kan en anta at dette antallet vil være representativt for detektorens feilrate.

Tabell 6.1 gir en oversikt over feilraten til hver detektor for de ulike eksperimentene. I henhold til forventningene viser kolonnen for MFCC en klart lavere feilrate enn deteksjon vha akustiske egenskaper. I snitt ligger de oppunder 15 % lavere. Eksperiment 2 som består av alle kombinasjoner av egenskaper og attributter bortsett fra de antatt svakeste, gir med liten margin den beste gjennomsnittlige feilraten.

En forbedring i feilrate etter utvelgelsen kan tyde på at de forkastede egenskapene faktiske ikke var spesielt egnede, slik en antok. Det gir derfor inntrykk av at utvelgelsen var en riktig operasjon. Eksperiment 2, som gav lavest feilprosent, er ytterligere presentert i tabell 6.2. Tabellen viser feilprosenten for hver attributtdetektor ved bruk av 10-komponents GMM. For hver attributt presenteres prosentvis korrekt deteksjon av attributten,

falsk akseptrate (FAR), falsk forkastningrate (FRR) og korrekt deteksjon av anti-attributten.

	Eksperiment:	*1*	*2*	*3*	*4*
Nr.	*Attributt*	*Alle*	*Utvalg 1*	*Utvalg 2*	*MFCC*
1	Frikativer	31.70 (3.10)	30.76 (3.05)	32.84 (3.78)	12.46 (1.72)
2	Nasaler	19.56 (1.28)	19.32 (1.32)	24.04 (1.52)	13.64 (0.94)
3	Stopp	30.36 (3.24)	30.74 (3.14)	30.25 (3.28)	25.78 (2.74)
4	Vokaler	33.96 (5.31)	28.11 (5.59)	31.72 (4.34)	14.35 (3.45)
5	Approksimanter	37.56 (1.71)	35.63 (1.86)	33.32 (1.63)	20.18 (1.09)
6	Stillhet	12.41 (0.73)	12.33 (0.76)	11.88 (0.69)	9.92 (0.73)
7	Alveolar	39.86 (4.06)	37.57 (4.24)	40.04 (4.06)	21.55 (3.00)
8	Dental	27.62 (0.35)	31.46 (0.39)	36.33 (0.42)	21.15 (0.28)
9	Lab-dent	23.51 (0.56)	20.86 (0.51)	26.91 (0.67)	14.24 (0.40)
10	Labial	31.16 (1.59)	32.51 (1.63)	33.52 (1.61)	29.95 (1.48)
11	Velar	40.14 (1.18)	43.03 (1.38)	44.76 (1.00)	21.83 (0.81)
12	Bak	38.32 (1.39)	38.39 (1.32)	38.09 (1.19)	21.55 (0.98)
13	Senter	41.66 (1.54)	39.53 (1.48)	39.72 (1.52)	18.43 (1.00)
14	Front	41.35 (3.25)	36.63 (3.78)	35.66 (3.94)	17.39 (2.07)
μ	Gj.snitt	33.62	32.33	34.21	18.74

Tabell 6.1: Feilrate i prosent for hver attributtdetektor for de fire ulike eksperimentene. Verdiene omsluttet av parenteser er prosentvis feil fra rammer i transisjonsområdet (to første og to siste rammer til en attributt).

Utvalgte Akustiske Egenskaper - Eksperiment 2

Frikativer	68.69	31.31
	30.64	69.36
Nasaler	75.39	24.61
	18.92	81.08
Stopp	68.48	31.52
	30.59	69.41
Vokaler	72.96	27.04
	28.79	71.21
Approksimanter	60.39	39.61
	35.32	64.68
Stillhet	85.61	14.39
	12.04	87.96
Alveolar	60.79	39.21
	37.10	62.90
Dental	65.46	34.54
	31.42	68.58
Lab-dent	74.28	25.72
	20.69	79.31
Labial	65.14	34.86
	32.30	67.70
Velar	51.55	48.45
	42.66	57.34
Bak	61.89	38.11
	38.42	61.58
Senter	61.47	38.53
	39.62	60.38
Front	65.31	34.69
	37.18	62.82

Beskrivelse:

Verdiene er i prosent (%):
(Type I feil = FAR & Type II feil = FRR)

Korrekt	Type I
Type II	Korrekt

Tabell 6.2: Resultater for hver attributtdetektor i eksperiment 2. Tabellen viser detektorene falsk akseptrate og falsk forkastningrate.

Kapittel 7

Evaluering og Diskusjon

Masteroppgavens mål var å undersøke potensialet til akustiske egenskapsvektorer i deteksjon av artikulatoriske taleenheter. MFCC har vist gjennom en årrekke å være den mest vellykkede måten å representere egenskapsvektorene. En håper likevel å oppnå tilsvarende gode resultater ved andre metoder. Visjonen som gir opphav til denne oppgaven er at taleenheter basert på talens artikulasjon (attributter) skal kunne gjenkjennes på grunnlag av talens akustiske egenskaper, som blir generert av nettopp artikulatorene. Ulike akustiske egenskaper har derfor blitt studert mot de ulike attributtene.

Kapitlene 5 og 6 presenterer henholdsvis utførte eksperimenter og deres resultater i form av feilrate i prosent. Dette kapittelet vil gi en evaluering av de simulerte eksperimentene og diskutere tilhørende resultater.

7.1 Akustiske Egenskapsvektorer

Resultatene presentert i tabell 6.1 bidro med å bekrefte den forventede påliteligheten til bruk av MFCC til fordel for akustiske egenskaper. Deres overlegenhet i detektering av blant annet vokaler er heller mer overraskende. Vokallyder har normalt akustisk karakteristikk distinktiv til konsonanter, deriblant frikativer, stopp og nasaler, som ideelt sett skulle resultere i lav feilrate. Flere av egenskapene er basert på formanter, som vanligvis brukes i gjenkjenning av vokaler. Teknikken for formantuttrekning som er anvendt

i dette arbeidet er hentet fra [14] (se kapittel 2.4.1). I hvor stor grad en kan stole på nøyaktigheten til de lånte funksjonene er vanskelig å si. Diagrammene presentert i kapittel 5, eller samlet i sin helhet i tillegg B, viser tildels tilfeldige fordelinger, noe som kan tyde på at de kan ha påvirket gjenkjenningsystemet negativt.

Tradisjonelt har gjenkjennere hatt problemer med gjenkjenning av *steder* for artikulasjon. Resultatene fra simuleringene viser derimot ikke tilsvarende forhold i feilrate mellom artikulasjonsmåte og -sted. For eksempel har en 20.86 % feil for attributten *Lab-dent*, mens *Vokaler* og *Frikativer* ligger faktisk nesten ti prosent *høyere*.

Bak, *Senter* og *Front* er underklasser av *Vokaler* og ligger på tilsvarende feilrate. *Alveolar* skiller seg svært dårlig fra resten utfra fordelingsdiagrammene og er dermed tildelt få egenskaper. Tabell A.1 viser at Alveolarklassen kun er gitt tre egenskaper som anses som klart egnede for detektoren. Om en er begrenset til et lite antall egenskaper oppnår en ofte at de har for dårlig bakgrunn til å kunne separere en klasse fra andre klasser. Dette kan føre til høy feilrate, slik som for Alveolardetektoren.

Forfatterne av [11] har foretatt undersøkelser med visse likheter til dette arbeidet. Omtrent like attributter er valgt. De har derimot ikke basert seg på akustiske egenskapsvektorer, men på logaritmisk spektralanalyse, lignende MFCC. Resultatene deres er svært gode i forhold til resultatene fra dette arbeidet, men kan ikke uten videre sammenlignes. Taledatabasen de har anvendt, NTIMIT, er en telefonibasert (båndbredde: 8 KHz) versjon av TIMIT og en har dermed ingen forutsetning for direkte sammenligning. I tillegg har de valgt å trekke ut kun de sterkest rammene for hver attributt. Forhold mellom resultatene til de ulike attributtene kan derimot undersøkes opp mot deteksjonsresultatene i 6.1. Det viser seg at gjenkjenningen av flere av artikulasjonsstedene i [11] gir meget dårlig feilrate, mens feilraten for de resterende attributtene gir klart lavere feilrate. Attributtdetektorene gir en mer jevn feilrate på tvers av attributtene, fra 12.3 % til 43 %. Til sammenligning har en i [11] oppnådd 2 % for *Vokaler* og 89 % feil for Dentalklassen. Dette er muligens tilfeldig, men kan også tyde på at de akustiske egenskapene i større grad karakteriserer stedsattributtene enn artikulasjonsmåte.

Kepstralkoeffisientene i MFCC er kjent for å ha en fin gaussisk fordeling. De akustiske egenskapene viste seg å fordele seg svært ulikt. Enkelte av dem er tilnærmet gaussiske og med god separasjon mellom attributtklasser. I kapittel 5 illustrerer figur 5.1 et slikt tilfelle, mens 5.3 viser en kombinasjon av Alveolarklassen og egenskapen *grad av stemthet* som er mindre heldig. Koeffisientene fra MFCC er av logaritmisk verdi for bedre gaussisk tilnærming, noe en ikke har anvendt ved bruk av de akustiske egenskapene. Det er mulighet for at logaritmiske verdier av egenskapene ville kunne senke feilraten til attributtdetektorene. En slik undersøkelse har ikke blitt prioritert i denne rapporten, men er aktuelt for eventuelle senere arbeidsoppgaver.

Teknikkene for utrekning av egenskapene er stort sett hentet fra ulike nettsider. En kan derfor ikke garantere for deres pålitelighet. Det kan være tilfeller der funksjonene genererer verdier som kan virke helt tilfeldig som da gir svært ulike verdier i forhold til naborammer eller verdier som ikke er i nærheten av majoriteten. Slike *"utstikkere"* fører til at treningsdelen av systemet produserer GMMer på feil premisser ved å tildele en tetthetskomponent til verdier som antageligvis *ikke* er representative for klassen. Én slik verdi for hver ramme kan bety *stort* utslag for sannsynlighetsverdien generert på bakgrunn av modellsammenligning i testdelen. MFCC unngår slike ved logaritmiske - og dekorreleringsoperasjoner. Slike feilgenererte rammeverdier kan anses å være en feilkilde ved bruk av akustiske egenskaper og kan være aktuelt å belyse nærmere i en videreføring av arbeidet fra denne rapporten.

Foruten *"utstikkere"* kan et *begrenset* talemateriale produsere utilfredsstillende gaussiske blandingsfordelinger. I tilfeller der en attributt opptrer sjeldent vil treningen basere seg på svært begrenset taledata og en vil kunne se GMMer som ikke klarer å tilnærme de faktiske data på akseptabel måte. Begrensningen fører til at dataene er tilnærmet deterministiske og vil da ikke kunne oppnå en god modell vha verktøy for generering av stokastiske GMMer. For å overkomme dette problemet kunne en forsøkt med deterministiske fordelingsmodeller.

7.2 Deteksjonsrate: Trening vs Test

Gjenkjenningsraten fra trening satt i sammenheng med raten på utgangen av gjenkjenningsverktøyet kan gi et mål på hvor godt verktøyet klarer å trenes på bakgrunn av dataene. Om resultatene er tilnærmet like har en nærmest oppnådd optimal trening mhp tilgjengelig taledata.

Akustiske egenskaper

Antall rammer

	Vokaler		Frikativer	
Trening	27911	11067	13019	5521
	17616	45870	24875	59049
Testing	15362	5692	6488	2957
	9564	23656	13737	31092

Prosent (%)

	Vokaler		Frikativer	
Trening	71.61	28.39	70.22	29.78
	27.75	72.25	29.64	70.36
Testing	72.97	27.03	68.69	31.31
	28.79	71.21	30.64	69.36

Tabell 7.1: Trening- og testresultat (rammer/%) for både Vokaler og Frikativer basert på akustike egenskaper

For enkelthetsskyld har en trukket ut kun to attributter , *Vokaler* og *Frikativer*, for nærmere inspeksjon. Tabell 7.1 legger frem resultater i både antall rammer og prosent for trening og test. FRR (%) og FAR (%) for trening og FRR (%) og FAR (%) for test er tilnærmet like. Systemets høye feilrate kan derfor tvilsomt skyldes større feil i treningsverktøyet. En tilsvarende tabell (7.2) er opprettet for MFCC. I likhet med tabell 7.1 viser også denne at trening har blitt utført på tilfredstillende måte. En antar derfor at treningsfunksjonen ihvertfall ikke er den sterkest bidragsyteren til forskjellen i feilrate mellom de akustiske egenskapsbaserte eksperimentene og MFCC.

MFCC

Antall rammer

		Vokaler		Frikativer	
Trening		33488	5490	16134	2406
		8958	55268	10699	73965
Testing		18140	2914	8098	1347
		4931	28675	5464	39751

Prosent (%)

		Vokaler		Frikativer	
Trening		85.92	14.08	87.02	12.98
		13.95	86.05	12.64	87.36
Testing		86.16	13.84	85.74	14.26
		14.67	85.33	12.09	87.91

Tabell 7.2: Trening- og testresultat (rammer/%) for både Vokaler og Frikativer basert på MFCC

7.3 Feildetektering av Transisjonsrammer

Oppgaven var berammet til å utforske detektorer som arbeidet på rammenivå. Ved inndeling av talesignalet i rammer på 10 ms kan en få problemer ved deteksjon av rammer i transisjonsområdet mellom ulike attributter. En attributt består av minimum 2 til rundt 15 rammer. Generelt er attributtens karakteristikk sterkest i de midtre rammene, mens de ytres verdi kan være svekket som resultat av påvirkning fra naboattributten. Det forventes derfor at de første og siste rammene til en attributt er vanskeligere å detektere.

Feil i transisjonsområdet kan ha stor innvirkning på systemets gjenkjenningsrate. Om for eksempel en 5-rammers attributt har to svake rammer (første og siste) vil feilraten for kun *denne* attributten være 40 %.

Tabell 6.1 i resultatkapittelet inneholder feilprosent som kommer fra de to første og de to siste rammene til en attributt. Feilraten til disse rammene er derimot oppsiktsvekkende lav og en kan dermed ikke anta feildeteksjon i transisjonsområdet som en stor feilkilde. Fra eksperiment 2 ser en at verdien spenner fra 0.39 % for *Dental* til 5.59 % for *Vokaler*. Ved gjenkjenning

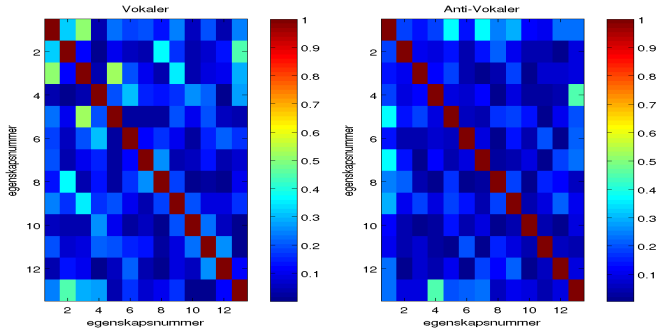
på hele segment (attributt) til fordel for korte tidsrammer vil en unngå feildeteksjon i overgangene. Segmentgjenkjenning er den mest anvendte teknikken og derfor også antatt beste, dvs gir lavest feilrate. I dette arbeidet var ikke optimalisering av feilraten i seg selv prioritert og en valgte derfor rammedeteksjon. Om en senere skal videreføre og forbedre systemets gjenkjenningsrate bør en muligens vurdere en segmentgjenkjenner i stedet.

7.4 Korrelasjon

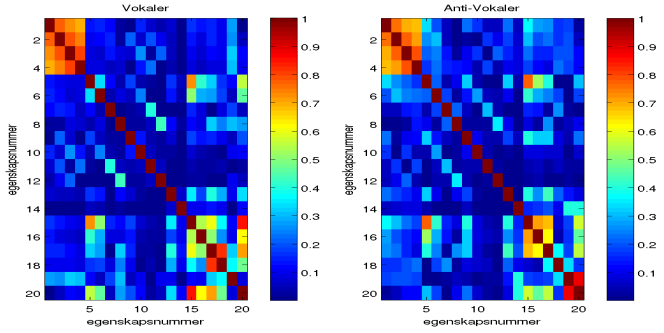
Foruten MFCCs bruk av mel-skalaen for tilnærming av menneskets følsomhet for frekvenser, anvendes DCT for dekorrelering av koeffisientene. Det finnes ulike oppfatninger om hvor vidt denne operasjonen påvirker gjenkjennings-raten. I [25] viser en til svært liten forbedring etter dekorrelering.

Korrelasjonen mellom de akustiske egenskapene og mellom koeffisientene i MFCC har blitt lettere undersøkt. Figur 7.1 viser korrelasjonen mellom de 13 koeffisientene for MFCC og 7.2 for akustiske egenskaper. I begge figurene presenteres korrelasjonen mellom rammer for *Vokaler* og *Anti-vokaler*. Inspeksjon av figurene bekrefter at koeffisientene fra MFCC har noe lavere korrelasjon enn de egendefinerte taleegenskapene.

I rapportens tillegg A.2 har en lagt ved korrelasjonsdiagrammer for *Dental* og *Vokaler* hvor kun de best korrelerte er tatt med. En beregning viser at MFCCs korrelasjon har en maksverdi på rundt 0.4. Tabell A.4 i tillegg A gir en oversikt over settet av alle 20 akustiske egenskaper med korrelasjon under 0.4.



Figur 7.1: Korrelasjon mellom de 13 koeffisientene i MFCC for attributten *Vokal*



Figur 7.2: Korrelasjon mellom de 20 initielle akustiske egenskapene for attributten *Vokal*

Kapittel 8

Konklusjon og Videre Arbeid

Arbeidet som er beskrevet i denne rapporten omhandler analysering og simulering av egendefinerte akustiske taleegenskapers potensial i deteksjon av artikulatoriske taleattributter.

Det er gitt beskrivelse av bakgrunnsteori for generell talegjenkjenning, men med spesielt fokus på oppgavens formål. Oppgaven var berammet til rammedeteksjon av artikulatoriske taleklasser. Gjenkjenningen av klassene var basert på akustiske egenskaper for talesignalet. 20 egenskaper var på forhånd definert og bestemt for oppgaven.

Det ble bygd et gjenkjenningsverktøy basert på HMM (HTK). Flere tester ble utført for ulik bruk av egenskapsvektorer. I tro om å forbedre gjenkjenningsraten ble alle kombinasjoner av egenskaper og attributter analysert ved manuell inspeksjon av histogrammer. En kunne da tilegne hver attributtdetektor et talesignal generert spesielt for den aktuelle attributten. Som referanse til simuleringene vha de egendefinerte egenskapene ble det anvendt tradisjonelle mel-frekvens kepstralkoeffisienter.

Tester gjort ved benyttelse av alle 20 egenskaper styrket den allerede forventede svake feilraten i sammenligning med MFCC. Ved å skreddersy detektorene etter egenskaper oppnår en beskjeden forbedring (se 6.1). Det bekrefter tvilen om at egenskapene en har definert ikke er spesielt egnede.

For ytterlig testing av de egendefinerte egenskapene undersøkte en treningskontra testfeilraten for både akustiske egenskaper og MFCC. Resultatene

viser typiske verdier der feilraten for trening ligger noe lavere enn for test. Differansen er derimot *ikke* betydelige høyere for de akustiske egenskapene og styrker usikkerheten for bruk av egenskapene.

Ettersom rammedeteksjon er valgt for dette arbeidet kan en forvente å oppleve stor feilprosent forårsaket av rammer i overgangene mellom naboattributter. Verdien i parentes i tabell 6.1 viser derimot at denne prosentandelen *ikke* har hatt betydelig innvirkning på resultatet. Størst er den for *Vokaler* som har 5.59 % feildeteksjon av transisjonsrammer. Utelating av disse ville gi en feilrate på 22.52 % kontra 28.11 %. For *Approksimanter* er derimot denne feilraten kun på 1.86 % og ville gitt en feilrate på 33.77 %, som ikke er tilfredstillende.

Til slutt ble korrelasjon for data prosessert vha akustiske egenskaper sjekket mot data prosessert vha MFCC. Figur 7.1 og 7.2 illustrerer henholdsvis korrelasjonen for MFCC og akustiske egenskaper (alle 20). Som en tilleggstest ble et nytt sett med egenskapsvektorer plukket ut, nå med hensyn på korrelasjon. Egenskaper fra eksperiment 2 (se tabell A.2) ble trukket ut etter tabell A.4, som er opprettet etter en egendefinert korrelasjonsterskel. Resultat er fremstilt i tabell A.3 og antyder en beskjeden forbedring. Det kan skyldes *nettopp* mindre korrelasjon, men også andre grunner. Ettersom utvelgelsen ble foretatt manuelt er det vanskelig å konkludere fra eller til.

Analysene som er blitt foretatt her gir grunn til tvil om potensialet til de egendefinerte akustiske egenskapene. En kan dermed konkludere med at egenskapene bør redefineres om en skal oppnå bedre gjenkjenningsrate.

For eventuell videreføring av arbeidet bør redefinering av egenskapene være første prioritet. En kan enten gjøre nærmere undersøkelse av uttrekningsfunksjonene til de eksisterende egenskapene eller innføre nye egenskaper. Spesielt kan en trekke frem at *grad av stemthet* har vist lite bruksverdi for *alle* typer attributter. Funksjonen en her har adoptert har vist gode resultater i [16] for detektering av stemte lyder. Den viser derimot lite egnethet i detektering av attributter, som ofte kan være både stemte og ustemte. I tillegg kunne en utviklet en mer nøyaktig funksjon for å trekket ut de best egnede egenskapene.

Om en videre ønsker å fokusere på forbedring av feilraten kunne en omgjøring til segmentbasert gjenkjenner være interessant. Til tross for at feil

i transisjonsrammene viste å gi mindre utslag kan en segmentgjenkjenner i tillegg overkommer dårlige senterrammer.

Bibliografi

[1] Automatic speech attribute transcription, `http://www.ece.gatech.edu/research/labs/asat/`.

[2] M. Brookes, VOICEBOX: Speech Processing Toolbox for MATLAB, 2000.

[3] L. Vox, History of speech recognition, `http://www.lumenvox.com/resources/tips/historyOfSpeechRecognition.aspx`.

[4] J. Clark and C. Yallop, *An Introduction to Phonetics and Phonology* (Blackwell, 1995).

[5] B. Goelzer, C. Hansen, and G. Sehrndt, *Occupational Exposure to Noise: Evaluation, Prevention and Control* (WHO; Federal Institute for Occupational Safety and Health, 2001).

[6] K. Johnson, *Acoustic and Auditory Phonetics* (Blackwell Pub, 2003).

[7] P. Ladefoged, *A course in phonetics* (Harcourt Brace Jovanovich New York, 1975).

[8] A. A. X. Huang and J.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development* (Prentice Hall PTR, New Jersey, US, 2001).

[9] J. Li, Y. Tsao, and C. Lee, Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on **1** (2005).

[10] A. Juneja and C. Espy-Wilson, Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on **2** (2002).

[11] S. Chang, S. Greenberg, and M. Wester, Proc. Eurospeech **1**, 7 (2001).

[12] D. Donoho, Manuscript (2000).

[13] J. Holmes, W. Holmes, and P. Garner, Proc. Eurospeech97 , 2083.

[14] M. Brookes, Voicebox: Speech processing toolbox for matlab, `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`.

[15] D. Kocharov, A. Zolnay, R. Schluter, and H. Ney, Proc. European Conf. on Speech Communication and Technology , 1101 (2005).

[16] X. Sun, Acoustics, Speech, and Signal Processing, 2002. Proceedings.(ICASSP'02). IEEE International Conference on **1** (2002).

[17] X. SUN, Pitch tracker, `http://www.ling.northwestern.edu/~jbp/sun/xuejing_sun.html`.

[18] S. Chaemmaghami, M. Deriche, and B. Boashash, Proc. TENCON **97** (1997).

[19] Y. Lau and C. Chan, Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on **33**, 320 (1985).

[20] J. Proakis and D. Manolakis, *Digital signal processing: principles, algorithms, and applications* (Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1996).

[21] D. Reynolds and R. Rose, Speech and Audio Processing, IEEE Transactions on **3**, 72 (1995).

[22] G. McLachlan and K. Basford, Statistics (1988).

[23] J. Odell, *The Use of Context in Large Vocabulary Speech Recognition.* (University of Cambridge, 1995).

[24] S. Young *et al.*, Entropic Ltd., Cambridge, England (1999).

[25] T. Eisele, R. Haeb-Umbach, and D. Langmann, Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on **1** (1996).

Tillegg A

Egenskapsanalyse

A.1 Utvelgelse av Akustiske Taleegenskaper

Egenskaper	*Approks.*	*Frikativer*	*Nasaler*	*Stopp*	*Vokaler*	*Stillhet*	*Alveolar*	*Lab-dent*	*Labial*	*Velar*	*Dental*	*Bak*	*Senter*	*Front*
Frekvens F1	X							X			X	X	X	X
Frekvens F2											X			
Frekvens F3			X			X			X				X	X
Frekvens F4	X	X					X	X	X		X	X	X	
Amplitude F1														
Amplitude F2						X								
Amplitude F3						X		X						
Amplitude F4								X				X	X	
Båndbredde F1		X				X					X	X	X	X
Båndbredde F2											X	X		
Båndbredde F3			X	X			X		X	X		X		
Båndbredde F4											X			
Grunnfrekvens														
Stemthetsgrad														
Spektralflathet				X	X	X			X			X	X	X
Midlet svingning i spekter	X			X		X			X			X	X	
Nullkryssingsrate	X	X	X	X	X	X	X			X	X	X	X	X
Nullkryssingsrate (høy-pass)	X		X		X				X		X	X	X	X
Energi (-F3)	X					X		X				X		X
Energi (F3-Nyq)				X	X	X								X

Tabell A.1: Utvelgelsestabell for egenskaper: X er sikre kombinasjoner

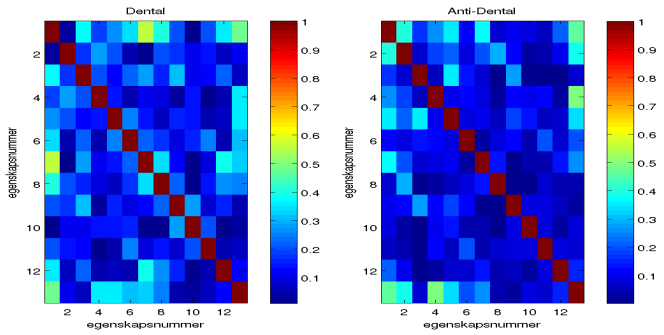
Egenskaper	*Approks.*	*Frikativer*	*Nasaler*	*Stopp*	*Vokaler*	*Stillhet*	*Alveolar*	*Lab-dent*	*Labial*	*Velar*	*Dental*	*Bak*	*Senter*	*Front*
Frekvens F1	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Frekvens F2	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Frekvens F3	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Frekvens F4	X	X	X	X		X		X	X	X	X	X	X	
Amplitude F1														
Amplitude F2				X		X								
Amplitude F3					X	X		X						
Amplitude F4								X			X	X	X	
Båndbredde F1		X			X	X					X	X	X	X
Båndbredde F2											X	X		
Båndbredde F3			X	X			X	X	X	X	X	X	X	
Båndbredde F4										X	X			
Grunnfrekvens														
Stemthetsgrad														
Spektralflathet	X		X	X	X	X			X	X	X	X	X	X
Midlet svingning i spekter	X		X	X		X	X		X	X	X	X	X	
Nullkryssingsrate	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Nullkryssingsrate (høy-pass)	X	X	X		X				X		X	X	X	X
Energi (-F3)	X		X	X	X	X	X	X			X	X		X
Energi (F3-Nyq)	X	X	X	X	X	X		X	X		X	X	X	X

Tabell A.2: Tabell for utvelgelse av egenskaper. I likhet med tabell A.1 velges de sikre kombinasjonene. I tillegg brukes kombinasjoner som ligger i tvilsområdet.

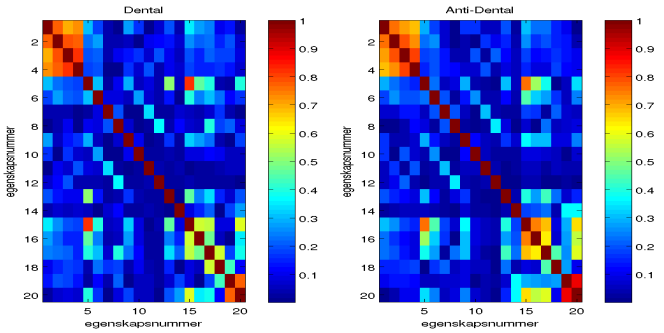
A.2 Korrelasjon i Egenskapsvektorer

Nr.	*Attributt*	*Feilrate*
1	Frikativer	32.22
2	Nasaler	17.22
3	Stopp	27.73
4	Vokaler	34.07
5	Approksimanter	35.26
6	Stillhet	12.39
7	Alveolar	38.34
8	Dental	34.16
9	Lab-dent	25.12
10	Labial	31.40
11	Velar	44.48
12	Bak	38.76
13	Senter	41.16
14	Front	34.15
μ	Gj.snitt	31.89

Tabell A.3: Egenskapsvektor trukket ut mhp korrelasjon. Feilrate i prosent er gitt for hver attributtdetektor.



Figur A.1: Korrelasjon mellom de 13 koeffisientene i MFCC for attributten *Dental*



Figur A.2: Korrelasjon mellom de 20 initielle akustiske egenskapene for attributten *Dental*

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*
1					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
2					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
3					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
4					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
5	X	X	X	X		X	X	X	X	X	X	X	X	X			X	X	X	X
6	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X	X	X
7	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X
8	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X
9	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X
10	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X
11	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X
12	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X
13	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X
14	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X
15	X	X	X	X			X	X	X	X	X	X	X	X				X	X	
16	X	X	X	X		X	X	X	X	X	X	X	X	X				X	X	
17	X	X	X	X	X	X	X	X	X	X	X	X		X					X	
18	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			X	X
19	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
20	X	X	X	X	X	X	X	X	X	X	X	X	X	X				X		

Tabell A.4: Korrelasjonsmatrise for alle 20 egenskaper. De kombinasjoner merket med X har en korrelasjon på mindre enn 0.4, alle andre har høyere korrelasjon

Tillegg B

Fordelingsdiagram for Egenskaper

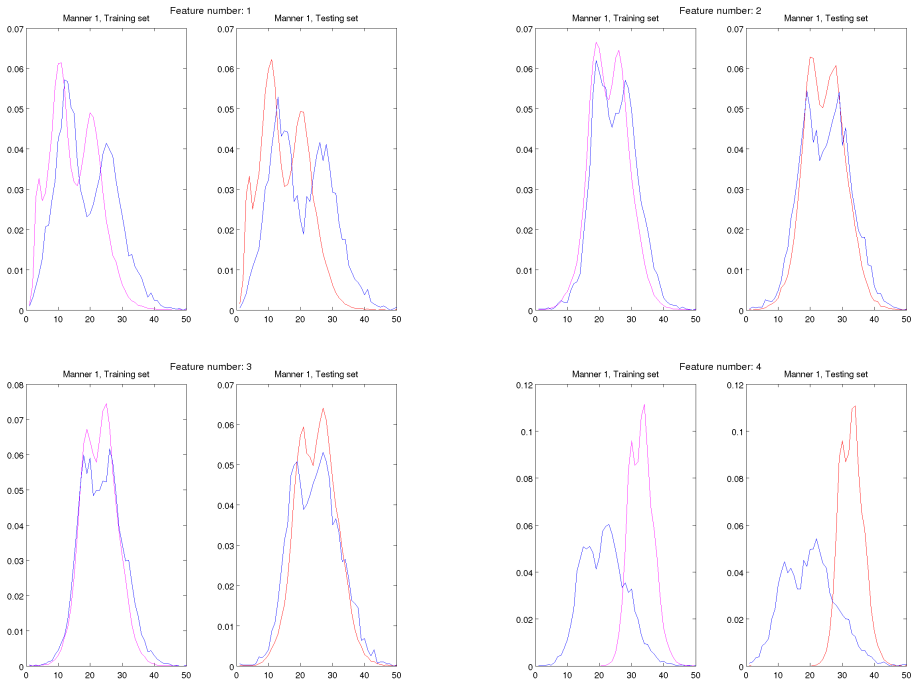
Dette tilleggskapittelet vil presentere noen utvalgte kombinasjoner av taleegenskapene og attributter. De 20 første histogrammene tar for seg alle mulige kombinasjoner med *Approksimanter*, deretter energiegenskapen (0-F3) for alle 14 attributter. Alle diagrammene er merket med attributt- og egenskapsnummer som er referert i tabell B.2 og B.1.

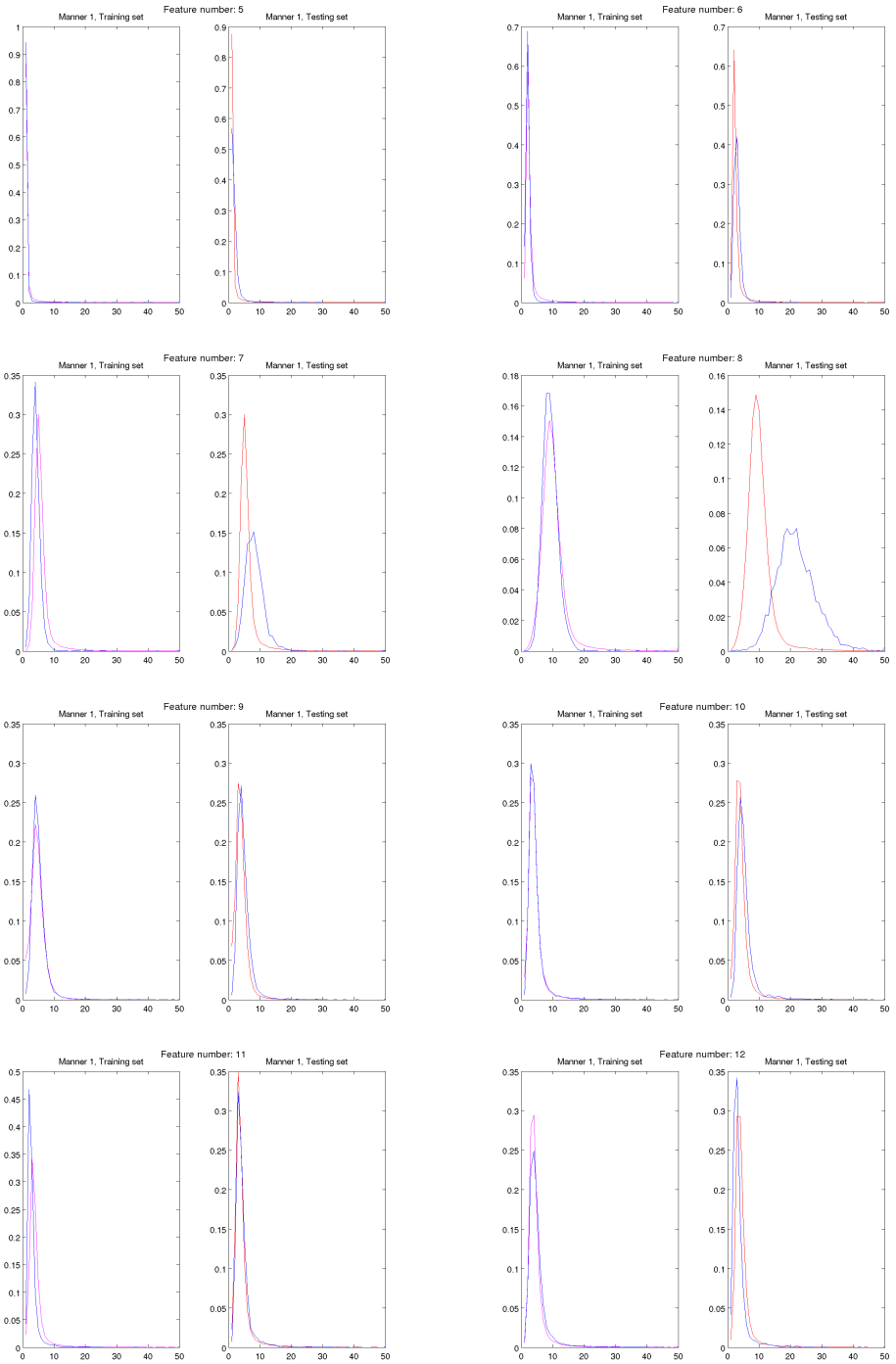
Nr.	*Egenskapsbeskrivelse*
1-4	Frekvens for formant F1,F2,F3 og F4
5-8	Amplitudeverdi for formant F1,F2,F3 og F4
9-12	Båndbredde for formant F1,F2,F3 og F4
13	Grunnfrekvens (F0)
14	Grad av stemthet
15	Spektralflathet
16	Midlet svingning i frekvensspekteret
17	Rate for hvor ofte signalet har negativ verdi
18	Rate for hvor ofte høy-passdelen av signalet har negativ verdi
19	Spektralenergi fra frekvens 0 til tredje formant (F3)
20	Spektralenergi fra F3 til Nyquist-frekvensen (samplingfrekvens/2)

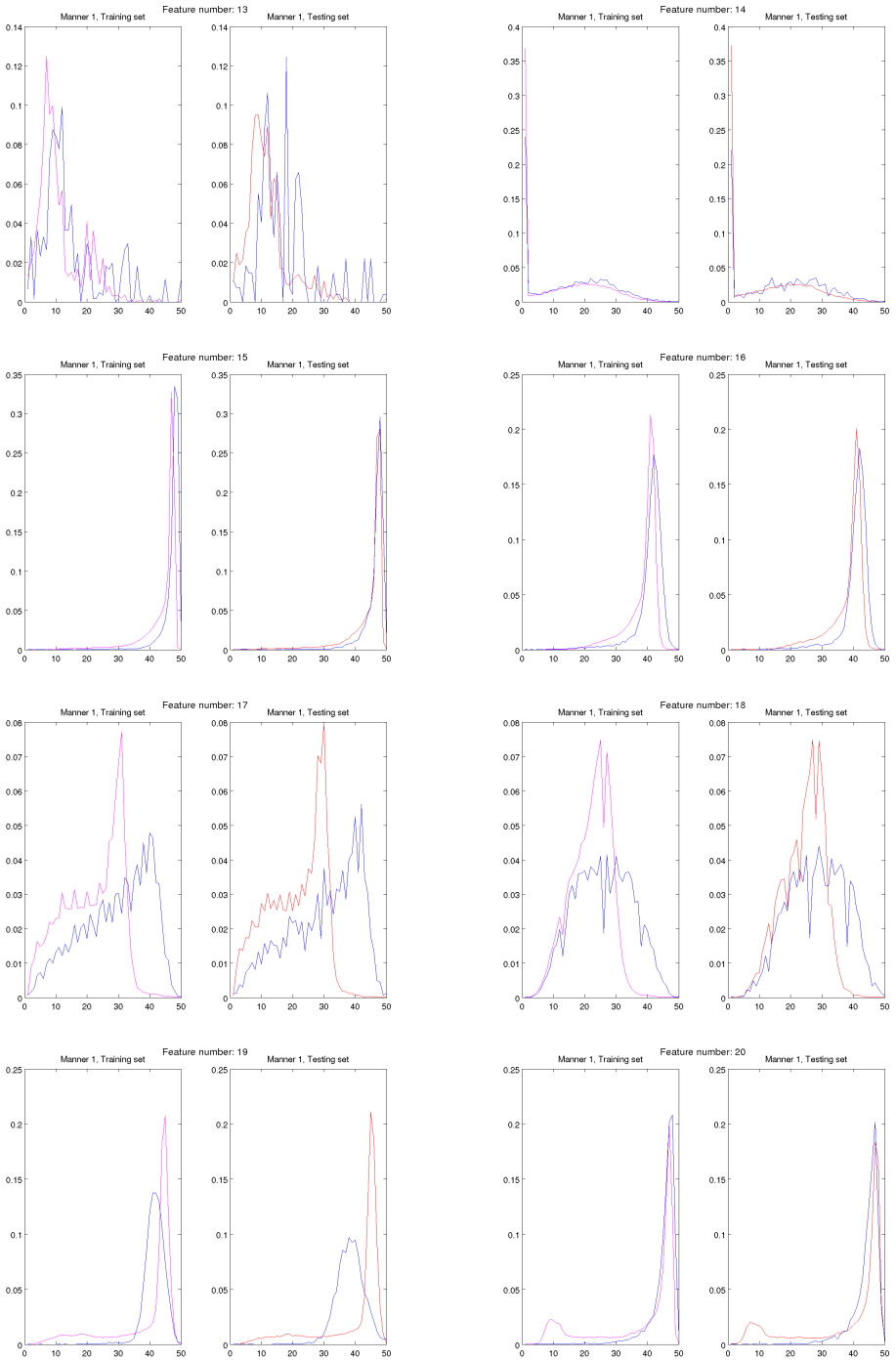
Tabell B.1: Nummerering av egenskaper

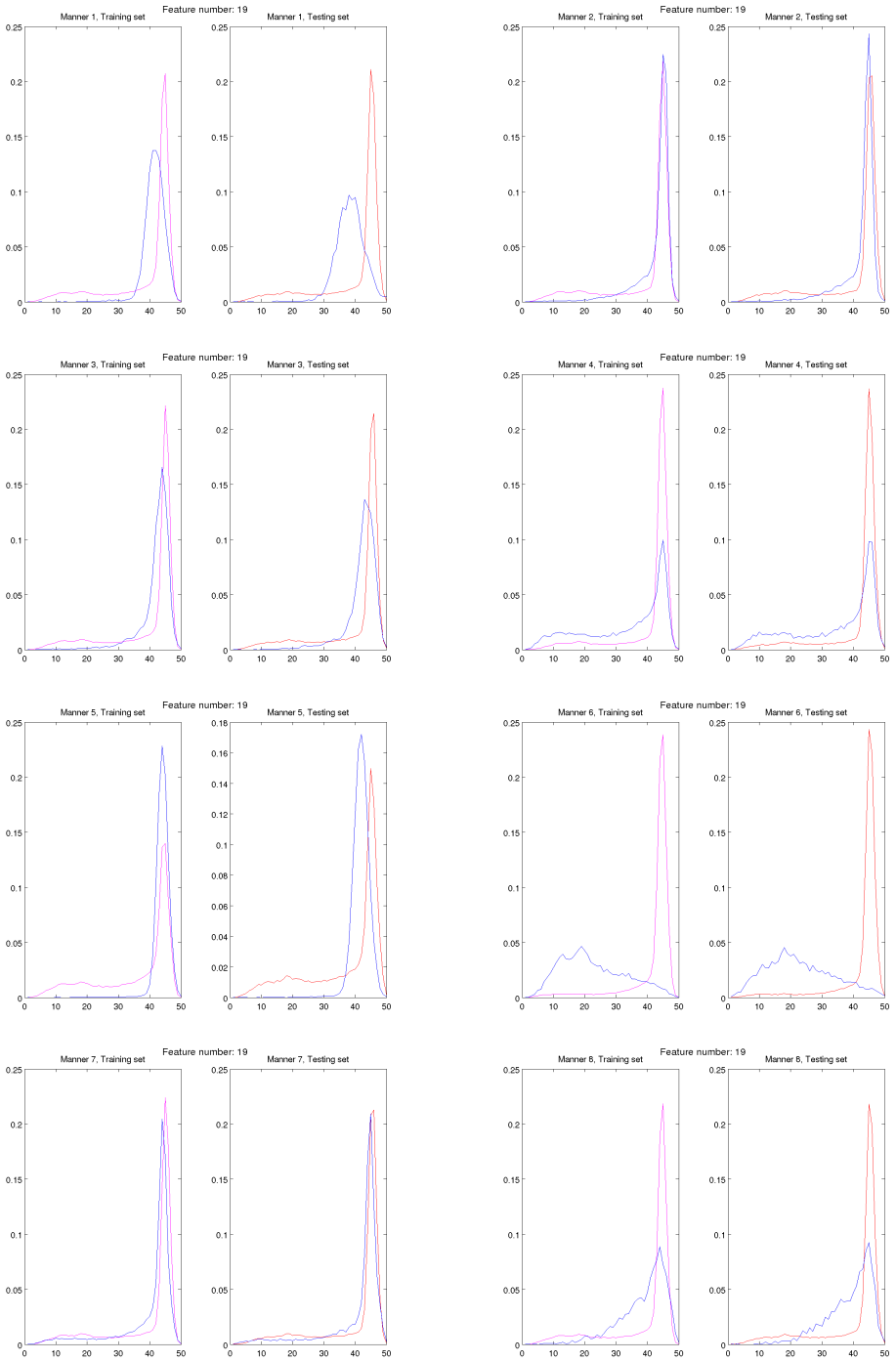
Nr.	*Attributt*
1	Frikativer
2	Nasaler
3	Stopp
4	Vokaler
5	Approksimanter
6	Stillhet
7	Alveolar
8	Dental
9	Lab-dent
10	Labial
11	Velar
12	Bak
13	Senter
14	Front

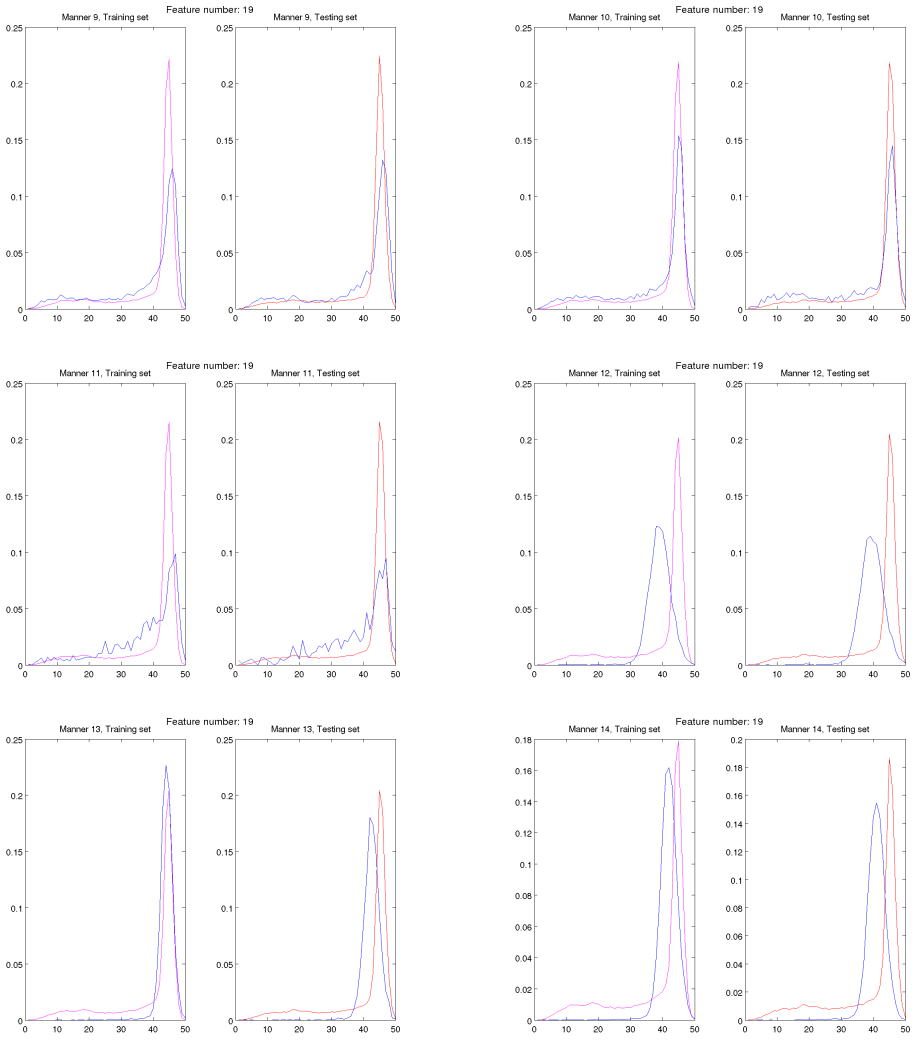
Tabell B.2: Feilrate i prosent for hver attributtdetektor fra ulike simuleringer




Feature number: 5
Manner 1, Training set
Manner 1, Testing set
Feature number: 6
Manner 1, Training set
Manner 1, Testing set
Feature number: 7
Manner 1, Training set
Manner 1, Testing set
Feature number: 8
Manner 1, Training set
Manner 1, Testing set
Feature number: 9
Manner 1, Training set
Manner 1, Testing set
Feature number: 10
Manner 1, Training set
Manner 1, Testing set
Feature number: 11
Manner 1, Training set
Manner 1, Testing set
Feature number: 12
Manner 1, Training set
Manner 1, Testing set


Feature number: 13
Manner 1, Training set
Manner 1, Testing set
Feature number: 14
Manner 1, Training set
Manner 1, Testing set
Feature number: 15
Manner 1, Training set
Manner 1, Testing set
Feature number: 16
Manner 1, Training set
Manner 1, Testing set
Feature number: 17
Manner 1, Training set
Manner 1, Testing set
Feature number: 18
Manner 1, Training set
Manner 1, Testing set
Feature number: 19
Manner 1, Training set
Manner 1, Testing set
Feature number: 20
Manner 1, Training set
Manner 1, Testing set


Feature number: 19
Manner 1, Training set
Manner 1, Testing set
Feature number: 19
Manner 2, Training set
Manner 2, Testing set
Feature number: 19
Manner 3, Training set
Manner 3, Testing set
Feature number: 19
Manner 4, Training set
Manner 4, Testing set
Feature number: 19
Manner 5, Training set
Manner 5, Testing set
Feature number: 19
Manner 6, Training set
Manner 6, Testing set
Feature number: 19
Manner 7, Training set
Manner 7, Testing set
Feature number: 19
Manner 8, Training set
Manner 8, Testing set


Feature number: 19
Manner 9, Training set
Manner 9, Testing set
Feature number: 19
Manner 10, Training set
Manner 10, Testing set
Feature number: 19
Manner 11, Training set
Manner 11, Testing set
Feature number: 19
Manner 12, Training set
Manner 12, Testing set
Feature number: 19
Manner 13, Training set
Manner 13, Testing set
Feature number: 19
Manner 14, Training set
Manner 14, Testing set

Tillegg C

Kildekode

Simuleringsverktøyet er utviklet vha av perl-skript, HTK-skript og funksjoner i Matlab. Alle HTK-skript og hovedfunksjonene i Matlab blir eksekvert fra Perlskriptene trainScript (trening) og recScript (testing/gjenkjenning). Figur C.1 viser hvordan de ulike skriptene blir kjørt. Programmet startes i quickstartup_trainScript. Her må en initiere ulike filbaner og parametere for simuleringen. Programmet kan bruke ulike antall blandingskomponter. Det starter med 1, så 2 og deretter en økning med 2. Programmet vil kjøre igjennom hele gjenkjenningssystemet for et antall komponenter før antallet økes. quickstartup_trainScript styrer dette med en løkke som øker antall komponenter for hver runde.

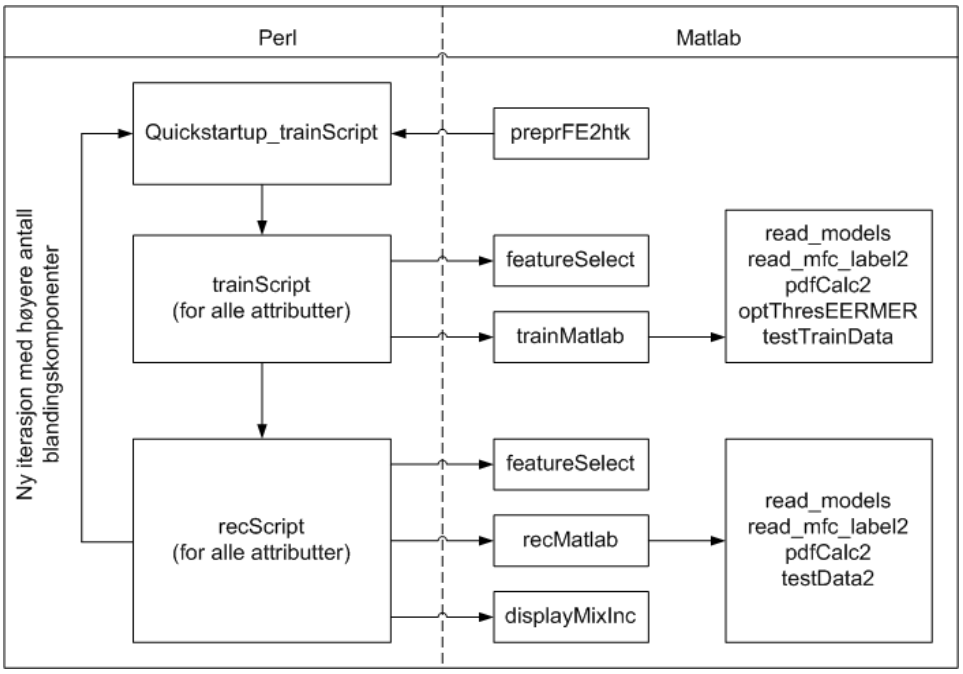
trainScript styrer hele treningsprosessen. Først velges egenskapsvektorene ut, så beregnes HMM-parametere vha HTK og deretter startes trainMatlab. Funksjonen trainMatlab eksekverer ulike funksjoner for å beregne terskelverdi og gjenkjenningsresultat fra trening. Når trening for attributter er ferdig startes gjenkjenningsdelen (recScript).

recScript velger først ut egenskapsvektorer. Deretter kjøres recMatlab for å beregne gjenkjenningsraten for testdataene. Til slutt lages tekstfiler for visning av resultater.

Hvis en skal øke antall blandingskomponenter vil quickstartup_trainScript kjøre igjennom tilsvarende runde med det nye antallet.

Dette kapittelet er delt i to deler, kildekode fra matlab og kildekode fra perl.

Hver fil er kommentert for de viktigste funksjoner.



Figur C.1: Funksjonsdiagram for kildekoden.

C.1 Matlab

C.1.1 preprFE2htk.m

```
1  % Dette programmet beregner egenskapsverdien for hver ramme.
2
3  %  Egenskapene blir beregnet i rekkefoelgen under:
4  %  Frekvens F1
5  %  Frekvens F2
6  %  Frekvens F3
7  %  Frekvens F4
8  %  Amplitude F1
9  %  Amplitude F2
10 %  Amplitude F3
11 %  Amplitude F4
12 %  Baandbredde F1
13 %  Baandbredde F2
```

```
14 %  Baandbredde F3
15 %  Baandbredde F4
16 %  Grunnfrekvens
17 %  Stemthetsgrad
18 %  Spektralflathet
19 %  Midlet svingning i spekter
20 %  Nullkryssingsrate
21 %  Nullkryssingsrate (hoey-pass)
22 %  Energi ( -F3)
23 %  Energi (F3-Nyq)
24
25 % symbol: >>>>>>>>>>>> indikerer hvor en egenskapsverdi
       kalkuleres
26
27
28
29 Fs = 16000; % samplingfrekvens
30 scale = 100; % number of 10 msec segments in 1 sec
31
32 num_fon = 56;
33
34 Samp_step = Fs*10/1000;
35 Samp_wind = Fs*20/1000;
36 p = 12; N_form = 4;
37 tresh = 10.0;                    % OBS!
38 F0MinMax = [50 550];
39 Time_frame = 40;
40 Time_step = 10;
41 SHR_Threshold = .4;
42 ceiling = 1250;
43 CHECK_VOICING = 1;
44 med_smooth = 13;
45 wham = hamming(Samp_wind);
46
47
48 [n_coef,fg,mg,w] = firpmord( [2800 3500], [1 0], [0.02 0.02],
       16000 );
49 fir_lp = firpm(n_coef,fg,mg,w);
50 [n_coef,fg,mg,w] = firpmord( [2800 3500], [0 1], [0.02 0.02],
       16000 );
51 fir_hp = firpm(n_coef,fg,mg,w);
52
53
54 feafile_in = 'tr1.mfc';
```

```
55 [fid, message] = fopen(feafile_in, 'r', 'b');
56     nframes=fread(fid,1,'int32');  % number of frames
57     sp=fread(fid,1,'int32');  % frame step in micro-sec
58     ssb=fread(fid,1,'short');  % number of bytes
59 
60     ssf=ssb/4;                % vector dimension
61     sk=fread(fid,1,'short');  % feature-kind
62     %disp([nframes sp ssf sk])
63     fea_mat=zeros(ssf,nframes);
64     fea_mat1=fread(fid,nframes*ssf,'float');
65     fea_mat(:)=fea_mat1;
66 fclose(fid);
67 
68 type = 'test';
69 startFilt = 0;
70 
71 if startFilt == 1
72 filt = '_filtered';
73 else
74 filt = '';
75 end
76 
77 fid_mfc = fopen(['/u/studenter/fasselan/HMM_testing/Test/lists
       /462_set/' type 'All_mfc' filt '.lis'],'r','b');
78 
79 if(eq(type(2),'r') == 1)
80 Nfiles = 4620;%305
81 else
82 Nfiles = 1680; % total : 157
83 end
84 
85 fid_wav = fopen(['/u/studenter/fasselan/HMM_testing/Test/lists
       /462_set/' type 'All_wav.lis'],'r','b');
86 fid_phn = fopen(['/u/studenter/fasselan/HMM_testing/Test/lists
       /462_set/' type 'All_phn.lis'],'r','b');
87 fea_all = [];
88 tar_all = [];
89 f_lengths = [];
90 
91 
92 disp(' ******** rett foer for loop gjennom alle filer ********'
       )
93 for kfile = 1:Nfiles
94     wavfile = fgetl(fid_wav);
```

```
95
96      fid = fopen(wavfile,'rb');
97          y = fread(fid,'int16','ieee-be');
98          x = y(513:length(y));
99      fclose(fid);
100
101     Lx = length(x);
102
103 if startFilt == 1
104   b = [1 -0.98];
105   x = filter(b,1,x);
106 end
107
108 % >>>>>>>>>>>>
109 %  Frekvens F1
110 %  Frekvens F2
111 %  Frekvens F3
112 %  Frekvens F4
113 %  Amplitude F1
114 %  Amplitude F2
115 %  Amplitude F3
116 %  Amplitude F4
117 %  Baandbredde F1
118 %  Baandbredde F2
119 %  Baandbredde F3
120 %  Baandbredde F4
121
122     [ar,e,kf] = lpcauto(x,p,[Samp_step Samp_wind]);
123     [NForm,FreqForm,AmpForm,BwForm] = lpcar2fm(ar,tresh);
124
125 % klipper bort unaturlige hoeye verdier for amplitude
126 % egenskapene 5,6,7,8
127 maxValue = [200 30 10 5];
128 for maxI = 1:1:4
129 indexMax = AmpForm(:,maxI)>maxValue(maxI);
130 AmpForm(indexMax,maxI) = maxValue(maxI);
131 end
132
133 % >>>>>>>>>>>>
134 %  Grunnfrekvens
135 %  Stemthetsgrad
136     [f0_time,f0_value,SHR,f0_candidates] = shrp(x,Fs,F0MinMax,
          Time_frame,Time_step,SHR_Threshold,ceiling,med_smooth,
          CHECK_VOICING);
```

```
137
138     x_hp = filtfilt(fir_hp,1,x);
139     Nfr = floor((Lx - Samp_wind + Samp_step)/Samp_step);
140     Spec_flat = zeros(Nfr,1);
141     Spec_deriv = Spec_flat; zero_cr = Spec_flat; zero_cr_hp =
            zero_cr; E_0_F3 = zero_cr; E_F3_Nyq = zero_cr;
142
143     for i = 1 : Nfr
144         frame_start = (i-1)*Samp_step + 1;
145         xfr = x(frame_start : frame_start + Samp_wind - 1);
146         xfr = xfr.*wham;
147         Xfr = abs(fft(xfr,512));
148         Xfr = Xfr(1:256);
149         Xfr_norm = sqrt(Xfr(1)^2 + Xfr(256)^2 + 2*sum((Xfr
                (2:255).^2)));
150     Xfrn = Xfr/Xfr_norm;
151
152 % >>>>>>>>>>>>
153 %  Spektralflathet
154         Spec_flat(i) = sum(log(Xfrn)) - log(sum(Xfrn));
155
156 % >>>>>>>>>>>>
157 %  Midlet svingning i spekter
158         Spec_deriv(i) = log(sum(abs(diff(Xfrn))));
159         [sp_cor,dum] = zerocros(xfr);
160
161 % >>>>>>>>>>>>
162 %  Nullkryssingsrate
163     zero_cr(i) = length(sp_cor);
164         xfr_hp = x_hp(frame_start : frame_start + Samp_wind -
                1);
165         [sp_cor_hp,dum] = zerocros(xfr_hp);
166
167 % >>>>>>>>>>>>
168 %  Nullkryssingsrate (hoey-pass)
169     zero_cr_hp(i) = length(sp_cor_hp);
170         F3 = FreqForm(i,3); F3_samp = round(F3*512);
171         if((F3 > 1/16) | (F3 < 15/32))
172 % >>>>>>>>>>>>
173 %  Energi ( -F3)
174             E_0_F3(i) = log(sum((Xfr(1:F3_samp).^2)));
175 % >>>>>>>>>>>>
176 %  Energi (F3-Nyq)
177         E_F3_Nyq(i) = log(sum((Xfr(F3_samp+1:256).^2)));
```

```
178         end
179     end
180
181     phnfile =fgetl(fid_phn);
182     fid = fopen(phnfile,'r','b');
183     tar = [];
184     k = 0;
185     sind_old = 0;
186
187     ss1f = []; ss2f = []; clf = [];
188     while 1
189         k = k+1;
190         ss = fscanf(fid,'%d %d');
191         if (length(ss) == 0) break; end
192         ss1f = [ss1f ss(1)]; ss2f = [ss2f ss(2)];
193         cl = fscanf(fid, '%s',1); clf{k} = cl;
194     end
195     Nseg = k - 1;
196     k = 1;
197     while (k < Nseg)
198         k = k+1;
199         a1 = (strcmp(strtrim(clf{k-1}),'kcl') ==1) & (strcmp(
                strtrim(clf{k}),'k') == 1);
200         a2 = (strcmp(strtrim(clf{k-1}),'bcl') ==1) & (strcmp(
                strtrim(clf{k}),'b') == 1);
201         a3 = (strcmp(strtrim(clf{k-1}),'pcl') ==1) & (strcmp(
                strtrim(clf{k}),'p') == 1);
202         a4 = (strcmp(strtrim(clf{k-1}),'dcl') ==1) & (strcmp(
                strtrim(clf{k}),'d') == 1);
203         a5 = (strcmp(strtrim(clf{k-1}),'tcl') ==1) & (strcmp(
                strtrim(clf{k}),'t') == 1);
204         a6 = (strcmp(strtrim(clf{k-1}),'tcl') ==1) & (strcmp(
                strtrim(clf{k}),'ch') == 1);
205         a7 = (strcmp(strtrim(clf{k-1}),'gcl') ==1) & (strcmp(
                strtrim(clf{k}),'g') == 1);
206         a8 = (strcmp(strtrim(clf{k-1}),'dcl') ==1) & (strcmp(
                strtrim(clf{k}),'jh') == 1);
207
208         if ( a1 || a2 || a3 || a4 || a5 || a6 || a7 || a8)
209             ss2f(k-1) = ss2f(k);
210             ss1f = [ss1f(1:k-1) ss1f(k+1:Nseg)]; ss2f = [ss2f
                    (1:k-1) ss2f(k+1:Nseg)];
211             if (k > 2)
212                 clf = [clf(1:k-2) clf(k:Nseg)];
```

```
213             else
214                 clf = clf(2:Nseg);
215             end
216             Nseg = Nseg - 1;
217         end
218     end
219     for k = 1 : Nseg
220             if (strcmp(strtrim(clf{k}),'kcl') ==1) clf{k} = 'k'
                    ; end
221             if (strcmp(strtrim(clf{k}),'bcl') ==1) clf{k} = 'b'
                    ; end
222             if (strcmp(strtrim(clf{k}),'tcl') ==1) clf{k} = 't'
                    ; end
223             if (strcmp(strtrim(clf{k}),'dcl') ==1) clf{k} = 'd'
                    ; end
224             if (strcmp(strtrim(clf{k}),'gcl') ==1) clf{k} = 'g'
                    ; end
225             if (strcmp(strtrim(clf{k}),'pcl') ==1) clf{k} = 'p'
                    ; end
226     end
227     fclose(fid);
228     M = Nfr - 3;
229     SHR_med = medsmooth(SHR,med_smooth);
230
231     For_Freq = zeros(M,N_form); For_Amp = For_Freq; For_Bw =
            For_Freq;
232     [dum,file_form] = size(FreqForm);
233     For_valg = min(N_form,file_form);
234     For_Freq(:,1:For_valg) = FreqForm(2:M+1,1:For_valg);
235     For_Amp(:,1:For_valg)  = AmpForm(2:M+1,1:For_valg);
236     For_Bw(:,1:For_valg) = BwForm(2:M+1,1:For_valg);
237
238     % Sammenslaaing av alle egenskapsverdier for alle rammer
            til en matrise
239     fea = [For_Freq For_Amp For_Bw f0_value(1:M) SHR_med(1:M)
            Spec_flat(2:M+1) Spec_deriv(2:M+1) zero_cr(2:M+1)
            zero_cr_hp(2:M+1) E_0_F3(2:M+1) E_F3_Nyq(2:M+1)];
240
241
242 [ns,ssf] =size(fea);  % define number of frames for file and
        vector dimension
243 ss = ssf*4; % number of bytes per frame
244 fea_all = [fea_all;fea];
245 fea = fea';
```

```
246 fea_corr = fea(:);
247
248 % Skrive data basert paa egnskapsverdiene til fil
249 mfcfile = fgetl(fid_mfc);
250 fid=fopen(mfcfile,'w','b');
251         fwrite(fid,ns,'int32');
252         fwrite(fid,sp,'int32');
253         fwrite(fid,ss,'short');
254         fwrite(fid,sk,'short');
255         fwrite(fid,fea_corr,'float');
256 fclose(fid);
257 tar_all = [tar_all;tar];
258
259 end
260 fclose(fid_wav);
261 fclose(fid_phn);
```

C.1.2 trainMatlab.m

```
1  % Hovedfunksjon for treningsdelen
2  % Flere andre funksjoner blir eksekvert nedenfor i denne
       rekkefoelgen:
3
4  % read_models.m
5  % read_mfc_label2.m
6  % pdfCalc2.m
7  % optThresEERMER.m
8  % testTrainData.m
9
10 disp(sprintf('------- trainMatlab.m -------'))
11
12 % filnavn til parametfil inputVar.lis
13 path_inputVar = '/u/studenter/fasselan/HMM_testing/Test/lists
       /462_set/inputVar.lis';
14
15 % Innlesing av parameter fra inputVar.lis
16 varArray = textread(path_inputVar,'%s');
17 p = str2double(char(varArray(1)));
18 nS = str2double(char(varArray(2)));
19 nM = str2double(char(varArray(3)));
20 vS = str2double(char(varArray(4)));
```

```
21 Nfiles = str2double(char(varArray(5)));
22 decRule = str2double(char(varArray(7)));
23 mfcTrigger = str2double(char(varArray(9)));
24 targetChoice = str2double(char(varArray(12)))
25 path = char(varArray(17))
26 TIMITpath = char(varArray(18))
27 listpath = char(varArray(19))
28
29 % Innlesing av parameter fra feaSelVar.lis
30 varArray2 = textread('/u/studenter/fasselan/HMM_testing/Test/
       lists/462_set/feaSelVar.lis','%s');
31 vS = str2double(char(varArray2(5)))
32 filelistSel = char(varArray2(7))
33
34 % henter inn liste over attributter
35 fid_att = fopen([path listpath 'attributes.lis'], 'r', 'b');
36
37 teller = 1;
38 t = fscanf(fid_att,'%s\n',1);
39 attributeArray = cellstr(t);
40 while 1
41 t = fscanf(fid_att,'%s\n',1);
42 if(strcmp(t,'END')==1)break;end
43 attributeArray = [attributeArray cellstr(t)];
44 end
45 fclose(fid_att);
46 antAtt = length(attributeArray); % antall attributter
47
48 % Henter inn terskelverdien for aa kunne legge til nye verdier
49 filepath = [path 'tempFiles/threshold.mat'];
50 load(filepath);
51     if(thresTeller≥antAtt)
52         thresTeller=0;
53         thresholds = [];
54     end
55 currAttr = attributeArray(thresTeller+1); % current attribute
56 currAttr = char(currAttr);
57 attStr = ['current attribute being processed is ' currAttr];
58 disp(' ')
59 disp(sprintf('%-50s\n',attStr))
60
61 % lese inn modellparametere fra HMM
62 if nM ≥ 2
```

```
63 mmfFile = [path 'hmms/hmm' num2str(nM) num2str(nM) '/' '
       attrMMF_' currAttr '.mmf'];
64 else
65 mmfFile = [path 'hmms/' 'attrMMF_' currAttr '.mmf'];
66 end
67 modList = [path listpath 'att.lis'];
68
69 cd Matlab;
70 read_models(mmfFile,modList,nM,vS);
71 cd ..;
72
73 % 'Master Label File' for den aktuelle attributten
74 labelList = [path listpath 'attribute.mlf'];
75 type = 'train';
76
77 % Liste over mulige fillister
78 mfcList1 = [path listpath 'trainAll_mfc.lis'];
79 mfcList2 = [path listpath 'train462_mfc2.lis'];
80 mfcList3 = [path listpath 'train462_htk_AND_FEmfc.lis'];
81 mfcList4 = [path listpath 'train462_mfc_NEW.lis'];
82 mfcList5 = [path listpath 'train462_mfc_NEW_filtered.lis'];
83 mfcList6 = [path listpath 'train462_mfc_NEW_reducedFEA.lis'];
84
85 if mfcTrigger == 1
86 mfcList = mfcList1;
87 mfcT = 'Feature_Vectors';
88 elseif mfcTrigger == 2
89 mfcList = mfcList2;
90 mfcT = 'MFCCs';
91 elseif mfcTrigger == 3
92 mfcList = mfcList3;
93 mfcT = 'Feature_Vectors_AND_MFCCs';
94 elseif mfcTrigger == 4
95 mfcList = mfcList4;
96 mfcT = 'Feature_Vectors_(MFCC_E)';
97 elseif mfcTrigger == 6
98 mfcList = mfcList6;
99 mfcT = 'Feature vectors_E_redFEA';
100 else
101 mfcList = mfcList5;
102 mfcT = 'Feature_Vectors_(MFCC_E)_FILT';
103 end
104 mfcList = filelistSel;
105
```

```
106 % lese filer og tilegne korrekte etiketter
107 cd Matlab;
108 read_mfc_label2(mfcList,labelList,currAttr,Nfiles,mfcTrigger,
        type,mfcT);
109 cd ..;
110
111 % Kalkulerer logaritmisk pdf-verdien for hver ramme
112 model = [path 'tempFiles/' 'model_' currAttr '_M' num2str(nM) '
        .mat'];
113 cd Matlab;
114 r= pdfCalc2(type,nS,nM,model,vS,currAttr,mfcT);
115 cd ..;
116 naivTerskel = r;
117
118 % Kalkulerer terskelverdi
119 cd Matlab;
120 r = optThresEERMER(nM,decRule,currAttr,targetChoice);
121 cd ..;
122 thresholds = [thresholds; r(1)];
123 signTh = [signTh; r(2)];
124 thresTeller = thresTeller + 1;
125 disp(' ')
126 disp('thresholdsrarray:');
127 disp(thresholds);
128
129 %  finne feilrate for trening
130 testdatapath = [path 'tempFiles/testTrainData_M' num2str(nM) '
        _feature' num2str(mfcTrigger) '_' p '_' decRule '.mat'];
131 trainRecAll = [path 'tempFiles/trainResult'];
132 cd Matlab;
133 trainRecResults = testTrainData(testdatapath, r(1), nM,currAttr
        ,targetChoice,trainRecAll,r(2));
134 cd ..;
135
136 % Lagrer data
137 save(filepath,'thresholds','thresTeller','signTh');
138 disp(' ')
139 disp('End of training')
```

C.1.3 recMatlab.m

```
% Hovedfunksjon for testingsdelen
% Flere andre funksjoner blir eksekvert nedenfor i denne
    rekkefoelgen:

% read_mfc_label2.m
% pdfCalc2.m
% testData2.m

disp(' ')
disp(sprintf('---- recMatlab.m -----'));

% filnavn til parametfil inputVar.lis
path_inputVar = '/u/studenter/fasselan/HMM_testing/Test/lists
    /462_set/inputVar.lis';

% Innlesing av parameter fra inputVar.lis
varArray = textread(path_inputVar,'%s');
p = str2double(char(varArray(1)));
nS = str2double(char(varArray(2)));
nM = str2double(char(varArray(3)));
vS = str2double(char(varArray(4)));
Nfiles = str2double(char(varArray(8)));
%Nfiles  = round(Nfiles/3);
decRule = str2double(char(varArray(7)));
mfcTrigger = str2double(char(varArray(9)));
targetChoice = str2double(char(varArray(12)));
path = char(varArray(17))
TIMITpath = char(varArray(18))
listpath = char(varArray(19))

% Innlesing av parameter fra feaSelVar.lis
varArray2 = textread('/u/studenter/fasselan/HMM_testing/Test/
    lists/462_set/feaSelVar.lis','%s');
vS = str2double(char(varArray2(5)))
filelistSel = char(varArray2(7))

% henter inn liste over attributter
fid_att = fopen([path listpath 'attributes.lis'], 'r', 'b');

if p == 1
p = 'phonemes';
else
p = 'attributes';
end
```

```
42 if decRule == 1
43 decRule = 'EER';
44 else
45 decRule = 'MER';
46 end
47
48 teller = 1;
49 t = fscanf(fid_att,'%s\n',1);
50 attributeArray = cellstr(t);
51 while 1
52 t = fscanf(fid_att,'%s\n',1);
53 if(strcmp(t,'END')==1)break;end
54 attributeArray = [attributeArray cellstr(t)];
55 end
56 fclose(fid_att);
57 antAtt = length(attributeArray); % antall attributter
58
59 % Henter inn terskelverdi
60 filepath = [path 'tempFiles/' 'threshold.mat'];
61 load(filepath);
62 recFile = [path 'tempFiles/recData.mat'];
63 load(recFile);
64 recTeller  = recTeller +1;
65 currAttr = attributeArray(recTeller); % current attribute
66 currAttr = char(currAttr);
67 attStr = ['current attribute being processed is ' currAttr];
68 disp(' ')
69 disp(sprintf('%-50s\n',char(attStr)))
70
71 % 'Master Label File' for den aktuelle attributten
72 labelList = [path listpath 'RecAttribute.mlf'];
73 type = 'test';
74
75 % Liste over mulige fillister
76 mfcList1 = [path listpath 'testAll_mfc.lis'];
77 mfcList2 = [path listpath 'test462_mfc2.lis'];
78 mfcList3 = [path listpath 'test462_htk_AND_FEmfc.lis'];
79 mfcList4 = [path listpath 'test462_mfc_NEW.lis'];
80 mfcList5 = [path listpath 'test462_mfc_NEW_filtered.lis'];
81 mfcList6 = [path listpath 'test462_mfc_NEW_reducedFEA.lis'];
82
83 if mfcTrigger == 1
84 mfcList = mfcList1;
85 mfcT = 'Feature_Vectors';
```

```
86  elseif mfcTrigger == 2
87  mfcList = mfcList2;
88  mfcT = 'MFCCs';
89  elseif mfcTrigger == 3
90  mfcList = mfcList3;
91  mfcT = 'Feature_Vectors_AND_MFCCs';
92  elseif mfcTrigger == 4
93  mfcList = mfcList4;
94  mfcT = 'Feature_Vectors_(MFCC_E)';
95  elseif mfcTrigger == 6
96  mfcList = mfcList6;
97  mfcT = 'Feature vectors_E_redFEA';
98  else
99  mfcList = mfcList5;
100 mfcT = 'Feature_Vectors_(MFCC_E)_FILT';
101 end
102 mfcList = filelistSel;
103
104 % lese filer og tilegne korrekte etiketter
105 cd Matlab;
106 read_mfc_label2(mfcList,labelList,currAttr,Nfiles,mfcTrigger,
        type,mfcT);
107 cd ..;
108
109 % Kalkulerer logaritmisk pdf-verdien for hver ramme
110 model = [path 'tempFiles/' 'model_' currAttr '_M' num2str(nM) '
        .mat']
111 cd Matlab;
112 r = pdfCalc2(type,nS,nM,model,vS,currAttr,mfcT);
113 cd ..;
114
115 % antall "aktive" blandingsfordelinger for henholdsvis
116 % attributt og anti-attributt
117 attrMix = r(2);
118 antiAttrMix = r(3);
119
120 % Kalkulere gjenkjenningsraten
121 testdatapath = [path 'tempFiles/testData_M' num2str(nM) '
        _feature' num2str(mfcTrigger) '_' p '_' decRule '.mat'];
122 cd Matlab;
123 recResults = testData2(testdatapath, filepath, nM,recTeller,
        currAttr,targetChoice);
124 cd ..;
125 finalRes(:,:,recTeller) = recResults(:,1:2);
```

```
126 finalResNUM(:,:,recTeller) = recResults(:,3:4);
127 edgeError = [edgeError recResults(1,5)];
128
129 if recTeller == antAtt
130 m1 = mean(finalRes(1,1,:));
131 m2 = mean(finalRes(1,2,:));
132 m3 = mean(finalRes(2,1,:));
133 m4 = mean(finalRes(2,2,:));
134 m11 = sum(finalResNUM(1,1,:));
135 m21 = sum(finalResNUM(1,2,:));
136 m31 = sum(finalResNUM(2,1,:));
137 m41 = sum(finalResNUM(2,2,:));
138 disp(sprintf('\n\n\n'));
139 disp(sprintf('|-------------|          |-------------|'));
140 disp(sprintf('|  Result %%   |          |   Result    |'));
141 disp(sprintf('|-------------|          |-------------|\n\n'));
142 disp(sprintf('|-------------|          |-------------|'));
143 disp(sprintf('|%-6.2f|%-6.2f|          |%-6.0f|%-6.0f|',m1,m2,
        m11,m21));
144 disp(sprintf('|------|------|          |------|------|   '));
145 disp(sprintf('|%-6.2f|%-6.2f|          |%-6.0f|%-6.0f|',m3,m4,
        m31,m41));
146 disp(sprintf('|-------------|          |-------------|'));
147 end
148
149 % Lagrer data
150 save(recFile,'recResults','finalRes','finalResNUM','recTeller',
        'attrMix','antiAttrMix','edgeError');
151 disp(' ')
152 disp('End of testing')
```

C.1.4 read_mfc_label2.m

```
1 function r = read_mfc_label2(mfcList,labelList,attribute,Nfiles
      ,mfcTrigger,type,mfcT)
2 % Leser inn datafiler og etikettfiler og smelter dem sammen i
      matriser
3 % Inndata:
4 % mfcList: filliste for datafiler
5 % labelList: filliste for etikettfiler
6 % attribute: attributten som prosesseres
```

```
7  % Nfiles : antall filer
8  % mfcTrigger: kildetype
9  % type: train eller test
10 % mfcT: streng som beskriver anvendte egenskapsvektoren
11
12 disp(' ')
13 disp(sprintf('----------- read_mfc_label2.m ------------'));
14 disp(sprintf(mfcList));
15 disp(sprintf(labelList));
16 disp(sprintf(attribute));
17
18 fid_mlf = fopen(labelList,'r','b');
19 [fid_mfc message] = fopen(mfcList,'r','b');
20 mlfstring = fscanf(fid_mlf, '%s',1);
21 tar_all = [];
22 centerTar_all = [];
23 fea_all = [];
24 counter =0;
25
26     disp(sprintf('\n\n
           |-------------------------------------------------------------------------
           '))
27     disp(sprintf('|  - %-23s

           |',mfcT))
28     disp(sprintf('
           |-------------------------------------------------------------------------
           '))
29     disp(sprintf('|                      filename
                                | frames | frame step | vector
           dim | feature kind |'))
30
31 % Etikettfiler for hver datafil hentes og legges inn i en stor
       vektor
32 % skriver ut filene som prosesseres
33 for km = 1:Nfiles
34    tar = [];
35    centerTar = [];
36     M = 0;
37     file = fscanf(fid_mlf, '%s',1);
38     ss1f = []; ss2f = []; clf = [];
39     sind_old =0;
40     k=0;
41     while 1
```

```
42          k = k+1;
43          ss = fscanf(fid_mlf,'%d %d');
44
45          if (length(ss) == 0)
46          dummy = fscanf(fid_mlf, '%s',1);
47              break;
48          end
49          ss1f = [ss1f ss(1)];
50          ss2f = [ss2f ss(2)];
51          cl = fscanf(fid_mlf, '%s',1);
52          clf{k} = cl;
53      end
54      for k = 1:length(clf)
55         if (strcmp(attribute,clf(k)) == 1)
56             tars = 1;
57         else
58             tars = 0;
59         end
60         sind = floor(scale*ss2f(k)/Fs);  % last frame
61         seglen = length(sind_old+1:sind);
62         sind_old = sind;
63         tar = [tar; tars*ones(seglen,1)];
64
65      if tars == 1
66      centerTemp = tars*ones(seglen,1);
67      antCent = ceil(length(centerTemp)*0.2);
68      tempA = tars*zeros(seglen,1);
69      lengthCent = length(centerTemp);
70      findIfOdd = lengthCent/2 - floor(lengthCent/2);
71      ned = round(antCent/2);
72      opp = antCent - 1;
73
74          if findIfOdd > 0
75          arrayCenter = ceil(lengthCent/2);
76          tempA((arrayCenter- ned + 1):(arrayCenter + opp)) = 1;
77          else
78          arrayCenter = ceil(lengthCent/2) + 1;
79          tempA((arrayCenter- ned):(arrayCenter + opp -1)) = 1;
80          end
81
82      centerTar = [centerTar; tempA];
83      else
84      centerTar = [centerTar; tars*ones(seglen,1)];
85      end
```

```
86      end
87
88
89  % Leser data fra fil
90      mfcfile = fgetl(fid_mfc);   % read filename
91      fid = fopen(mfcfile,'rb','b');
92      nframes=fread(fid,1,'int32');  % number of frames
93      sp=fread(fid,1,'int32');  % frame step in micro-sec
94      ssb=fread(fid,1,'short');  % number of bytes
95      ssf=ssb/4;                % vector dimension
96      sk=fread(fid,1,'short');  % feature-kind
97
98  % setter antall rammer for baade etikettvektor og datamatrise
99      [M N] = min([length(tar) nframes]);
100     if(N==1)
101     counter = counter +1;
102     end
103     M = M -3;
104
105     disp(sprintf('
            |-------------------------------------------------|-------|------------|-------
            '))
106     disp(sprintf('|%-52s|%-8.0d|%-12.0d|%-12.0d|%-14.0d|',
            mfcfile,M,sp,ssf,sk))
107
108     fea_mat=zeros(ssf,M);
109     fea_mat1=fread(fid,M*ssf,'float');
110     fea_mat(:)=fea_mat1(1:M*ssf);
111     fea_mat = fea_mat';
112     fclose(fid);
113
114 fea_all = [fea_all;fea_mat];
115 tar_all = [tar_all; tar(2:M+1)];
116 centerTar_all = [centerTar_all; centerTar(2:M+1)];
117
118 end
119 totalFrames = length(tar_all);
120     disp(sprintf('
            |-------------------------------------------------|--------------------------
            '))
121     disp(sprintf('|                              Total number of
            frames:   |%-8.0d
                                                    |',totalFrames)
            )
```

```
122     disp(sprintf('
            |-----------------------------------------------------------------
            n\n'))
123 fclose(fid_mlf);
124 fclose(fid_mfc);
125
126 % Lagrer data
127 filepath =['/u/studenter/fasselan/HMM_testing/Test/tempFiles/
       mfc_label_' type '_' attribute '_' mfcT '.mat'];
128 delete (filepath)
129 save (filepath,'fea_all','tar_all','centerTar_all','counter')
130
131 % Lagrer etikettfiler for alle attributtyper
132 target_label = ['/u/studenter/fasselan/HMM_testing/Test/
       tempFiles/target_label_' type '.mat'];
133 load(target_label);
134 type
135 sT = size(target_label_all)
136 lengdeTar = length(tar_all)
137 target_label_all = [target_label_all tar_all];
138 save (target_label,'target_label_all')
```

C.1.5 read_models.m

```
1  function r = read_models(mmfFile,modelList,nM,vS)
2  % Leser modellparameter fra HMM (mmf-fil) generert av HTK
3  % Inndata
4  % mmfFile : mmf-fil
5  % modelList: fil med liste over modellene (attributt og anti-
       attributt)
6  % nM : antall blandingskomponenter
7  % vS : antall egenskaper
8
9  disp(mmfFile)
10 nclass = 2;
11 endian = 'b';
12
13 % liste over modeller
14 filename = modelList;
15 fid = fopen(filename,'r',endian);
16 phone=fscanf(fid,'%s\n',1);
```

```
17 fonmap = cellstr(phone);
18 for i = 2:nclass
19     phone=fscanf(fid,'%s\n',1);
20     fonmap = [fonmap cellstr(phone)];
21 end
22 fclose(fid);
23 for i = 1:nclass
24     t = char(fonmap(i));
25 end
26
27 if nM ≥ 2
28 jump = 10;
29 else
30 jump = 8;
31 end
32
33 dropR = 0;
34 currAtt = fonmap(1);
35
36 % Leser mmf-file
37 fid_init = fopen(mmfFile,'r',endian);
38 for j = 1:jump
39     s1 = fscanf(fid_init,'%s\n',1);
40     switch j
41         case 2
42             s = fscanf(fid_init,'%d',1);
43             s2 = fscanf(fid_init,'%d\n',1);
44             xdim = s2;
45         case 3
46             s = fscanf(fid_init,'%s\n',1);
47         case 4
48             s = fscanf(fid_init,'%s\n',1);
49         case 6
50             s2 = fscanf(fid_init,'%d\n',1);
51             nstate = s2 - 2
52         case 7
53             s = fscanf(fid_init,'%s %d\n',1);
54         case 8
55         if jump == 10
56                 s2 = fscanf(fid_init,'%d\n',1);
57                 mix = s2
58         else
59         mix = 1
60         s = fscanf(fid_init,'%s %d\n',1);
```

```
61                  nonsense = fscanf(fid_init,'%d\n',1);
62          end
63      case 9
64           s = fscanf(fid_init,'%s\n',1);
65      case 10
66                  s2 = fscanf(fid_init,'%d\n',1);
67                  blbabal = s2;
68
69      end
70  end
71  fclose(fid_init);
72
73  a=zeros(nstate,nstate,nclass);
74  mum=zeros(nstate*mix,xdim,nclass);
75  mixWeight = zeros(nstate*mix,nclass);
76  stm = mum;
77  numMixInUse = 0;
78  mixMix = 0;
79  attrMix = 0;
80  antiAttrMix = 0;
81
82  fid2 = fopen(mmfFile,'r','b');
83
84  for class = 1 : nclass
85      dropR = 0;
86      mixMix = 0;
87      model=char(fonmap(class));
88          for j = 1:6
89              tull = fgetl(fid2);
90          end
91          for i=1:nstate
92                  ss=fscanf(fid2,'%s',1);
93                  s(i)=fscanf(fid2,'%d\n',1);
94              if (mix ≥ 2)
95                  numM=fscanf(fid2,'%s',1);
96                  numMval=fscanf(fid2,'%d\n',1);
97              end
98              for p = 1:mix
99          mixMix = mixMix + 1;
100                 if (mix > 1)
101             if dropR == 0
102             mixNrTag=fscanf(fid2,'%s',1);
103             end
104         dropR = 0;
```

```
105                 mixNr=fscanf(fid2,'%d',1);
106                 mixW=fscanf(fid2,'%f\n',1);
107                 mixWeight((i-1)*mix+p,class)=mixW;
108         else
109         mixWeight(1,class) = 1;
110         mixWeight(1,class) = 1;
111                 end
112
113                 mean_s=fscanf(fid2,'%s',1);
114                 v_size=fscanf(fid2,'%d\n',1);
115         temp = fscanf(fid2,'%f\n',v_size);
116                 temp1 = sprintf('%d',temp);
117                 mum((i-1)*mix+p,:,class) = temp;
118                 var_s=fscanf(fid2,'%s',1);
119                 var_size=fscanf(fid2,'%d\n',1);
120                 temp2 = fscanf(fid2,'%f\n',var_size);
121                 temp3 = sprintf('%d',temp2);
122         temp2(find(temp2==0)) = 0.0001;
123                 stm((i-1)*mix+p,:,class) = temp2;
124                 gcs=fscanf(fid2,'%s',1);
125
126             if (eq(gcs(2),'G')==1)
127             gc=fscanf(fid2,'%f',1);
128             trans_mean =fscanf(fid2,'%s',1);
129                 if(eq(trans_mean(2),'T')==1)
130                 disp('fant en <TRANSP>, hopper ut')
131                 dropR = 1;
132                 break;
133                 end
134             dropR = 1;
135             elseif(eq(gcs(2),'T')==1)
136             dropR = 1;
137             break;
138             else
139             dropR = 1;
140             end
141
142             if mixNr == mix
143             break;
144             end
145             end
146         end
147         if dropR == 0
148         trs=fscanf(fid2,'%s',1);
```

```
149          else
150          disp('og havnet her')
151          end
152          trns=fscanf(fid2,'%d\n',1);
153          addstate=nstate+2;
154
155          for i=0:addstate-1
156          for h=0:addstate-1
157          dummy=fscanf(fid2,'%f',1);
158          a(i+1,h+1,class)=dummy;
159          end
160          end
161          ehs=fscanf(fid2,'%s\n',1);
162
163      if class == 1
164      attrMix = mixMix
165      else
166      antiAttrMix = mixMix
167      end
168
169
170  end
171  fclose(fid2);
172
173  % Lagrer data
174  filepath =strcat('/u/studenter/fasselan/HMM_testing/Test/
         tempFiles/model_',char(currAtt),'_M',num2str(nM),'.mat');
175  delete (filepath)
176  save (filepath, 'mum', 'stm','attrMix','antiAttrMix', '
         mixWeight', 'a', 'fonmap')
```

C.1.6 pdfCalc2.m

```
1  function r = pdfCalc(type,nstate,mix,pathStr2,vS,currAttr,mfcT)
2  % Kalkulerer logaritmisk pdf-verdien for hver ramme
3  % Inndata:
4  % type: train eller test
5  % nstate: antall tilstander (ikke anvendt mer enn 1)
6  % mix : antall blandingsfordelinger
7  % pathStr2: fil til attributtens modellparametere
8  % currAtt: attributten som prosesseres
```

```
9  % mfcT: streng som beskriver anvendte egenskapsvektoren
10
11 disp(' ')
12 disp('---------- pdfCalc2.m ----------')
13 % Henter inn data prosessert av read_mfc_label2.m
14 pathStr3 = ['/u/studenter/fasselan/HMM_testing/Test/tempFiles/
       mfc_label_' type '_' currAttr '_' mfcT '.mat'];
15
16 load(pathStr2);
17 load(pathStr3);
18 [Nframes manPl] = size(tar_all);
19
20 % beregner standardavvik fra varians
21 std = sqrt(stm(:,:,:));
22
23 nstate = nstate - 2;
24 nclass = 2;
25 xdim = vS
26 sizeFea = size(fea_all)
27 sizeMum = size(mum)
28 sizeStd = size(std)
29
30 temp = 0;
31 temp2 = 0;
32 temp3 = 0;
33 mum2 = zeros(mix,xdim,nclass);
34 std2 = zeros(mix,xdim,nclass);
35 mixWeight2 = zeros(mix,nclass);
36 [a b c] = size(mum);
37
38 mum2 = mum;
39 std2 = std;
40 proddiff = [];
41 proddiffLog = [];
42 mixWeight
43 mixWeightNan = sum(sum(isnan(mixWeight)));
44 feaNaN = sum(sum(isnan(fea_all)));
45 mumNaN = sum(sum(isnan(mum)));
46 stmNaN = sum(sum(isnan(stm)));
47 y1_prod = zeros(Nframes,1);
48 y2_prod = zeros(Nframes,1);
49 sansTest = 0;
50
```

```
% Beregner den samlede pdf-verdien paa tvers av
    blandingskomponenter
for k =1:1:attrMix
    m11 = ones(Nframes,1)*mum2(k,:,1);
    std11 = ones(Nframes,1)*std2(k,:,1);
    pdf1_array= normpdf(fea_all,m11,std11);
    y1_prod = y1_prod + prod(pdf1_array(:,:),2)*mixWeight(k,1);
end

for k =1:1:antiAttrMix
    m22 = ones(Nframes,1)*mum2(k,:,2);
    std22 = ones(Nframes,1)*std2(k,:,2);
    pdf2_array= normpdf(fea_all,m22,std22);
    y2_prod = y2_prod + prod(pdf2_array(:,:),2)*mixWeight(k,1);
end

% Finner differansen mellom logaritmeverdien fra attributt og
    anti-attributt
proddiff = log(y1_prod + exp(-745)) - log(y2_prod + exp(-745));
proddiffLog = [proddiff tar_all centerTar_all];

manIndex = find(proddiffLog(:,2)==1);
man = proddiffLog(manIndex,1);
notmanIndex = find(proddiffLog(:,2)~= 1);
notman = proddiffLog(notmanIndex,1);
naivTerskel = abs(manMean - notmanMean)/2 + min(notmanMean,
    manMean);

eval(strcat('pdf_',type,'_',int2str(mix),'=[proddiffLog];'));
filepath = ['pdfCalc_',type,'_',int2str(mix),'_',currAttr,'.mat
    '];

% Lagre data
delete(filepath)
save(filepath,'pdf1_array','pdf2_array',strcat('pdf_',type,'_',
    int2str(mix)))
r = [naivTerskel attrMix antiAttrMix];
disp('----------END pdfCalc2.m ----------')
```

C.1.7 optThresEERMER.m

```
function r = optThresEERMER(mix,decRule,currAttr,targetChoice)
% Beregner terskelverdi basert paa EER eller MER
%  mix: antall blandingskomponenter
%  decRule: 1 = EER, -1 = MER
% currAtt: attributten som prosesseres
% targetChoice: 2 for vanlig gjenkjenning, 3 for aa gjenkjenne
    kun midtrammer

disp(' ')
disp('---------- optThresEERMER.m ----------')

% Laster inn den logaritmiske differanseverdien for datasettet
pathStr = ['pdfCalc_train_' num2str(mix) '_' currAttr '.mat'];
load(pathStr)
numMix =mix;
proddiffLog = [];
eval(strcat('proddiffLog =', 'pdf_train_',int2str(numMix),';'))
    ;
[Nframes dum] = size(proddiffLog);

% Velger ut attributtrammer og anti-attributtrammer
manIndex = find(proddiffLog(:,targetChoice)==1);
man = proddiffLog(manIndex,1);
notmanIndex = find(proddiffLog(:,targetChoice)~= 1);
notman = proddiffLog(notmanIndex,1);

if length(manIndex) > 0
manMean = mean(man)
else
manMean = 0
end
notmanMean = mean(notman)
signTh =1;

% sjekker om middelverdi til attributtene er stoerre/mindre enn
    resten
% Bergner nedre og oevre grense for terskelverdien
if notmanMean > manMean
    disp('notmanMean is larger than manMean!!!')
    signTh=-1;
    difference = (manMean -notmanMean)*-1;
    st=manMean - difference*0.5;
    step=difference/100;
    stop = notmanMean + difference*1.5;
```

```
42 else
43     difference = (manMean -notmanMean);
44     st=notmanMean - difference*0.5;
45     step=difference/100;
46     stop = manMean + difference*1.5;
47 end
48
49 eerDiff = 10000;
50 tempPerc = 0;
51 thresValue =0;
52 thresValueMax =0;
53 threshold =st:step:stop;
54 numThres = length(threshold);
55 thresh = st;
56
57 % Tester alle terskelverdier og bestemmer seg for den som gir
       best resultat
58 for f = 1:1:numThres
59     frics = find(proddiffLog(:,1)*signTh >thresh*signTh);
60     isFrics = zeros(Nframes,1);
61     isFrics(frics) =1;
62
63     percentageCorrect =0;
64     correct1 = find(proddiffLog(:,2) == 1 & isFrics(:)==1);
65     correct2 = find(proddiffLog(:,2) == 0 & isFrics(:)==0);
66     wrong1 = find(proddiffLog(:,2) == 1 & isFrics(:)==0);
67     wrong2 = find(proddiffLog(:,2) == 0 & isFrics(:)==1);
68
69
70     numcorr1 =length(correct1);
71     numcorr2 =length(correct2);
72     numwro1 =length(wrong1);
73     numwro2 =length(wrong2);
74     if (numcorr1+numwro1) > 0
75     percentageCorrect1 = numcorr1/(numcorr1+numwro1)*100;
76     percentageWrong1 = numwro1/(numcorr1+numwro1)*100;
77     else
78     percentageCorrect1 = 0
79     end
80     percentageCorrect2 = numcorr2/(numcorr2+numwro2)*100;
81     percentageWrong2 = numwro2/(numcorr2+numwro2)*100;
82     percentageCorrect = percentageCorrect1 + percentageCorrect2
           ;
83
```

```
84      eerDiffTemp = percentageWrong2 - percentageWrong1*decRule;
85
86      if percentageCorrect > tempPerc
87      tempPerc = percentageCorrect;
88      thresValueMax = thresh;
89      end
90
91  eerDiffTemp = abs(eerDiffTemp);
92      if eerDiffTemp < eerDiff
93          eerDiff = eerDiffTemp;
94      thresh;
95      utskrift = [percentageCorrect1 percentageWrong1;
            percentageWrong2 percentageCorrect2];
96          thresValue = threshold(f);
97      end
98      thresh = thresh + step;
99  end
100
101 str = strcat('thresholdValue_',int2str(numMix),'=thresValue');
102 eval([str ';']);
103 filepath = ['optThres_',int2str(numMix)];
104
105 % Lagre data
106 save(filepath,strcat('thresholdValue_',int2str(numMix)),'signTh
        ')
107 r= [thresValue signTh];
108 disp('----------END optThresEERMER.m ----------')
```

C.1.8 testData2.m

```
1   function r = testData2(testdatapath,pa2,numMix,recTeller,
        currAttr,targetChoice)
2   % Funksjon som beregner gjenkjenningsraten til attributt X
3   % Inndata:
4   % testdatapath : filnavn for lagring av testdata
5   % pa2 : fil som inneholder terskelverdier
6   % numMix : antall blandingsfordelinger
7   % recTeller : iterasjonsnummer (for oekning i
        blandingsfordelinger)
8   % currAtt: attributten som prosesseres
```

```
% targetChoice: 2 for vanlig gjenkjenning, 3 for aa gjenkjenne
    kun midtrammer

disp(' ');
disp('----- testData2.m -----\n');

% Laster inn den logaritmiske differanseverdien for datasettet
pa1 = ['pdfCalc_test_' int2str(numMix) '_' currAttr '.mat'];
load(pa1) % pdf values
load(pa2) % threshold values
proddiffLog = [];
eval(strcat('proddiffLog =', 'pdf_test_',int2str(numMix),';'));
[Nframes manPl] = size(proddiffLog);
thresh = thresholds(recTeller)

% Sjekker fortegnsverdi fra terskelberegning.
signThVal = signTh(recTeller);
if(signThVal == 1)
manEstIndex = find(proddiffLog(1:Nframes,1) > thresh);
else
manEstIndex = find(proddiffLog(1:Nframes,1) < thresh);
end
manEst = zeros(Nframes,1);
manEst(manEstIndex) =1;
tar = proddiffLog(:,2);

% Beregner antall transisjonsrammer som feildetekteres
edgeFrames = 0;
tempCount1 = 0;
tempCount2 = 0;
for i=3:Nframes-2
    if tar(i) == 1
        if tar(i) ≠ tar(i-1)
            if manEst(i) ≠ tar(i)
            edgeFrames = edgeFrames + 1;
            tempCount1 = i;
            end
            if manEst(i+1) ≠ tar(i+1)
            edgeFrames = edgeFrames + 1;
            tempCount2 = i+1;
            end
        end
        if tar(i) ≠ tar(i+1)
            if manEst(i) ≠ tar(i) && i ≠ tempCount2
```

```
52              edgeFrames = edgeFrames + 1;
53              end
54              if manEst(i-1) ≠ tar(i-1)  && (i-1) ≠ tempCount1
55              edgeFrames = edgeFrames + 1;
56              end
57          end
58      end
59  end
60
61  edgeError = edgeFrames/Nframes;
62  eArray = [edgeError 0; 0 0];
63
64  % kalkulerer gjenkjenningsraten:
65  % Korrekt gjenkjent rammer for attributt X
66  % Korrekt gjenkjent rammer for anti-attributt X
67  % Feil aksepterte rammer
68  % Feil forkastede rammer
69  correct1 = find(proddiffLog(:,targetChoice) == 1 & manEst(:)==
        1);
70  correct2 = find(proddiffLog(:,targetChoice) == 0 & manEst(:)==
        0);
71  wrong1 = find(proddiffLog(:,targetChoice) == 1 & manEst(:)== 0)
        ;
72  wrong2 = find(proddiffLog(:,targetChoice) == 0 & manEst(:)== 1)
        ;
73  corr1 = zeros(Nframes,1);
74  corr2 = zeros(Nframes,1);
75  wro1 = zeros(Nframes,1);
76  wro2 = zeros(Nframes,1);
77  corr1(correct1)=1;
78  corr2(correct2)=1;
79  wro1(wrong1)=1;
80  wro2(wrong2)=1;
81  numcorr1 = sum(corr1);
82  numcorr2 = sum(corr2);
83  numwro1 = sum(wro1);
84  numwro2 = sum(wro2);
85  if (numcorr1+numwro1) > 0
86  perCorr1 = (numcorr1)/(numcorr1+numwro1)*100;
87  perWro1 = (numwro1)/(numcorr1+numwro1)*100;
88  else
89  perCorr1 = 0;
90  perWro1 = 0;
91  end
```

```
92  perCorr2 = (numcorr2)/(numcorr2+numwro2)*100;
93  perWro2 = (numwro2)/(numcorr2+numwro2)*100;
94  numCorr = [numcorr1 numwro1;numwro2 numcorr2];
95  percentage_array = [perCorr1 perWro1;perWro2 perCorr2];
96
97  disp(sprintf('|-------------|          |-------------|'));
98  disp(sprintf('|  Result %%   |          |   Result    |'));
99  disp(sprintf('|-------------|          |-------------|\n\n'));
100 disp(sprintf('|-------------|          |-------------|'));
101 disp(sprintf('|%-6.2f|%-6.2f|          |%-6.0f|%-6.0f|',
        percentage_array(1,1),percentage_array(1,2),numCorr(1,1),
        numCorr(1,2)));
102 disp(sprintf('|------|------|          |------|------|   '));
103 disp(sprintf('|%-6.2f|%-6.2f|          |%-6.0f|%-6.0f|',
        percentage_array(2,1),percentage_array(2,2),numCorr(2,1),
        numCorr(2,2)));
104 disp(sprintf('|-------------|          |-------------|'));
105
106 r= [percentage_array numCorr eArray];
107 % Lagrer data
108 filepath = testdatapath;
109 eval(strcat('resultArray_',int2str(numMix),'=r;'));
110 delete(filepath)
111 save(filepath,strcat('resultArray_',int2str(numMix)))
```

C.1.9 featureSelect.m

```
1  % Funksjon for aa velge ut egnede egenskaper basert paa
       forhaandslagde matriser. de blir lagret under matlab-mappen
        som feaSel.mat
2
3  disp('------------ featureSelect.m ------------');
4
5  % filnavn til parametfil inputVar.lis
6  path_inputVar = '/u/studenter/fasselan/HMM_testing/Test/lists
       /462_set/inputVar.lis';
7
8  % Innlesing av parameter fra inputVar.lis
9  varArrayInputVar = textread(
10 path_inputVar,'%s');
11 feaSelStatus = str2double(char(varArrayInputVar(15)));
```

```
12 mfcTrigger = str2double(char(varArrayInputVar(9)));
13 path = char(varArray(17));
14 TIMITpath = char(varArray(18));
15 listpath = char(varArray(19));
16 matlabpath = char(varArray(20));
17
18
19 % filnavn til parametfil feaSelVar.lis
20 path_feaSelVar = '/u/studenter/fasselan/HMM_testing/Test/lists
       /462_set/feaSelVar.lis';
21
22 % Innlesing av parameter fra feaSelVar.lis
23 varArray = textread(path_feaSelVar,'%s');
24 attributeNr = str2double(char(varArray(1)))
25 filelist_train = char(varArray(2))
26 filelist_trainORG = char(varArray(3))
27 NFiles = str2double(char(varArray(4)))
28 type = char(varArray(6))
29 filelistSel = char(varArray(7))
30
31 filepath = [path matlabpath '/feaSel.mat'];
32 load(filepath);
33
34 % velger egenskapsmatrise
35 all = ones(20,14);
36 allMFCC = ones(13,14);
37 if feaSelStatus == 1
38 feaSel = filtrert;
39 elseif feaSelStatus == 2
40 feaSel = notFilt_med_usikre;
41 elseif feaSelStatus == 3
42 feaSel = notFilt_sikker;
43 elseif feaSelStatus == 4
44 feaSel = allMFCC;
45 elseif feaSelStatus == 5
46 feaSel = notFilt_med_flere_usikre;
47 elseif feaSelStatus == 6
48 feaSel = notFilt_mhp_Korr;
49 else
50 feaSel = all;
51 end
52
53 feaSelArray = feaSel(:,attributeNr);
54
```

```
55 if feaSelStatus == 2
56 feaSelArray(13) = 1;
57 end
58
59 antFea = sum(feaSelArray)
60 ind = find(feaSelArray==1);
61 ind = ind';
62 s = eq(type(2),'r');
63 if s == 1
64 disp('TRAIN')
65 fid_mfc_INN = fopen(filelist_train,'r','b');
66 fid_mfc_OUT = fopen(filelistSel,'r','b');
67 else
68 disp('TEST')
69 filelist_test = filelist_trainORG;
70 fid_mfc_INN = fopen(filelist_test,'r','b');
71 fid_mfc_OUT = fopen(filelistSel,'r','b');
72 end
73
74 fea_all = [];
75
76 % ved bruk av mfcc paa htk-format kopieres de direkte
77 if feaSelStatus == 4
78 copyfile(['/u/studenter/fasselan/TIMIT/htk_mfc_' type '/*'],['/
       u/studenter/fasselan/TIMIT/feaSel_' type])
79 else
80
81     % Et sett med egenskaper blir valgt ut for hver ramme
82     disp(' ---- Selection starts ---- ');
83     for kfile = 1:NFiles
84     %lese fil
85     fileINN = fgetl(fid_mfc_INN);
86     fid2 = fopen(fileINN,'rb','b');
87     nframes2=fread(fid2,1,'int32');  % number of frames
88     sp2=fread(fid2,1,'int32');  % frame step in micro-sec
89     ssb2=fread(fid2,1,'short');  % number of bytes
90     ssf2=ssb2/4;                % vector dimension
91     sk2=fread(fid2,1,'short');  % feature-kind
92
93     fea_mat2=zeros(ssf2,nframes2);
94     fea_mat12=fread(fid2,nframes2*ssf2,'float');
95     fea_mat2(:)=fea_mat12(1:nframes2*ssf2);
96     fea_mat2 = fea_mat2';
97     fclose(fid2);
```

```
98
99      [foer_a foer_b] = size(fea_mat2);
100     fea = fea_mat2(:,ind);
101     [nf vectorSize] = size(fea);
102     fea_all = [fea_all;fea];
103     fea = fea';
104     fea_sel = fea(:);
105     vecS = vectorSize*4;
106
107     %skrive til fil
108     fileOUT = fgetl(fid_mfc_OUT);
109     fid=fopen(fileOUT,'w','b');
110         fwrite(fid,nframes2,'int32');
111         fwrite(fid,sp2,'int32');
112         fwrite(fid,vecS,'short');
113         fwrite(fid,sk2,'short');
114         fwrite(fid,fea_sel,'float');
115     fclose(fid);
116
117     end
118 disp(' ---- Feature selection done ---- ');
119 end
```

C.1.10 testTrainData.m

```
1  function r = testTrainData(testdatapath,thresh,numMix,currAttr,
       targetChoice,trainRecAll,signTh)
2  % Funksjon for beregning av gjenkjenningsrate fra
       treningssettet
3  % Inndata:
4  % testdatapath : filnavn for lagring av testdata
5  % thresh: terskel
6  % numMix : antall blandingsfordelinger
7  % currAtt: attributten som prosesseres
8  % targetChoice: 2 for vanlig gjenkjenning, 3 for aa gjenkjenne
       kun midtrammer
9  % trainRecAll: fil for lagring av alle resultater fra trening
       paa tvers av attributter
10 % signTh : fortegn for terskelverdi
11
12
```

```
13 disp(' ');
14 disp('----- testTrainData.m -----\n');
15
16 % Laster inn den logaritmiske differanseverdien for datasettet
17 pa1 = ['pdfCalc_train_' int2str(numMix) '_' currAttr '.mat'];
18 load(pa1)
19 load(trainRecAll);
20 proddiffLog = [];
21 eval(strcat('proddiffLog =', 'pdf_train_',int2str(numMix),';'))
       ;
22 [Nframes manPl] = size(proddiffLog);
23
24 % Sjekker fortegnsverdi fra terskelberegning.
25 signThVal = signTh;
26 if(signThVal == 1)
27 manEstIndex = find(proddiffLog(1:Nframes,1) > thresh);
28 else
29 manEstIndex = find(proddiffLog(1:Nframes,1) < thresh);
30 end
31 manEst = zeros(Nframes,1);
32 manEst(manEstIndex) =1;
33
34 % kalkulerer gjenkjenningsraten:
35 % Korrekt gjenkjent rammer for attributt X
36 % Korrekt gjenkjent rammer for anti-attributt X
37 % Feil aksepterte rammer
38 % Feil forkastede rammer
39 correct1 = find(proddiffLog(:,targetChoice) == 1 & manEst(:)==
       1);
40 correct2 = find(proddiffLog(:,targetChoice) == 0 & manEst(:)==
       0);
41 wrong1 = find(proddiffLog(:,targetChoice) == 1 & manEst(:)== 0)
       ;
42 wrong2 = find(proddiffLog(:,targetChoice) == 0 & manEst(:)== 1)
       ;
43 corr1 = zeros(Nframes,1);
44 corr2 = zeros(Nframes,1);
45 wro1 = zeros(Nframes,1);
46 wro2 = zeros(Nframes,1);
47 corr1(correct1)=1;
48 corr2(correct2)=1;
49 wro1(wrong1)=1;
50 wro2(wrong2)=1;
51 numcorr1 = sum(corr1);
```

```
numcorr2 = sum(corr2);
numwro1 = sum(wro1);
numwro2 = sum(wro2);
if (numcorr1+numwro1) > 0
perCorr1 = (numcorr1)/(numcorr1+numwro1)*100;
perWro1 = (numwro1)/(numcorr1+numwro1)*100;
else
perCorr1 = 0;
perWro1 = 0;
end
perCorr2 = (numcorr2)/(numcorr2+numwro2)*100;
perWro2 = (numwro2)/(numcorr2+numwro2)*100;
numCorr = [numcorr1 numwro1 numwro2 numcorr2];
percentage_array = [perCorr1 perWro1 perWro2 perCorr2];

trainNumResult = [trainNumResult; numCorr];
trainResult = [trainResult; percentage_array];

% Lagrer data
save([trainRecAll '.mat'],'trainNumResult','trainResult')
save([trainRecAll '_M' int2str(numMix) '.mat'],'trainNumResult'
    ,'trainResult')
r = trainNumResult
```

C.1.11 displayMixInc.m

```
% Henter alle resultater fra gjenkjenningen og presenter dem
    som leservennlig txt-fil

disp('----------- displayMixInc.m ------------');

% filnavn til parametfil inputVar.lis
path_inputVar = '/u/studenter/fasselan/HMM_testing/Test/lists
    /462_set/inputVar.lis';

% Innlesing av parameter fra inputVar.lis
varArray = textread(path_inputVar,'%s');
p = str2double(char(varArray(1)));
nS = str2double(char(varArray(2)));
nM = str2double(char(varArray(3)));
vS = str2double(char(varArray(4)));
```

```
14 Nfiles = str2double(char(varArray(5)));
15 numberOfAtt = str2double(char(varArray(6)));
16 decRule = str2double(char(varArray(7)));
17 NfilesTest = str2double(char(varArray(8)));
18 mfcTrigger = str2double(char(varArray(9)));
19 antIt = str2double(char(varArray(13)));
20 iterationNumber = str2double(char(varArray(14)));
21 path = char(varArray(17));
22 TIMITpath = char(varArray(18));
23 listpath = char(varArray(19));
24
25 filepath = [path 'tempFiles/' 'recData.mat'];
26 load(filepath);
27
28 if p == 1
29 p = 'phonemes';
30 else
31 p = 'attributes';
32 end
33
34
35 if decRule == 1
36 decRule = 'EER';
37 else
38 decRule = 'MER';
39 end
40
41 if mfcTrigger == 1
42 mfcT = 'Feature_Vectors';
43 elseif mfcTrigger == 2
44 mfcT = 'MFCCs';
45 elseif mfcTrigger == 3
46 mfcT = 'Feature_Vectors_AND_MFCCs';
47 elseif mfcTrigger == 4
48 mfcT = 'Feature_Vectors_(MFCC_E)';
49 else
50 mfcT = 'Feature_Vectors_(MFCC_E)_FILT';
51 end
52
53 % Leser inna resultater fra alle iterasjonene og legger dem i
      allRes
54 mix = [1 2 4 6 8 10 12 14 16 18 20];
55 allRes = zeros(numberOfAtt,11,antIt);
56 for i = 1:antIt
```

```
57
58 recDataMix = [path 'tempFiles/' 'recData_ItNum' int2str(i) '
       .mat'];
59 load(recDataMix);
60     for attr = 1:1:numberOfAtt
61
62     err = ((finalResNUM(1,2,attr) + finalResNUM(2,1,attr))/(
           finalResNUM(1,1,attr)+finalResNUM(1,2,attr)+finalResNUM
           (2,1,attr)+finalResNUM(2,2,attr)))*100;
63
64     cor =  ((finalResNUM(1,1,attr) + finalResNUM(2,2,attr))/(
           finalResNUM(1,1,attr)+finalResNUM(1,2,attr)+finalResNUM
           (2,1,attr)+finalResNUM(2,2,attr)))*100;
65
66     allRes(attr,1,i) = finalRes(1,1,attr);
67     allRes(attr,2,i) = finalRes(1,2,attr);
68     allRes(attr,3,i) = finalRes(2,1,attr);
69     allRes(attr,4,i) = finalRes(2,2,attr);
70     allRes(attr,5,i) = err;
71     allRes(attr,6,i) = cor;
72     allRes(attr,7,i) = edgeError(attr)*100;
73     allRes(attr,8,i) = finalResNUM(1,1,attr);
74     allRes(attr,9,i) = finalResNUM(1,2,attr);
75     allRes(attr,10,i) = finalResNUM(2,1,attr);
76     allRes(attr,11,i) = finalResNUM(2,2,attr);
77     end
78 end
79
80 fid = fopen('/u/studenter/fasselan/HMM_testing/Test/lists/462
       _set/attributes.lis','r','b');
81 t = fscanf(fid,'%s\n',1);
82
83     if(strcmp(t,'q')==1)
84     attributeArray = textread('/u/studenter/fasselan/
           HMM_testing/Test/lists/462_set/phonemes.lis','%s');
85
86     else
87     attributeArray = textread('/u/studenter/fasselan/
           HMM_testing/Test/lists/462_set/
           attributes_REAL_NAMES.lis','%s');
88     end
89     fclose(fid);
90
91
```

```
92 % Presenterer resultatene
93 for mixI = 0:1:antIt
94 if mixI == 0
95 disp(sprintf('                        '));
96 disp(sprintf('      |%-19s|%-19s|%-19s|%-19s|',char(
       attributeArray(1)),char(attributeArray(2)),char(
       attributeArray(3)),char(attributeArray(4))))
97 else
98 disp(sprintf('%-6.0d|     %-8.2f(%-2.2f)|     %-8.2f(%-2.2f)|
           %-8.2f(%-2.2f)|     %-8.2f(%-2.2f)|',mix(mixI),allRes
       (1,5,mixI),allRes(1,7,mixI),allRes(2,5,mixI),allRes(2,7,
       mixI),allRes(3,5,mixI),allRes(3,7,mixI),allRes(4,5,mixI),
       allRes(4,7,mixI)));
99 end
100
101 end
102
103
104 for mixI = 0:1:antIt
105 if mixI == 0
106 disp(sprintf('                        '));
107 disp(sprintf('      |%-19s|%-19s|%-19s|%-19s|',char(
       attributeArray(5)),char(attributeArray(6)),char(
       attributeArray(7)),char(attributeArray(8))));
108 else
109 disp(sprintf('%-6.0d|     %-8.2f(%-2.2f)|     %-8.2f(%-2.2f)|
           %-8.2f(%-2.2f)|     %-8.2f(%-2.2f)|',mix(mixI),allRes
       (5,5,mixI),allRes(5,7,mixI),allRes(6,5,mixI),allRes(6,7,
       mixI),allRes(7,5,mixI),allRes(7,7,mixI),allRes(8,5,mixI),
       allRes(8,7,mixI)));
110 end
111
112 end
113
114 for mixI = 0:1:antIt
115 if mixI == 0
116 disp(sprintf('                        '));
117 disp(sprintf('      |%-19s|%-19s|%-19s|%-19s|',char(
       attributeArray(9)),char(attributeArray(10)),char(
       attributeArray(11)),char(attributeArray(12))));
118 else
119 disp(sprintf('%-6.0d|     %-8.2f(%-2.2f)|     %-8.2f(%-2.2f)|
           %-8.2f(%-2.2f)|     %-8.2f(%-2.2f)|',mix(mixI),allRes
       (9,5,mixI),allRes(9,7,mixI),allRes(10,5,mixI),allRes(10,7,
```

```
        mixI),allRes(11,5,mixI),allRes(11,7,mixI),allRes(12,5,mixI)
        ,allRes(12,7,mixI)));
120 end
121
122 end
123 for mixI = 0:1:antIt
124     if mixI == 0
125     disp(sprintf('                        '));
126     disp(sprintf('      |%-19s|%-19s|',char(attributeArray(13))
            ,char(attributeArray(14))));
127     else
128     disp(sprintf('%-6.0d|     %-8.2f(%-2.2f)|     %-8.2f(%-2.2f
            )|',mix(mixI),allRes(13,5,mixI),allRes(13,7,mixI),
            allRes(14,5,mixI),allRes(14,7,mixI)));
129     end
130 end
131 gjennomsnittFeil = mean(allRes(:,5,antIt))
132
133 TESTFriknum = [allRes(2,8,antIt) allRes(2,9,antIt);allRes(2,10,
        antIt) allRes(2,11,antIt)]
134 TESTVokalnum = [allRes(5,8,antIt) allRes(5,9,antIt); allRes
        (5,10,antIt) allRes(5,11,antIt)]
135 TESTFriknumProsent = [allRes(2,1,antIt) allRes(2,2,antIt);
        allRes(2,3,antIt) allRes(2,4,antIt)]
136 TESTVokalnumProsent = [allRes(5,1,antIt) allRes(5,2,antIt);
        allRes(5,3,antIt) allRes(5,4,antIt)]
137
138
139 trainAllRes = zeros(numberOfAtt,8,antIt);
140 for i = 1:antIt
141 trainDataResult = [path 'tempFiles/' 'trainResult_M' int2str(
        mix(i)) '.mat'];
142 load(trainDataResult);
143     for attr = 1:numberOfAtt
144     trainAllRes(attr,1,i) = trainNumResult(attr,1);
145     trainAllRes(attr,2,i) = trainNumResult(attr,2);
146     trainAllRes(attr,3,i) = trainNumResult(attr,3);
147     trainAllRes(attr,4,i) = trainNumResult(attr,4);
148     trainAllRes(attr,5,i) = trainResult(attr,1);
149     trainAllRes(attr,6,i) = trainResult(attr,2);
150     trainAllRes(attr,7,i) = trainResult(attr,3);
151     trainAllRes(attr,8,i) = trainResult(attr,4);
152     end
153 end
```

```
154
155 TRENINGFrikRes = [trainAllRes(2,1,antIt) trainAllRes(2,2,antIt)
        ;trainAllRes(2,3,antIt) trainAllRes(2,4,antIt)]
156 TRENINGVokalRes = [trainAllRes(5,1,antIt) trainAllRes(5,2,antIt
        );trainAllRes(5,3,antIt) trainAllRes(5,4,antIt)]
157 TRENINGFrikResProsent = [trainAllRes(2,5,antIt) trainAllRes
        (2,6,antIt);trainAllRes(2,7,antIt) trainAllRes(2,8,antIt)]
158 TRENINGVokalResProsent = [trainAllRes(5,5,antIt) trainAllRes
        (5,6,antIt);trainAllRes(5,7,antIt) trainAllRes(5,8,antIt)]
159
160 % Lagrer data
161 filepath =strcat('/u/studenter/fasselan/HMM_testing/Test/
        Results/Result_fra_displayMix/res_',mfcT,'.mat');
162 save(filepath,'allRes');
```

C.2 Perl

C.2.1 quickstartup_trainScript

```
#!/usr/bin/perl
# Startskript for gjenkjenningsverktoeyet

# sette filbaner
$TestDir = "/u/studenter/fasselan/HMM_testing/Test";
$TIMITpath = "/u/studenter/fasselan/TIMIT/";
$set462 = "lists/462_set";
$Matlab = "Matlab";

# Initiere datafiler
system "matlab -nojvm -nodisplay < $Matlab/
    initTEMPResult.m > $Matlab/initTEMPResultLog.txt";
system "matlab -nojvm -nodisplay < $Matlab/initTrainRec
    .m > $Matlab/initTrainRecLog.txt";

# sette parameterverdier
$nS = 3;
$phon = 0; # 0= attributter og 1=fonemer (Systemet
    fungerer ikke fult ut med fonemer)
$NFilesTrain = 305; #4620; # max = 305
$NFilesTest = 157; #1680; # max = 157
$decRule = "EER";
$antIt = 6;
$mfcTrigger = 2; # 2 = MFCC, 4 = egendefinerte
    egenskaper
$extraEst = 0; # eldre variabel som er utelatt
$targetChoice = 2; #% 2 for vanlig target, 3 for center
     targets
$feaSel = 4; # 0=alle 20 egenskaper, 1 = filtrert (ikke
     anvendt), 2= med usikre, 3= sikre (eksperimet 3),
    4=htkMFCC_all, 5=med flere usikre (eksperiment 2),
    6=mhp Korr
$numHRestIt = 9; # antall iterasjoner som HRest skal
    utfoere

if ($phon == 0){
$numberOfAtt = 14;
}
else{
```

```
$numberOfAtt = 56;
}

if ($mfcTrigger == 1){
$vS = 40;
}
elsif ($mfcTrigger == 2){
$vS = 13;
}
elsif ($mfcTrigger == 3){
$vS = 33;
}
elsif ($mfcTrigger == 4 || $mfcTrigger == 5){
$vS = 20;
}
elsif ($mfcTrigger == 6){
$vS = 9;
}
else {
$vS = 40;
}

$startRec = "Y";

if ($decRule eq "EER") {
$decRule = 1;
$decRuleName = "EER";
}
else {
$decRule = -1;
$decRuleName = "MER";
}


#delete all hmm-models
system "rm `find hmms/ -name *.mmf`";
system "rm `find targetHMM/ -name *.mmf`";

# Starter gjenkjenning. Hver iterasjon gaar gjennom
    hele systemet med et gitt antall GMM-komponenter
# foerste bruker 1, saa 2, deretter 4, 6, 8... osv opp
    til antIt = f.eks 5 iterasjoner opp til 8
    komponenter
$initMix = 1;
```

```
$nM = 1;
for($n=1;$n<=$antIt;$n++){
$iterationNumber = $n;
# skriver inn data i inputVar.lis
# De fleste andre skript trenger disse parameterne.
open (VAR,">$TestDir/$set462/inputVar.lis") || die ("
    Unable to open $TestDir/$set462/inputVar.lis");
print VAR ("$phon\n");
print VAR ("$nS\n");
print VAR ("$nM\n");
print VAR ("$vS\n");
print VAR ("$NFilesTrain\n");
print VAR ("$numberOfAtt\n");
print VAR ("$decRule\n");
print VAR ("$NFilesTest\n");
print VAR ("$mfcTrigger\n");
print VAR ("$startRec\n");
print VAR ("$extraEst\n");
print VAR ("$targetChoice\n");
print VAR ("$antIt\n");
print VAR ("$iterationNumber\n");
print VAR ("$feaSel\n");
print VAR ("$numHRestIt\n");
print VAR ("$TestDir\n");
print VAR ("$TIMITpath\n");
print VAR ("$set462\n");
print VAR ("$Matlab\n");
close(VAR);

# Startet trainScript
system "./trainScript";

if($n == 1){
$nM = 2;
}
else{
$nM = $nM + 2;
}
}
```

C.2.2 trainScript

```
#!/usr/bin/perl
# trainScript beregner HMM for hver attributt/anti-
```

```
    attributt vha HTK.
# Skriptet vil starte Matlab-funksjoner for
    terskelberegning

print "trainScript started\n";

# filbaner
$TestDir = "/u/studenter/fasselan/HMM_testing/Test";
$TIMITDir = "/u/studenter/fasselan/TIMIT";
$set462 = "lists/462_set";
$outputMLFHLEd = "attribute.mlf";
$outputMLFHLEd_HTK = "attribute_HTK.mlf";
$orgLedFile = "phonManPla.led";
$Matlab = "Matlab";
$inputAttr = "$TestDir/$set462/attributes.lis";
$attr = "$TestDir/$set462/man_pla_attributes.lis";
$phoneme = "$TestDir/$set462/phonemes.lis";
$inputmlf = "$TestDir/$set462/train462.mlf";
$inputmlf2 = "$TestDir/$set462/train462MFC.mlf";
$protoHmm = "$TestDir/protoHmm";
$targetHMMdir = "$TestDir/targetHMM";
$protodir = "$TestDir/proto";
$attrONLY = "$TestDir/$set462/att.lis";
$attrONLY_man = "$TestDir/$set462/a.lis";
$attrONLY_anti = "$TestDir/$set462/aa.lis";
$hmmdir = "$TestDir/hmms";
$filelist1 = "$TestDir/$set462/trainAll_mfc.lis";
$filelist2 = "$TestDir/$set462/train462_mfc2.lis";
$filelist3 = "$TestDir/$set462/train462_htk_AND_FEmfc.
    lis";
$filelist4 = "$TestDir/$set462/train462_mfc_NEW.lis";
$filelist5 = "$TestDir/$set462/
    train462_mfc_NEW_filtered.lis";
$filelist6 = "$TestDir/$set462/
    train462_mfc_NEW_reducedFEA.lis";
$config1 = "$TestDir/configs/ClausConfig.conf";
$config2 = "$TestDir/configs/ClausConfigMFCC_E.conf";

$traceHCompV = 1;
$traceHRest = 1;
$floorMINMIX = 0;

# henter data fra inputVar.lis
open (VAR,"$TestDir/$set462/inputVar.lis") || die ("
```

```
    Unable to open $TestDir/$set462/inputVar.lis");
@inputVar = <VAR>;
close(VAR);
chomp(@inputVar);
$phon = $inputVar[0];
$nS = $inputVar[1];
$nM = $inputVar[2];
$vS = $inputVar[3];
$NFilesTrain = $inputVar[4];
$numberOfAtt = $inputVar[5];
$decRule = $inputVar[6];
$NFilesTest = $inputVar[7];
$mfcTrigger = $inputVar[8];
$startRec = $inputVar[9];
$extraEst = $inputVar[10];
$feaSel = $inputVar[14];
$numHRestIt = $inputVar[15];

if ($NFilesTrain == 4620){
$filelistSel = "$TestDir/$set462/trainAll_feaSel.lis";
}
else{
$filelistSel = "$TestDir/$set462/train462_feaSel.lis";
}

# velger antall egenskaper for hver attributt
if ($feaSel == 1){
@vectorSize =
    (17,12,16,16,15,14,10,14,14,16,14,14,15,14);
}
elsif($feaSel == 2){
#@vectorSize = (6,3,4,7,5,9,1,5,6,5,14,11,9,8);
@vectorSize = (7,4,5,8,6,10,2,6,7,6,15,12,10,9);
}
elsif($feaSel == 3){# sikre
@vectorSize = (6,3,4,5,4,9,1,5,6,2,8,11,9,8);
#@vectorSize = (7,4,5,6,5,10,2,6,7,3,9,12,10,9);
}
elsif($feaSel == 4){
@vectorSize =
    (13,13,13,13,13,13,13,13,13,13,13,13,13,13);
}
elsif($feaSel == 5){
@vectorSize = (10,8,11,11,10,12,7,10,10,9,15,14,12,10);
```

```
}
elsif($feaSel == 6){
@vectorSize = (4,4,5,5,5,5,4,6,4,4,8,8,6,4);
}
else{
@vectorSize =
    (20,20,20,20,20,20,20,20,20,20,20,20,20,20);
}

print "feaSel is: $feaSel\n";

if ($decRule == 1) {
$decRule = 1;
$decRuleName = "EER";
}
else {
$decRule = -1;
$decRuleName = "MER";
}

if ($mfcTrigger == 1) {
$mfcTriggerDesc = "Feature vectors";
$config = $config2;
$filelistOrg = $filelist1;
$Type = "MFCC_E";
}
elsif ($mfcTrigger == 2){
$mfcTriggerDesc = "MFCC";
$config = $config2;
$filelistOrg = $filelist2;
$Type = "MFCC_E";
}
elsif ($mfcTrigger == 3){
$mfcTriggerDesc = "Feature vectors & MFCC";
$config = $config2;
$filelistOrg = $filelist3;
$Type = "MFCC_E";
}
elsif ($mfcTrigger == 4) {
$mfcTriggerDesc = "Feature vectors_E";
$config = $config2;
$filelistOrg = $filelist4;
$Type = "MFCC_E";
}
```

```
elsif ($mfcTrigger == 6) {
$mfcTriggerDesc = "Feature vectors_E_redFEA";
$config = $config2;
$filelistOrg = $filelist6;
$Type = "MFCC_E";
}
else {
$mfcTriggerDesc = "Feature vectors_E_FILT";
$config = $config2;
$filelistOrg = $filelist5;
$Type = "MFCC_E";
}

# lager nye filister mhp paa utvelgelsen av nye
    egenskaper
$filelistEDIT = "$TestDir/$set462/train_edited.lis";
open (FILELIST,">$filelistEDIT") || die ("Unable to
    open $filelistEDIT");
        open (ORGFILES,"$filelistOrg") || die ("Unable
            to open $filelistOrg");
        @allFiles = <ORGFILES>;
        close(VAR);
@SelectedFiles = @allFiles[0..$NFilesTrain-1];
print FILELIST @SelectedFiles;
close(FILELIST);

$tempfilelist = "$TestDir/$set462/train_temp.lis";
open (FILELIST,">$tempfilelist") || die ("Unable to
    open $tempfilelist");
        open (ORGFILES,"$filelistSel") || die ("Unable
            to open $filelistSel");
        @allFiles = <ORGFILES>;
        close(VAR);
@SelectedFiles = @allFiles[0..$NFilesTrain-1];
print FILELIST @SelectedFiles;
close(FILELIST);
$filelist = $tempfilelist;


if($phon==1){
$stringAtt = "phonemes";
}
else{
$stringAtt = "attributes";
```

```
}

if($phon==1){
system "cp $phoneme $inputAttr";
}
else {
system "cp $attr $inputAttr";
}

# Initierer av datafiler
system "matlab -nojvm -nodisplay < $Matlab/initThres.m
    > initThres.txt";
system "matlab -nojvm -nodisplay < $Matlab/initTrainRec
    .m> initTrainLog.txt";

open (ATTR, "$inputAttr") || die ("Unable to open
    $inputAttr");
@attrList = <ATTR>;
close(ATTR);
pop(@attrList);

$nM_Init = 1;
$count = 0;

# starter gjenkjenning. Loekke gaar gjennom en gang per
    attributt
foreach $attribute (@attrList){
        $count += 1;
        if($count > $numberOfAtt ) {
        last;
        }
        chomp($attribute);
        $attribute =~ s/\s+$//;
        print"$stringAtt number: $count\n";

        # Skript som endrer "alfabet"-filene etter
            hvilken attributt som prosesseres
        if ($phon == 1){
        system "./makePhonemeDictLed $attribute
            $orgLedFile";
        }
        else{
        system "./makeDictLed $attribute $orgLedFile";
        }
```

```
# Skript som endrer mmf-fil etter hvilken
    attributt som prosesseres
system "HLEd -X phn -l '*' -d $TestDir/$set462/
    att.dict -i $TestDir/$set462/$outputMLFHLEd
    $TestDir/$set462/att.led $inputmlf";
system "HLEd -X phn -l '*' -d $TestDir/$set462/
    att.dict -i $TestDir/$set462/
    $outputMLFHLEd_HTK $TestDir/$set462/att.led
    $inputmlf2";
$inputMLFHERest = $outputMLFHLEd_HTK;
$vS = $vectorSize[$count - 1];

# Legger inn parametere i feaSelVar.lis
open (FEASEL,">$TestDir/$set462/feaSelVar.lis")
     || die ("Unable to open $TestDir/$set462/
    feaSelVar.lis");
print FEASEL ("$count\n");
print FEASEL ("$filelistEDIT\n");
print FEASEL ("$filelistOrg\n");
print FEASEL ("$NFilesTrain\n");
print FEASEL ("$vS\n");
print FEASEL ("train\n");
print FEASEL ("$filelistSel\n");
close(FEASEL);

print "Vector size is: $vS\n";

# Omgjoer datafiler ved aa velge ut enkelte
    egenskaper
system "matlab -nojvm -nodisplay < $Matlab/
    featureSelect.m > $Matlab/logfiles/
    LogfeatureSelect_train_$attribute.txt";


if ($nM==1){
        $proto = "protoM$nM_Init\S$nS\V$vS";
        system "./tools/makeProtoClaus $nS
            $nM_Init $vS $Type";

        #HCompV: initialiserer HMM
        system "HCompV -f 0.01 -v 0.01 -T
            $traceHCompV -C $config $trace -m -S
            $filelist -o $attribute -M
```

```
            $targetHMMdir $protodir/$proto";

        system "HCompV -f 0.01 -v 0.01 -T
            $traceHCompV -C $config $trace -m -S
            $filelist -o A$attribute -M
            $targetHMMdir $protodir/$proto";

        # reestimerer HMM
        system "HRest -t -v 0.01 -T $traceHRest
            -X phn -I $TestDir/$set462/
            $inputMLFHERest -C $config -i
            $numHRestIt -l $attribute -S
            $filelist -M $hmmdir $targetHMMdir/
            $attribute";

        system "HRest -t -v 0.01 -T $traceHRest
            -X phn -I $TestDir/$set462/
            $inputMLFHERest -C $config -i
            $numHRestIt -l A$attribute -S
            $filelist -M $hmmdir $targetHMMdir/
            A$attribute";

$temp1 = "$TestDir/$set462/a.lis";
$temp2 = "$TestDir/$set462/aa.lis";

# Inkrementerer antall blandingskomponeter til
    2
if($nM == 2){
# endrer HMM etter oekning av antall
    blandingskomponeter
system "HHEd -H $hmmdir/$attribute -M $hmmdir/
    hmm2 $TestDir/$set462/incMix.2.hed
    $attrONLY_man";

system "HHEd -H $hmmdir/A$attribute -M $hmmdir/
    hmm2 $TestDir/$set462/incMix.2.hed
    $attrONLY_anti";

system "HRest -t -v 0.01 -T $traceHRest -X phn
    -I $TestDir/$set462/$inputMLFHERest -C
    $config -i $numHRestIt -l $attribute -S
    $filelist -M $hmmdir/hmm22 $hmmdir/hmm2/
    $attribute";
```

```
system "HRest -t -v 0.01 -T $traceHRest -X phn
    -I $TestDir/$set462/$inputMLFHERest -C
    $config -i $numHRestIt -l A$attribute -S
    $filelist -M $hmmdir/hmm22 $hmmdir/hmm2/
    A$attribute";
}

# Inkrementerer antall blandingskomponeter
if($nM >= 4){
        $nF = $nM-2;
        # endrer HMM etter oekning av antall
            blandingskomponeter
        system "HHEd -H $hmmdir/hmm$nF$nF/
            $attribute -M $hmmdir/hmm$nM
            $TestDir/$set462/incMix.$nM.hed
            $attrONLY_man";

        system "HHEd -H $hmmdir/hmm$nF$nF/
            A$attribute -M $hmmdir/hmm$nM
            $TestDir/$set462/incMix.$nM.hed
            $attrONLY_anti";

        # reestimerer HMM
        system "HRest -t -v 0.01 -T $traceHRest
            -X phn -I $TestDir/$set462/
            $inputMLFHERest -C $config -i
            $numHRestIt -l $attribute -S
            $filelist -M $hmmdir/hmm$nM$nM
            $hmmdir/hmm$nM/$attribute";

        system "HRest -t -v 0.01 -T $traceHRest
            -X phn -I $TestDir/$set462/
            $inputMLFHERest -C $config -i
            $numHRestIt -l A$attribute -S
            $filelist -M $hmmdir/hmm$nM$nM
            $hmmdir/hmm$nM/A$attribute";
}

if($nM==1){
system "cat $hmmdir/A$attribute $hmmdir/
    $attribute > $hmmdir/attrMMF_$attribute.mmf"
    ;
}
else{
```

```
        system "cat $hmmdir/hmm$nM$nM/A$attribute
            $hmmdir/hmm$nM$nM/$attribute > $hmmdir/
            hmm$nM$nM/attrMMF_$attribute.mmf";
        }


        # Kjoerer matlab-funksjonen trainMatlab.m
        print "Running Matlab for attribute $attribute\
            n\n";
        system "matlab -nojvm -nodisplay < $Matlab/
            trainMatlab.m > $Matlab/logfiles/
            trainLog_$attribute\_$nM.txt";

}

print "********** End of trainScript
    ******************\n";
print "* Detector type : ________________$stringAtt__
    ____\n";
print "* Number of Mixtures: ___________$nM_________
    ____\n";
print "* Number of States: _____________$nS_________
    ____\n";
print "* Vector size : _________________$vS_________
    ____\n";
print "* Number of files (training): ____$NFilesTrain
    ____\n";
print "* Decision rule: _______________$decRuleName
    ____\n";
print "* Speech representation: _________$mfcTriggerDesc
    _\n";
print "
    ***********************************************\n
    \n\n";


# Starter recScript etter at alle attributter er
    prosessert
if ($startRec eq "Y"){
print "** Automatically starts the recSript as
    initially requested **\n";
system "./recScript";
}
```

C.2.3 recScript

```
#!/usr/bin/perl
# recScript tar for seg utvelgelsen av aktuell
    attributt og endrer etikettfiler etter den.
# Skriptet vil starte Matlab-funksjoner for
    gjenkjenning og
# til slutt funksjoner for aa presentere resultat

print "recScript started\n";

# filbaner
$outputmlf = "RecAttribute.mlf";
$orgLedFile = "phonManPla.led";
$TestDir = "/u/studenter/fasselan/HMM_testing/Test";
$TIMITDir = "/u/studenter/fasselan/TIMIT";
$set462 = "lists/462_set";
$Matlab = "Matlab";
$Results = "Results";
$tempFiles = "tempFiles";
$inputmlf = "$TestDir/$set462/test462.mlf";
$inputAttr = "$TestDir/$set462/attributes.lis";
$attr = "$TestDir/$set462/man_pla_attributes.lis";
$phoneme = "$TestDir/$set462/phonemes.lis";
$filelist1 = "$TestDir/$set462/testAll_mfc.lis";
$filelist2 = "$TestDir/$set462/test462_mfc2.lis";
$filelist3 = "$TestDir/$set462/test462_htk_AND_FEmfc.
    lis";
$filelist4 = "$TestDir/$set462/test462_mfc_NEW.lis";
$filelist5 = "$TestDir/$set462/test462_mfc_NEW_filtered
    .lis";
$filelist6 = "$TestDir/$set462/
    test462_mfc_NEW_reducedFEA.lis";
$trace = 0;

# henter data fra inputVar.lis
open (VAR,"$TestDir/$set462/inputVar.lis") || die ("
    Unable to open $TestDir/$set462/inputVar.lis");
@inputVar = <VAR>;
close(VAR);
chomp(@inputVar);
$phon = $inputVar[0];
$nS = $inputVar[1];
$nM = $inputVar[2];
```

```
$numberOfAtt = $inputVar[5];
$decRule = $inputVar[6];
$NFiles = $inputVar[7];
$mfcTrigger = $inputVar[8];
$antIt = $inputVar[12];
$iterationNumber = $inputVar[13];
$feaSel = $inputVar[14];

if ($NFiles == 1680){
$filelistSel = "$TestDir/$set462/testAll_feaSel.lis";
}
else{
$filelistSel = "$TestDir/$set462/test462_feaSel.lis";
}

# velger antall egenskaper for hver attributt
if ($feaSel == 1){
@vectorSize =
   (17,12,16,16,15,14,10,14,14,16,14,14,15,14);
}
elsif($feaSel == 2){
#@vectorSize = (6,3,4,7,5,9,1,5,6,5,14,11,9,8);
@vectorSize = (7,4,5,8,6,10,2,6,7,6,15,12,10,9);
}
elsif($feaSel == 3){
#@vectorSize = (6,3,4,5,4,9,1,5,6,2,8,11,9,8);
@vectorSize = (7,4,5,6,5,10,2,6,7,3,9,12,10,9);
}
elsif($feaSel == 4){
@vectorSize =
   (13,13,13,13,13,13,13,13,13,13,13,13,13,13);
}
elsif($feaSel == 5){
@vectorSize = (10,8,11,11,10,12,7,10,10,9,15,14,12,10);
}
elsif($feaSel == 6){
@vectorSize = (4,4,5,5,5,5,4,6,4,4,8,8,6,4);
}
else{
@vectorSize =
   (20,20,20,20,20,20,20,20,20,20,20,20,20,20);
}

if ($decRule == 1) {
```

```
$decRuleName = "EER";
}
else {
$decRuleName = "MER";
}

if ($mfcTrigger == 1) {
$mfcTriggerDesc = "Feature vectors";
}
elsif ($mfcTrigger == 2){
$mfcTriggerDesc = "MFCC";
}
elsif ($mfcTrigger == 3){
$mfcTriggerDesc = "Feature vectors & MFCC";
}
elsif ($mfcTrigger == 4){
$mfcTriggerDesc = "Feature vectors_E";
}
else {
$mfcTriggerDesc = "Feature vectors_E_FILT";
}

if ($mfcTrigger == 1) {
$mfcTriggerDesc = "Feature vectors";
$config = $config1;
$filelistOrg = $filelist1;
$Type = "MFCC_E_D";
}
elsif ($mfcTrigger == 2){
$mfcTriggerDesc = "MFCC";
$config = $config2;
$filelistOrg = $filelist2;
$Type = "MFCC_E";
}
elsif ($mfcTrigger == 3){
$mfcTriggerDesc = "Feature vectors & MFCC";
$config = $config2;
$filelistOrg = $filelist3;
$Type = "MFCC_E";
}
elsif ($mfcTrigger == 4) {
$mfcTriggerDesc = "Feature vectors_E";
$config = $config2;
$filelistOrg = $filelist4;
```

```
$Type = "MFCC_E";
}
elsif ($mfcTrigger == 6) {
$mfcTriggerDesc = "Feature vectors_E_redFEA";
$config = $config2;
$filelistOrg = $filelist6;
$Type = "MFCC_E";
}


sub round {
    my($number) = shift;
    return int($number + .5);
}

if($phon==1){
$stringAtt = "phonemes";
}
else{
$stringAtt = "attributes";
}

if($phon==1){
system "cp $phoneme $inputAttr";
}
else {
system "cp $attr $inputAttr";
}

# Initierer resultatfiler
system "matlab -nojvm -nodisplay < $Matlab/initResult.m
    > $Matlab/initResultLog.txt";

open (ATTR, "$inputAttr") || die ("Unable to open
    $inputAttr");
@attrList = <ATTR>;
close(ATTR);
pop(@attrList);

# starter gjenkjenning. Loekke gaar gjennom en gang per
    attributt
$count = 0;
foreach $attribute (@attrList){
        $count += 1;
```

```
        chomp($attribute);

        $vS = @vectorSize[$count-1];
        print "vector size is $vS\n";

        $filelistEDIT = "dummy";
        open (FEASEL,">$TestDir/$set462/feaSelVar.lis")
            || die ("Unable to open $TestDir/$set462/
            feaSelVar.lis");
        print FEASEL ("$count\n");
        print FEASEL ("$filelistEDIT\n");
        print FEASEL ("$filelistOrg\n");
        print FEASEL ("$NFiles\n");
        print FEASEL ("$vS\n");
        print FEASEL ("test\n");
        print FEASEL ("$filelistSel\n");
        close(FEASEL);

        # Omgjoer datafiler ved aa velge ut enkelte
            egenskaper
        system "matlab -nojvm -nodisplay < $Matlab/
            featureSelect.m > $Matlab/logfiles/
            LogfeatureSelect_test_$attribute.txt";

        # Skript som endrer "alfabet"-filene etter
            hvilken attributt som prosesseres
        if ($phon == 1){
        system "./makePhonemeDictLed $attribute
            $orgLedFile";
        }
        else{
        system "./makeDictLed $attribute $orgLedFile";
        }

        # Skript som endrer mmf-fil etter hvilken
            attributt som prosesseres
        system "HLEd -X phn -l '*' -d $TestDir/$set462/
            att.dict -i $TestDir/$set462/$outputmlf
            $TestDir/$set462/att.led $inputmlf";


# starter matlabfunksjonen, recMatlab.m
print "kjoerer naa Matlab for attribute $attribute\n\n"
    ;
```

```
system "matlab -nojvm -nodisplay < $Matlab/recMatlab.m
    > $Matlab/logfiles/recLog_$attribute\_$nM.txt";
          print"$count\n";
          if($count >= $numberOfAtt ) {
          last;
          }
}

system "cp $TestDir/$tempFiles/recData.mat $TestDir/
    $tempFiles/recData_ItNum$iterationNumber.mat";

# lager leservennlig oversiktstabell over resultatene
    for alle attributter og for alle antall
    blandingskomponenter
if ($iterationNumber == $antIt){
system "matlab -nojvm -nodisplay < $Matlab/
    displayMixInc.m > $Results/MixInc_F$NFiles\
    _$decRuleName\_$mfcTriggerDesc.txt";
}

# kopierer datafile til resultatmappe
system "cp $TestDir/$tempFiles/recData.mat $TestDir/
    $Results/recData_M$nM\S$nS\V$vS\_F$NFiles\
    _$stringAtt\_$decRuleName.mat";



print "**********  End of recScript  
    ******************\n";
print "* Detector type : ______________$stringAtt
    ______\n";
print "* Number of Mixtures : ___________$nM_____
    ________\n";
print "* Number of States : _____________$nS_____
    ________\n";
print "* Vector size : __________________$vS_____
    ________\n";
print "* Number of files (testing): _____$NFiles 
    ________\n";
print "* Decision rule : _______________$decRuleName
    ____\n";
print "* Speech representation : ________$mfcTriggerDesc
    \n";
print "
```

```
**********\n\n\n";
```