

Κωδικοποίηση φωνής για αναγνώριση σε ενσύρματα και ασύρματα δίκτυα δεδομένων(διαδίκτυο,GPRS) μέσα από δίαυλο 2Kbps



ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών
Εργαστήριο Τηλεπικοινωνιών Πληροφορίας και Δικτύων

Περακάκης Μανόλης
Χανιά , Δεκέμβριος 2000

Αφιερώνεται στους γονείς μου
Τηλέμαχο και Γαλήνη Περαιβάκη
για την αγάπη και την
υποστήριξή τους

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον υπεύθυνο της διπλωματικής εργασίας Διγαλάκη Βασίλειο για την συμμετοχή μου στο πολύ ενδιαφέρον αυτό θέμα, καθώς και για την υποστήριξή του στην περάτωση της εργασίας. Επίσης ευχαριστώ και τους συνάδελφους στο εργαστήριο για τη συνεργασία τους.

Αυτό το κείμενο

Αυτό το κείμενο γράφτηκε σε L^AT_EX 2_ε με τη βοήθεια ελληνικών γραμματοσειρών, σε περιβάλλον Linux. Τόσο ο πηγαίος κώδικας (.tex) όσο και όλες οι άλλες μορφές του (.ps, .pdf, .html) είναι διαθέσιμες από την διεύθυνση : <http://www.telecom.tuc.gr/~perak/diploma-thesis>. Στον ίδιο δικτυακό τόπο βρίσκονται και οι διαφάνειες της παρουσίασης σε postscript μορφή (.ps). Οι εικόνες του πρώτου κεφαλαίου είναι από την εργασία [1], ενώ οι υπόλοιπες είναι φτιαγμένες σε xfig. Όλοι οι λατινικοί όροι είναι στοιχειοθετημένοι με italics. Για όποια σχόλια ή ερωτήσεις, παρακαλώ επικοινωνείτε μαζί μου στο : perak@telecom.tuc.gr

Ανάπτυξη κώδικα

Η ανάπτυξη των τεχνικών που περιγράφονται σε αυτή την εργασία έγιναν στα συστήματα Decipher & Yarrow. Και τα δυο είναι copyright (©) του S.R.I (Stanford Research Institute). Το Yarrow API θα είναι διαθέσιμο εντός του εργαστηρίου, καθώς και το Speech Recognition Applet Demo.

Εισαγωγή

Τα τελευταία χρόνια είμαστε όλοι μάρτυρες μιας έκρηξης στο χώρο των δικτύων, σε επίπεδο τόσο *software* όσο και *hardware*, που οδήγησε στην ραγδαία εξάπλωση του παγκόσμιου δικτύου (*World Wide Web*). Αυτή η αλλαγή συνετέλεσε στο να αλλάξει ο κόσμος των υπολογιστών όπως τον ξέραμε. Έτσι, όχι μόνο θα εμφανισθεί μια ολόκληρη νέα γκάμα εφαρμογών για να εκμεταλλευτεί την ιντερνετική τεχνολογία, αλλά γίνεται επιτακτική η ανάγκη να ενσωματωθεί και σε υπάρχουσες εφαρμογές.

Ανάλογες ραγδαίες εξελίξεις βλέπουμε να συντελούνται και στο χώρο των ασύρματων τηλεπικοινωνιών, καθιστώντας δυνατή την επικοινωνία από οπουδήποτε οποτεδήποτε. Τα συμβατικά τηλέφωνα αντικαθίστανται από κινητά τηλέφωνα με εντυπωσιακές δυνατότητες και νέες πρωτοπóρες υπηρεσίες τα καθιστούν πολύτιμα εργαλεία σε αντιστοιχία με τους προσωπικούς υπολογιστές. Η τάση αυτή γίνεται όλο και πιο εμφανής καθώς πλησιάζει η εμφάνιση της νέας 3ης γενεάς συστημάτων κινητής τηλεφωνίας.

Σε αυτό το συναρπαστικό κόσμο της επικοινωνίας, η τεχνολογία αναγνώρισης φωνής καλείται να εξέλθει από τα ερευνητικά εργαστήρια και να διαδραματίσει το ρόλο που της αναλογεί, ο οποίος δεν είναι και μικρός, αν αναλογιστεί κανείς ότι η φωνή αποτελεί τον πιο φυσικό τρόπο επικοινωνίας. Προκειμένου όμως να συμβεί κάτι τέτοιο, επιβάλλεται η προσαρμογή των ήδη υπαρχόντων τέτοιων εφαρμογών στις νέες συνθήκες. Κάτι τέτοιο δεν είναι πάντα εύκολο και συχνά μια τέτοια μετάβαση απαιτεί να λυθούν πολλά προβλήματα και να βρεθούν νέες καταλυτικές μέθοδοι που να καταστήσουν κάτι τέτοιο δυνατό.

Σήμερα, η πιο ευρεία χρησιμοποιούμενη εφαρμογή αναγνώρισης φωνής, είναι η ανάκτηση πληροφοριών μέσω τηλεφωνικού δικτύου. Με σκοπό να καταστήσουμε εύκολη, δυνατή, και με ελάχιστο κόστος την χρήση της αναγνώρισης φωνής μέσω τόσο του διαδικτύου, όσο και ασύρματων δικτύων, ερευνούμε κατάλληλους τρόπους κωδικοποίησης της φωνής, με στόχο οι απαιτήσεις εύρους ζώνης να μην ξεπερνούν τα *2kbp/s*. Κάτι τέτοιο, θα αποτελέσει καταλυτικό παράγοντα για την εκμετάλλευση εφαρμογών αναγνώρισης φωνής στο περι-

βάλλον που μόλις περιγράψαμε.

Για τις εφαρμογές αυτές αναγνώρισης φωνής, προτείνουμε το μοντέλο πελάτη - εξυπηρετητή. Ο πελάτης είναι οποιαδήποτε συσκευή ικανή να παράγει και να αποστέλλει τις κωδικοποιημένες παραμέτρους της φωνής στον εξυπηρετητή ο οποίος αναλαμβάνει το 'δύσκολο' έργο της αναγνώρισης. Η εργασία αυτή επικεντρώνεται στην εύρεση βέλτιστων τρόπων μείωσης της απαιτούμενης πληροφορίας που απαιτείται για την αναπαράσταση των παραμέτρων της φωνής, εισάγοντας ένα νέο σχήμα κωδικοποίησης με στόχο την καλλίτερη δυνατή συμπίεση φωνής προς αναγνώριση. Κάτι τέτοιο μειώνει σημαντικά τις απαιτήσεις εύρους ζώνης του διαύλου μεταξύ πελάτη και εξυπηρετητή, τόσο από την πλευρά του πελάτη, όσο κυρίως από την πλευρά ενός εξυπηρετητή. Μειώνοντας το απαιτούμενο *bandwidth* σε μόλις *2kbps*, καθιστούμε δυνατή την εφαρμογή της αναγνώρισης φωνής στο διαδίκτυο και παρουσιάζουμε μια τέτοια υλοποίηση, όπου ο πελάτης είναι ένα *Java applet*.

Η τεκμηρίωση της εργασίας χωρίζεται σε 5 κεφάλαια :

Κεφάλαιο 1 Γίνεται μια σύντομη εισαγωγή στα συστήματα αναγνώρισης φωνής. Δίνεται η συνολική εικόνα ενός τέτοιου συστήματος με όλα τα υποσυστήματα που το αποτελούν, ενώ στη συνέχεια μελετώνται τα σημαντικότερα τέτοια υποσυστήματα, όπως ο μηχανισμός δημιουργίας παραμέτρων φωνής, τα ακουστικά μοντέλα καθώς και η κατάλληλη επιλογή αυτών για την περίπτωση αυτής της εργασίας.

Κεφάλαιο 2 Το πρώτο μέρος αυτού του Κεφαλαίου, αρχίζει με το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση βασικών εννοιών. Γίνεται μια αναλυτική παρουσίαση για τις πηγές και μοντέλα πληροφορίας, την θεωρία πηγής, ρυθμού-απώλειας και κωδικοποίησης πηγής χωρίς απώλειες. Παρουσιάζονται επίσης οι τεχνικές κβαντισμού με έμφαση σε τεχνικές διανυσματικού κβαντισμού, ενώ ακολουθεί μια καταγραφή των τεχνικών κωδικοποίησης φωνής.

Στο δεύτερο μέρος δίνεται μια αναλυτική περιγραφή του σχήματος κωδικοποίησης φωνής αυτής της εργασίας, η οποία αποσκοπεί στην συμπίεση φωνής προς αναγνώριση. Αναλύονται οι αλγόριθμοι και περιγράφονται οι διαδικασίες εκπαίδευσης, κβαντισμού και πειραμάτων. Τέλος παρατίθενται τρόποι επιλογής των παραμέτρων του σχήματος κωδικοποίησης και η αξιολόγηση του. Το σχήμα κωδικοποίησης βασίζεται στην ιδέα της διανυσματικής κβαντοποίησης των διανυσμάτων των παραμέτρων της φωνής. Για την υλοποίηση των παραπάνω χρησιμοποιήθηκαν τα συστήματα φωνής *Yarrow* και *Decipher*.

Κεφάλαιο 3 Σε αυτό το κεφάλαιο παρουσιάζεται η υλοποίηση της κωδικοποίησης με το σύστημα *Yarrow*. Επεκτείνοντας το αρχικό σύστημα *Yarrow* με το σχήμα κωδικοποίησης που αναπτύξαμε, το καθιστούμε μια ολοκληρωμένη πλατφόρμα ανάπτυξης εφαρμογών κωδικοποίησης φωνής. Σε επίπεδο κώδικα περιγράφονται αναλυτικά οι βιβλιοθήκες που αναπτύχθηκαν καθώς και οι αλλαγές που έγιναν στο σύστημα αναγνώρισης *Decipher*. Σε επίπεδο εφαρμογής αναλύεται η χρήση του *Yarrow* για τα πειράματα της αξιολόγησης του σχήματος κωδικοποίησης τα οποία περιγράφονται στο Κεφάλαιο 4.

Κεφάλαιο 4 Στο Κεφάλαιο αυτό παρουσιάζονται τα πειράματα κωδικοποίησης που έγιναν με τα συστήματα φωνής *Yarrow* και *Decipher*. Περιγράφονται πειράματα βαθμωτού και διανυσματικού κβαντισμού, ο αλγόριθμος *bit allocation* για την εύρεση του επιτρεπόμενου ορίου συμπίεσης χωρίς απώλειες στην απόδοση αναγνώρισης και η επίδραση του θορύβου στην κωδικοποίηση. Περισσότερα πειράματα υπάρχουν στο Παράρτημα στο τέλος της εργασίας.

Κεφάλαιο 5 Σε αυτό το Κεφάλαιο παρουσιάζονται οι εφαρμογές αναγνώρισης φωνής σήμερα, ενώ στη συνέχεια γίνεται μια εκτενή ανάλυση για τις εξελίξεις σε θέματα δικτύων δεδομένων (ενσύρματων αλλά κυρίως ασύρματων, όπως κινητής τηλεφωνίας 2.5 και 3ης γεννεάς). Αυτό γίνεται, γιατί τα ασύρματα δίκτυα ιδιαίτερα, αναμένεται να αποτελέσουν ιδανικό περιβάλλον για ανάπτυξη εφαρμογών που θα εκμεταλλεύονται την αναγνώριση φωνής με βάση το μοντέλο που προτείνουμε, κυρίως για εφαρμογές ανάκτησης πληροφοριών από ανάλογες συσκευές ασύρματης ανάκτησης πληροφοριών. Στη συνέχεια παρουσιάζεται μια υλοποίηση του μοντέλου πελάτη - εξυπηρετητή για αναγνώριση φωνής στο διαδίκτυο. Περιγράφεται το *Speech Recognition Applet Demo*, το οποίο χρησιμοποιεί το σύστημα *Yarrow* στον πελάτη για την κωδικοποίηση της φωνής και το σύστημα αναγνώρισης φωνής *Decipher* στον εξυπηρετητή. Η επικοινωνία γίνεται μέσα από ένα δίαυλο εύρους *2kbps*.

Περιεχόμενα

| | | |
|----------|-----------------------------------------------------------------|-----------|
| 1 | Σύστημα αναγνώρισης φωνής | 2 |
| 1.1 | Εισαγωγή | 2 |
| 1.2 | Συνολική εικόνα ενός συστήματος αναγνώρισης ομιλίας | 2 |
| 1.3 | Ο front-end μηχανισμός | 3 |
| 1.3.1 | Η διαδικασία κβαντισμού των παραμέτρων του front-end | 5 |
| 1.4 | Acoustic Modeling | 7 |
| 1.5 | Είδη HMMs | 8 |
| 1.5.1 | Διακριτά HMMs | 8 |
| 1.5.2 | Συνεχή HMMs - Gaussian Mixture HMMs | 9 |
| 1.5.3 | Μείγματα διακριτών HMMs από κβαντοποίηση συνεχών HMMs | 10 |
| 1.6 | Συμπεράσματα | 11 |
| 1.7 | Παραπομπές | 11 |
| 2 | Κωδικοποίηση φωνής | 12 |
| 2.1 | Εισαγωγή | 12 |
| 2.2 | Πηγές πληροφορίας και κωδικοποίηση | 13 |
| 2.2.1 | Μοντέλο πληροφορίας | 13 |
| 2.3 | Τεχνικές κβαντισμού | 14 |
| 2.4 | Βαθμωτός κβαντισμός | 15 |
| 2.5 | Διανυσματικός κβαντισμός - VQ | 16 |
| 2.5.1 | Ανάλυση απόδοσης και πλεονεκτήματα VQ | 17 |
| 2.5.2 | Είδη VQ, παράμετροι σχεδιασμού και τεχνικές | 18 |
| 2.6 | Τεχνικές κωδικοποίησης φωνής | 21 |
| 2.7 | Waveform Coding τεχνικές | 21 |
| 2.7.1 | Τεχνικές στο πεδίο του χρόνου | 21 |
| 2.7.2 | Τεχνικές στο πεδίο της συχνότητας | 23 |
| 2.8 | Model based τεχνικές (Vocoders) | 23 |
| 2.9 | Το σχήμα κωδικοποίησης που χρησιμοποιήσαμε | 25 |

| | | |
|----------|-------------------------------------------------------------------------------|-----------|
| 2.9.1 | Βασικές έννοιες | 26 |
| 2.9.2 | Διαδικασία εκπαίδευσης | 27 |
| 2.9.3 | Διαδικασία κβαντισμού | 30 |
| 2.10 | Αξιολόγηση κωδικοποίησης | 32 |
| 2.11 | Συμπεράσματα | 32 |
| 2.12 | Παραπομπές | 33 |
| 3 | Υλοποίηση της κωδικοποίησης - το σύστημα YARROW | 34 |
| 3.1 | Εισαγωγή | 34 |
| 3.2 | Βιβλιοθήκες του αρχικού συστήματος | 35 |
| 3.3 | Επέκταση του συστήματος για υποστήριξη VQ | 36 |
| 3.3.1 | Σκοπός | 36 |
| 3.4 | Αρχιτεκτονική συστήματος και υλοποίηση | 36 |
| 3.5 | Βιβλιοθήκη sri.star.coding.base | 37 |
| 3.6 | Βιβλιοθήκη πελατών (sri.star.coding.clients.*) | 38 |
| 3.6.1 | Πελάτες αρχείων | 38 |
| 3.6.2 | Πελάτες front-end | 40 |
| 3.6.3 | Πελάτες κβαντισμού | 40 |
| 3.6.4 | Πελάτες εκπαίδευσης | 41 |
| 3.6.5 | Υπόλοιποι πελάτες | 41 |
| 3.7 | Βιβλιοθήκη εφαρμογών (sri.star.coding.apps*) | 42 |
| 3.8 | Πειράματα VQ με το Yarrow | 44 |
| 3.8.1 | Παραγωγή επιθυμητού feature από αρχεία κυματομορφών | 44 |
| 3.8.2 | Προετοιμασία εκπαίδευσης | 45 |
| 3.8.3 | Διαδικασία εκπαίδευσης | 45 |
| 3.8.4 | Διαδικασία κβαντισμού | 46 |
| 3.8.5 | Έλεγχος με Αναγνώριση | 47 |
| 3.9 | Αναγνώριση με το Decipher | 48 |
| 3.10 | Συμπεράσματα | 48 |
| 4 | Πειράματα κωδικοποίησης | 50 |
| 4.1 | Διαδικασία πειραμάτων | 51 |
| 4.2 | Baseline πείραμα | 53 |
| 4.3 | Πειράματα βαθμωτού κβαντισμού | 53 |
| 4.4 | Πειράματα διανυσματικού κβαντισμού | 55 |
| 4.4.1 | Επιλογή παραμέτρων συστήματος | 55 |
| 4.4.2 | Επιλογή υποδιανυσμάτων | 55 |
| 4.4.3 | Επιλογή αριθμού bits ανά υποδιάνυσμα : bit allocation αλγόριθμος | 56 |
| 4.4.4 | Πειράματα με θόρυβο. | 58 |

| | | |
|----------|--------------------------------------------------------------------------------|-----------|
| 4.5 | Συμπεράσματα | 58 |
| 5 | Αναγνώριση σε ασύρματα και ενσύρματα δίκτυα δεδομένων | 60 |
| 5.1 | Εισαγωγή | 60 |
| 5.2 | Εφαρμογές αναγνώρισης φωνής σήμερα | 61 |
| 5.3 | Εξελίξεις στο χώρο των επικοινωνιών : δίκτυα δεδομένων | 62 |
| 5.4 | Ασύρματα δίκτυα νέας γενεάς | 62 |
| 5.4.1 | Οφέλη για παροχείς δικτύων, εφαρμογών και τελικό χρήστη | 63 |
| 5.5 | Χρήση κωδικοποίησης φωνής αυτής της εργασίας | 64 |
| 5.6 | Χρήση στις ασύρματες επικοινωνίες | 64 |
| 5.6.1 | Σχετικά πρότυπα : ETSI proposal | 65 |
| 5.6.2 | Τεχνολογία ασύρματων συσκευών πληροφορίας | 65 |
| 5.6.3 | Δίκτυα 2.5 και 3ης γενεάς : υποσχέσεις, περιορισμοί και συμπεράσματα | 66 |
| 5.7 | Χρήση στο διαδίκτυο | 67 |
| 5.8 | Γενική εικόνα του συστήματος | 68 |
| 5.9 | Αρχιτεκτονική του πελάτη | 69 |
| 5.9.1 | Audio Plugin | 69 |
| 5.9.2 | Compression Plugin | 70 |
| 5.9.3 | Network Plugin | 70 |
| 5.10 | Επικοινωνία πελάτη αναγνωριστή | 71 |
| 5.10.1 | Εναλλακτικές προσεγγίσεις | 72 |
| 5.11 | Speech Recognition Applet Demo | 72 |
| 5.11.1 | AudioPanel | 73 |
| 5.11.2 | CompressPanel | 73 |
| 5.11.3 | RecognizePanel | 74 |
| 5.11.4 | HelpPanel | 75 |
| 5.12 | Συμπεράσματα | 75 |
| A-1.1 | Πειράματα bit allocation | 77 |
| A-1.2 | Πείραμα 3 | 78 |
| A-1.3 | Πείραμα 3+ | 79 |
| A-1.4 | Πείραμα 4 | 80 |
| A-1.5 | Πείραμα 5 | 81 |
| A-1.6 | Πείραμα 5+ | 82 |
| A-1.7 | Πείραμα 6 | 83 |
| A-1.8 | Σύγκριση αποτελεσμάτων bit allocation | 83 |

Κατάλογος Σχημάτων

| | | |
|-----|--------------------------------------------------------|----|
| 1.1 | Συνολική εικόνα στατιστικής αναγνώρισης φωνής. | 4 |
| 1.2 | Παραγωγή MFCC διανύσματος. | 6 |
| 1.3 | HMM μοντέλο. | 8 |
| 2.1 | Μοντέλο παραγωγής φωνής | 23 |
| 2.2 | Codebook & Centroids. | 28 |
| 2.3 | Διαδικασία εκπαίδευσης. | 28 |
| 2.4 | Εκπαίδευση και δημιουργία codebook | 29 |
| 2.5 | Διαδικασία κβαντισμού. | 31 |
| 2.6 | Έξοδος συστήματος κβαντιστών και αναγνώριση. | 31 |
| 3.1 | Αρχιτεκτονική Yarrow συστήματος | 36 |
| 4.1 | Διαδικασία πειραμάτων. | 51 |
| 5.1 | Επικοινωνία πελάτη - αναγνωριστή μέσω Web | 69 |
| 5.2 | Το applet με τα 3 plugins που το αποτελούν. | 69 |
| 5.3 | Επικοινωνία πελάτη αναγνωριστή | 71 |
| 5.4 | AudioPanel | 73 |
| 5.5 | CompressPanel | 74 |
| 5.6 | RecognizePanel | 74 |
| 5.7 | HelpPanel | 75 |
| A-1 | Σύγκριση αποτελεσμάτων | 85 |

Κατάλογος Πινάκων

| | | |
|-----|------------------------------------------------------------------------------------|----|
| 3.1 | Παραγόμενα features και καταλήξεις αρχείων | 39 |
| 4.1 | Baseline πείραμα | 53 |
| 4.2 | Bit-rates & WER για scalar quantization , για τηλεφωνικής ποιότητας φωνή | 54 |
| 4.3 | Bit rates & WER για scalar quantization , για υψηλής ποιότητας φωνή | 54 |
| 4.4 | Σύγκριση correlation & knowledge based partitioning | 56 |
| 4.5 | Bit allocation για τα 5 υποδιανύσματα | 57 |
| 4.6 | Bit allocation για τη scalar περίπτωση | 57 |
| 4.7 | Κβαντισμός παρουσία θορύβου | 58 |

Κεφάλαιο 1

Σύστημα αναγνώρισης φωνής

1.1 Εισαγωγή

Τα συστήματα αναγνώρισης ομιλίας είναι πολύπλοκα συστήματα που εμπεριέχουν ένα μεγάλο σύνολο γνωστικών πεδίων όπως : στατιστική επεξεργασία σήματος , κατανόηση φυσικής γλώσσας, νευρωνικά συστήματα, αναγνώριση προτύπων, φωνολογία κ.α. Σε αυτό το εισαγωγικό κεφάλαιο πάνω στη αναγνώριση φωνής, θα μελετήσουμε μόνο τα κομμάτια που έχουν να κάνουν περισσότερο με το θέμα αυτής της εργασίας, που είναι η επεξεργασία του σήματος της φωνής (*front-end processing*) και η αναγνώριση με τη χρήση κρυφών μαρκοβιανών μοντέλων (*Hidden Markov Models - HMMs*). Πρώτα όμως, θα δούμε τη συνολική εικόνα ενός συστήματος αναγνώρισης ομιλίας.

1.2 Συνολική εικόνα ενός συστήματος αναγνώρισης ομιλίας

Τα σύγχρονα συστήματα στηρίζονται στις αρχές της στατιστικής αναγνώρισης προτύπων . Οι βασικές μεθοδολογίες της εφαρμογής των παραπάνω αρχών στο πρόβλημα της αναγνώρισης φωνής μελετήθηκαν και εφαρμόστηκαν στη δεκαετία του 70 στην *IBM*. Η συνολική εικόνα έχει ως εξής :

Μια άγνωστη κυματομορφή φωνής μετατρέπεται από τον *front-end* επεξεργαστή σήματος, σε μια ακολουθία από ακουστικά διανύσματα : $Y = y_1, y_2, \dots, y_t$. Κάθε ένα από τα παραπάνω διανύσματα, αναπαριστά το βραχύως χρόνου φάσμα (*short time speech spectrum*) μιας περιόδου τυπικής διάρκειας *10 msec*s. Έτσι, για παράδειγμα, μια φράση 3 δευτερολέπτων, μπορεί να αναπαρασταθεί από μια ακολουθία 300 τέτοιων διανυσμάτων. Η φράση αποτελείται από μια ακολουθία λέξεων, $W = w_1, w_2, \dots, w_n$ και είναι ευθύνη

του συστήματος αναγνώρισης να βρει την πιο πιθανή ακολουθία λέξεων W , δεδομένου της ακολουθίας του ακουστικού σήματος Y . Για να γίνει αυτό, χρησιμοποιείται ο κανόνας του *Bayes* :

$$\widehat{W} = \arg \max_w P(W|Y) = \arg \max_w \frac{P(W)P(Y|W)}{P(Y)} \quad (1.1)$$

Η παραπάνω εξίσωση δηλώνει ότι η πιο πιθανή ακολουθία λέξεων W , είναι εκείνη που μεγιστοποιεί το γινόμενο των $P(W)$ και $P(Y|W)$. Ο πρώτος όρος αντιπροσωπεύει την *a-priori* πιθανότητα να παρατηρηθεί το W ανεξάρτητα από το ακουστικό σήμα και καθορίζεται από το γλωσσικό μοντέλο (*language model*). Ο δεύτερος όρος αντιπροσωπεύει την πιθανότητα να παρατηρηθεί η ακολουθία Y , δεδομένου της ακολουθίας W , και αυτή η πιθανότητα, καθορίζεται από το ακουστικό μοντέλο (*acoustic model*).

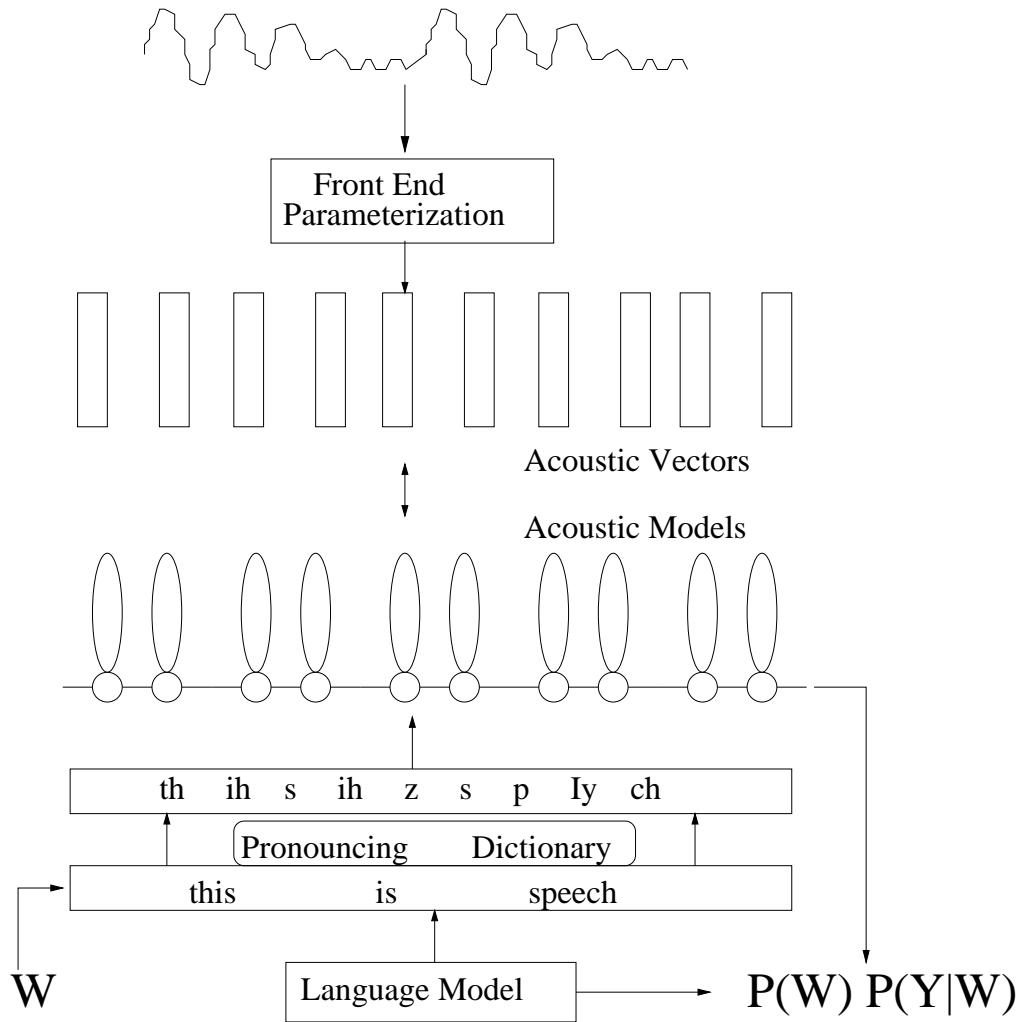
Το Σχήμα 1.1 δείχνει πως τα παραπάνω συνδέονται μεταξύ τους. Η ακολουθία λέξεων « *This is speech* » δίνεται σαν είσοδος και το γλωσσικό μοντέλο υπολογίζει την πιθανότητα $P(W)$. Στη συνέχεια, κάθε λέξη μετατρέπεται σε μια ακολουθία από βασικούς ήχους (*phones*), χρησιμοποιώντας το λεξικό προφοράς (*pronunciation dictionary*). Για κάθε φώνημα υπάρχει ένα αντίστοιχο στατιστικό μοντέλο, το *HMM*. Η ακολουθία των *HMMs* που χρειάζεται για να αναπαραστήσουμε τη φράση συνενώνονται για να σχηματίσουν ένα σύνθετο μοντέλο, και υπολογίζεται η πιθανότητα αυτό το μοντέλο να παράγει την ακολουθία Y , δηλαδή η πιθανότητα $P(Y|W)$.

Στη συνέχεια θα δούμε αναλυτικότερα τα εξής :

- Πως είναι σχεδιασμένος ένας *front-end* μηχανισμός για να μπορεί να εξάγει από την κυματομορφή όλη την απαραίτητη ακουστική πληροφορία σε μια μορφή συμβατή με τα *HMM* ακουστικά μοντέλα.
- Πως τα *HMM* μοντέλα αναπαριστούν τη βασική μονάδα αναγνώρισης που είναι το φώνημα.

1.3 Ο front-end μηχανισμός

Μια βασική υπόθεση στην οποία βασίζονται οι αναγνωριστές, είναι ότι το σήμα της φωνής μπορεί να θεωρηθεί σαν στάσιμο (*stationary*-δηλαδή τα χαρακτηριστικά του φάσματος μπορούν να θεωρηθούν σταθερά) σε ένα διάστημα λίγων *msecs*. Το σήμα φωνής χωρίζεται σε ένα σύνολο από διαστήματα (*frames*) και για κάθε διάστημα υπολογίζεται ένα ομαλοποιημένο φάσμα. Τα διαστήματα έχουν συνήθως μήκος *10 msecs* και αλληλεπικαλύπτονται δίνοντας έτσι ένα



Σχήμα 1.1: Συνολική εικόνα στατιστικής αναγνώρισης φωνής. Το διάγραμμα δείχνει τον υπολογισμό της πιθανότητας $P(W|Y)$ της ακολουθίας λέξεων W δεδομένου του ακουστικού σήματος Y . Η *prior* πιθανότητα $P(W)$ υπολογίζεται απ' ευθείας από το γλωσσικό μοντέλο (*language model*). Η πιθανότητα $P(Y|W)$ υπολογίζεται χρησιμοποιώντας το σύνθετο *HMM* που αναπαριστά το W και κατασκευάζεται από απλά *HMM* φωνητικά μοντέλα ενωμένα στη σειρά, ανάλογα με τις προφορές των λέξεων που είναι αποθηκευμένες στο λεξικό προφορών.

παράθυρο μεγαλύτερου μήκους. Χρησιμοποιώντας για παράδειγμα παράθυρο *Hamming* και εφαρμόζοντας ανάλυση *Fourier* ή γραμμικής πρόβλεψης (*LPC*) παίρνουμε τις βασικές φασματικές παραμέτρους τις οποίες με διάφορους μετασχηματισμούς μετατρέπουμε σε κάποια κατάλληλη μορφή.

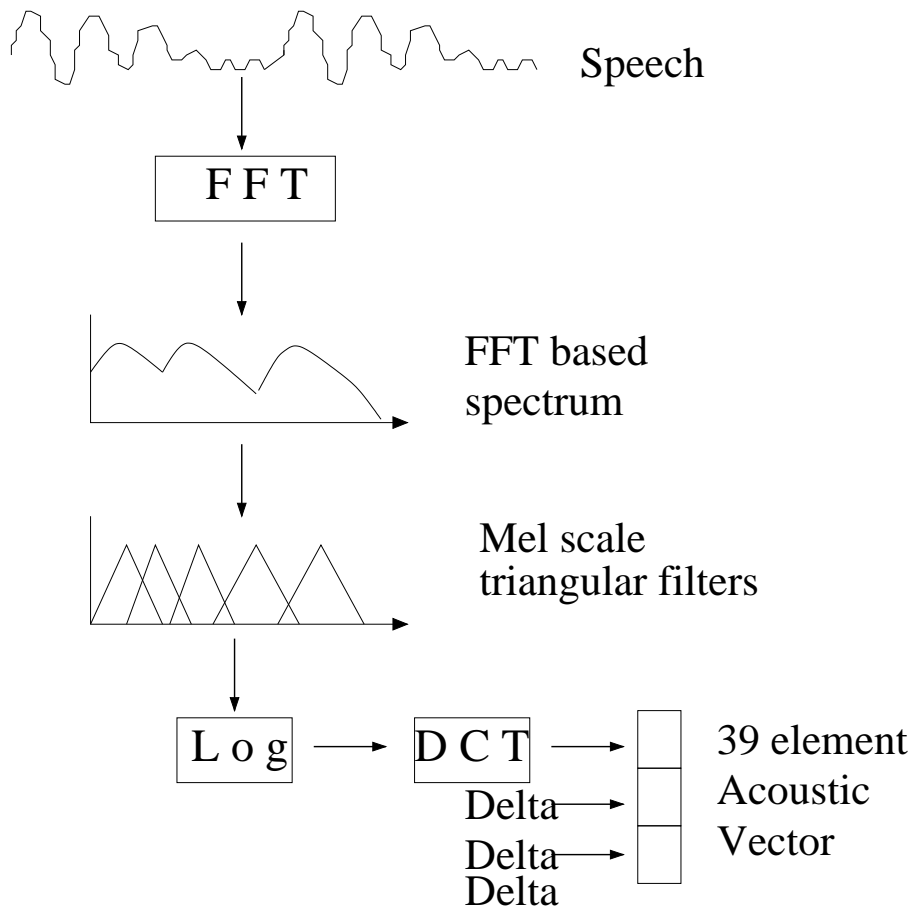
Μια τέτοια μορφή για τα ακουστικά διανύσματα είναι οι *mel-frequency cepstral coefficients (MFCCs)*. Ένας τρόπος για να υπολογιστούν, φαίνεται στο Σχήμα 1.2. Αφού πάρουμε το φάσμα του σήματος εφαρμόζουμε μια κλίμακα (*mel-frequency*) η οποία σχεδιάζεται ουσιαστικά ώστε να προσεγγίζει την φασματική ανάλυση της ανθρώπινης ακοής, που είναι γραμμική έως τα 1000Hz και λογαριθμική από και πέρα. Η κλίμακα αυτή έχει αποδειχθεί πειραματικά ότι βελτιώνει την ακρίβεια αναγνώρισης. Στη συνέχεια, για να κάνουμε την ισχύ του φάσματος περίπου γκαουσιανή εφαρμόζουμε λογαριθμική συμπίεση. Τέλος, αφού εφαρμόσουμε *Discrete Cosine Transform*, παίρνουμε ένα ακουστικό διάνυσμα που αποτελείται από 12 *cepstral* συντελεστές και την ενέργεια του σήματος. Αν προσθέσουμε τις πρώτες και δεύτερες παραγωγούς έχουμε τελικά ένα διάνυσμα μεγέθους 39.

1.3.1 Η διαδικασία κβαντισμού των παραμέτρων του front-end

Το τελικό αποτέλεσμα επομένως, είναι η παραγωγή ενός *MFCC* διανύσματος για κάθε *frame* (δηλαδή για κάθε 10 msec) φωνής. Ο βασικός σκοπός αυτής της εργασίας, είναι ο βέλτιστος διανυσματικός κβαντισμός των παραπάνω διανυσμάτων. Η διαδικασία αυτή περιγράφεται εκτενώς σε επόμενο κεφάλαιο. Εδώ πολύ σύντομα θα αναφέρουμε απλά την κεντρική ιδέα που είναι η εξής :

Το *MFCC* διάνυσμα Y χωρίζεται κατάλληλα σε μικρότερα υποδιανύσματα y_1, y_2, \dots, y_L και κάθε τέτοιο υποδιάνυσμα κβαντίζεται στο πιο κοντινό σε αυτό υποδιάνυσμα \hat{y}_i , από ένα σύνολο από τέτοια κατάλληλα επιλεγμένα υποδιανύσματα (το σύνολο αυτό ονομάζεται *codebook*)

Το αποτέλεσμα του κβαντισμού είναι η συμπίεση του σήματος (αφού κάθε υποδιάνυσμα αναπαρίσταται από το δείκτη του υποδιανύσματος στο οποίο κβαντίστηκε) και η επιτάχυνση της αναγνώρισης καθώς όπως θα δούμε και σε επόμενη παράγραφο, δημιουργήσαμε μια νέα κατηγορία ακουστικών μοντέλων που εκμεταλλεύεται την ύπαρξη του κβαντισμού των ακουστικών διανυσμάτων στο *front-end*.



Σχήμα 1.2: Παραγωγή MFCC διανύσματος. Για να γίνει *pattern matching*, η κυματομορφή μετατρέπεται σε μια ακολουθία ακουστικών διανυσμάτων, τα οποία αντιπροσωπεύουν ένα ομαλοποιημένο λογαριθμικό φάσμα που υπολογίζεται για κάθε 10 msec φωνής. Χρησιμοποιώντας μια μη-γραμμική *mel-frequency* κλίμακα και Μετασχηματισμό Διακριτού Συνημιτόνου (*Discrete Cosine Transform - DCT*) βελτιώνουμε την απόδοση. Το πρώτο, έχει σαν αποτέλεσμα την συμπίεση της πληροφορίας στους πρώτους συντελεστές του διανύσματος, ενώ το τελευταίο, έχει σαν αποτέλεσμα την αποσυσχέτιση (*decorrelation*) του σήματος, βελτιώνοντας τις υποθέσεις για στατιστική ανεξαρτησία και επιτρέποντας τη χρήση διαγωνίων πινάκων συνμεταβλητότητας (*Covariance matrices*). Τέλος, για να ενσωματώσουμε δυναμική πληροφορία για το σήμα, προσθέτουμε τις πρώτες και δεύτερες παραγώγους.

1.4 Acoustic Modeling

Τα ακουστικά μοντέλα παρέχουν ένα τρόπο για τον υπολογισμό της πιθανότητας $P(Y|W)$. Μονάδα μοντελοποίησης είναι το φώνημα που αναπαρίσταται από ένα *HMM* (*triphone*¹). Ένα *HMM* έχει μία κατάσταση εισόδου, μία κατάσταση εξόδου και 3 ενδιάμεσες καταστάσεις. Οι καταστάσεις εισόδου-εξόδου χρησιμεύουν στην ένωση πολλών διαδοχικών *HMMs*, τα οποία σχηματίζουν ένα σύνθετο *HMM*, το οποίο μπορεί να αναπαραστήσει μεγαλύτερες μονάδες, όπως λέξεις ή μια ολόκληρη πρόταση. Το *HMM* μπορεί να θεωρηθεί σαν μια γεννήτρια ακολουθίας συμβόλων (στη περίπτωση μας, ακουστικών διανυσμάτων) μοντελοποιώντας παράλληλα μια κρυφή ακολουθία καταστάσεων (αποτελούμενων από τις 3 ενδιάμεσες καταστάσεις). Η πιθανότητα μετάβασης από την κατάσταση i στην κατάσταση j καθορίζεται από την διακριτή πιθανότητα a_{ij} , ενώ η πιθανότητα παραγωγής ενός συμβόλου στη κατάσταση j , καθορίζεται από τη πιθανότητα εξόδου b_j .

Στο Σχήμα 1.3 φαίνεται ένα παράδειγμα της παραπάνω διαδικασίας όπου το μοντέλο μεταβαίνει μέσω της ακολουθίας καταστάσεων $X = 1, 2, 2, 3, 4, 4, 5$, για να παράγει την ακολουθία y_1 έως y_5 . Η συνδυασμένη πιθανότητα μιας ακολουθίας διανυσμάτων Y και μιας ακολουθίας καταστάσεων X , δεδομένου κάποιου μοντέλου M (πιθανότητες μετάβασης και εξόδου), υπολογίζεται σαν το γινόμενο των πιθανοτήτων μετάβασης και των πιθανοτήτων εξόδου. Για την ακολουθία καταστάσεων X του σχήματος 1.3 έχουμε ότι :

$$P(Y, X|M) = a_{12}b_1(y_1)a_{22}b_2(y_2)a_{23}b_3(y_3) \dots \quad (1.2)$$

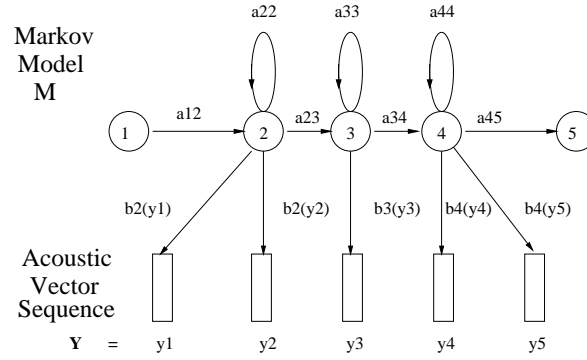
Στην γενική περίπτωση που $X = x(1), x(2), x(3), \dots, x(T)$, η παραπάνω πιθανότητα γίνεται :

$$P(Y, X|M) = a_{x(0)x(1)} \prod_{t=1}^T b_{x(t)}(y_t) a_{x(t)x(t+1)} \quad (1.3)$$

όπου $x(0)$ είναι η κατάσταση εισόδου του μοντέλου και $x(T+1)$ η κατάσταση εξόδου.

Στην πραγματικότητα, εμείς γνωρίζουμε μόνο την ακολουθία Y , ενώ η ακολουθία X μας είναι άγνωστη, για αυτό και μιλάμε για κρυφές Μαρκοβιανές ακολουθίες. Για τον υπολογισμό λοιπόν της $P(Y|M)$ αρκεί να αθροίσουμε την 1.3 για όλες τις πιθανές μεταβάσεις καταστάσεων. Ένας αποτελεσματικός τρόπος για να γίνει αυτό είναι ο *Forward-Backward* αλγόριθμος.

¹Κάθε φώνημα μπορεί να υπάρχει σε διαφορετικά *contexts* (συμφραζόμενα) τα οποία καθορίζονται από τα αντίστοιχα *HMMs* (*triphones*). Για τα 45 φωνήματα της αγγλικής γλώσσας υπάρχουν 45³ *triphones*, από τα οποία μόνο 60.000 χρησιμοποιούνται στη πράξη



Σχήμα 1.3: **HMM μοντέλο.** Ένα HMM μπορεί να θεωρηθεί σαν μια γεννήτρια ακολουθίας συμβόλων (στη περίπτωση μας, ακουστικών διανυσμάτων) μοντελοποιώντας παράλληλα μια κρυφή ακολουθία καταστάσεων. Μεταβαίνει μεταξύ των καταστάσεων βάσει των πιθανοτήτων μετάβασης και κάθε χρονική περίοδο παράγει ένα καινούριο ακουστικό διάνυσμα σύμφωνα με την κατανομή εξόδου της παρούσας κατάστασης.

1.5 Είδη HMMs

Η επιλογή της κατανομής εξόδου, παίζει σημαντικό ρόλο αφού μοντελοποιεί την μεταβλητότητα του φάσματος της φωνής, σε αντίθεση με το μοντέλο μετάβασης καταστάσεων το οποίο έχει να κάνει με την διάρκεια. Ανάλογα λοιπόν με το πως μοντελοποιούνται οι κατανομές εξόδου, έχουμε τρεις κατηγορίες ακουστικών μοντέλων.

1.5.1 Διακριτά HMMs

Τα πρώτα συστήματα αναγνώρισης φωνής χρησιμοποιούσαν διακριτές κατανομές εξόδου σε συνδυασμό με διανυσματικό κβαντιστή (*Vector Quantizer - VQ*). Έτσι, κάθε ακουστικό διάνυσμα αντικαθίσταται από το *index* (δείκτη) του πιο κοντινού σε αυτό διανύσματος από ένα σύνολο τέτοιων διανυσμάτων (*codebook*), ενώ οι κατανομές εξόδου ισοδυναμούν με *look-up tables* που περιέχουν τις πιθανότητες για κάθε πιθανό δείκτη. Για τη κατανομή εξόδου $b_j(y_t)$ στην κατάσταση j , με y_t το ακουστικό διάνυσμα τη χρονική στιγμή t θα ισχύει, δεδομένου της απεικόνισης του y_t στο διάνυσμα y_m του *codebook* :

$$\sum_{m=1}^M b_j(y_m) = 1 \quad (1.4)$$

Το μεγάλο πλεονέκτημα με τα διακριτά μοντέλα είναι ότι υπολογιστικά αυτός ο τρόπος είναι αρκετά αποτελεσματικός από άποψη αναγνώρισης, αφού η πιθανότητα εξόδου δεν χρειάζεται να υπολογιστεί, αλλά αναχτάται μέσω ενός *look-up table*. Αντίθετα το μεγαλύτερο μέρος του χρόνου σπαταλάται στον κβαντισμό, όπου πρέπει να υπολογιστεί η ελάχιστη απόσταση του διανύσματος εισόδου με τα πρότυπα ακουστικά διανύσματα που περιέχονται στο *codebook*. Βλέπουμε λοιπόν ότι ο μηχανισμός *VQ* δρα σαν προ-επεξεργαστής αναγνώρισης (*recognition preprocessor*).

Όμως εκ των πραγμάτων, η ανάλυση του ακουστικού χώρου είναι μικρή (αφού έχουμε πεπερασμένο αριθμό συμβόλων εξόδου) με επιπτώσεις στην ακρίβεια της αναγνώρισης, οι οποίες ενισχύονται και λόγω του κβαντισμού ο οποίος εισαγάγει επιπλέον θόρυβο. Επίσης η χρήση *VQ* με είσοδο 'ολόκληρο' το διάνυσμα ως έχει, απαιτεί πολύ μεγάλα *codebooks* για να καλύψει το ακουστικό χώρο, ακόμα και όταν αυτός παραμένει σχετικά μικρός.

Βέβαια για τις πρώτες εφαρμογές αναγνώρισης φωνής το μοντέλο αυτό λειτουργούσε ικανοποιητικά, αφού εκείνη την εποχή η διαθεσιμότητα σε πόρους όπως μνήμη και υπολογιστική ισχύς ήταν περιορισμένη, ενώ και τα λεξιλόγια ήταν σχετικά μικρά. Καθώς περνούσαν τα χρόνια, με την εισαγωγή όλο και ισχυρότερων υπολογιστικών συστημάτων αλλά κυρίως με τις απαιτήσεις εφαρμογών για πολύ μεγαλύτερο λεξιλόγιο, έγινε φανερό ότι το συγκεκριμένο μοντέλο δεν ήταν πια κατάλληλο.

1.5.2 Συνεχή HMMs - Gaussian Mixture HMMs

Έτσι τα σύγχρονα συστήματα αναγνώρισης χρησιμοποιούν παραμετρικές συνεχείς κατανομές εξόδου που μοντελοποιούν τα ακουστικά διανύσματα απευθείας. Για το σκοπό αυτό συνήθως χρησιμοποιούνται μείγματα γκαουσιανών κατανομών και η πιθανότητα εξόδου (*output probability*) είναι η εξής :

$$b_j(y_t) = \sum_{m=1}^M c_{jm} N(y_t; \mu_{jm}, \Sigma_{jm}) \quad (1.5)$$

όπου c_{jm} είναι το βάρος (*weight*) του μείγματος m στην κατάσταση j και $N(y_t; \mu_{jm}, \Sigma_{jm})$ είναι η πολυδιάστατη (*multivariate*) γκαουσιανή με μέση τιμή μ και συνδιακύμανση Σ για το μείγμα m στην κατάσταση j . Συνήθως «παρόμοιες» καταστάσεις μεταξύ *HMMs* χρησιμοποιούν κοινές γκαουσιανές, έτσι ώστε να μειώνεται ο συνολικός αριθμός τους. Παρ' όλα αυτά, ο χρόνος που χρειάζεται τόσο για την εκπαίδευση των παραμέτρων τους, όσο και κατά την διαδικασία της αναγνώρισης, είναι αρκετά μεγαλύτερος σε σχέση με τους αντίστοιχους χρόνους με διακριτά *HMM* μοντέλα.

1.5.3 Μείγματα διακριτών HMMs από κβαντοποίηση συνεχών HMMs

Όπως είναι φανερό από τα παραπάνω, παρουσιάζεται το φαινόμενο τα μειονεκτήματα της μιας κατηγορίας *HMMs* να αποτελούν πλεονεκτήματα για το άλλο είδος και αντίστροφα. Στο εργαστήριο μας υλοποιήθηκε ένα καινούριο είδος ακουστικών μοντέλων που συγκεντρώνει τα πλεονεκτήματα των 2 μεθόδων και προκύπτει με τον εξής συλλογισμό :

Η ιδέα είναι να εφαρμόσουμε το *VQ* όχι απ'ευθείας στα ακουστικά μοντέλα αλλά στο *front-end*. Με την έξοδο του *VQ* να μην είναι πια ο δείκτης, αλλά το ίδιο το πρότυπο διάνυσμα, μπορούμε να πειραματιστούμε μέχρι να καταλήξουμε σε κάποιο σχήμα κωδικοποίησης που να δίνει υψηλή ακρίβεια αναγνώρισης για το ζητούμενο βαθμό συμπίεσης (με χρήση πάντα των *Gaussian Mixture HMMs*). Εφαρμόζοντας το σχήμα κωδικοποίησης στο οποίο καταλήξαμε, κβαντίζουμε σε δείκτη τώρα πια και παράλληλα διακριτοποιούμε τα μείγματα των γκαουσιανών. Τώρα, περιμένουμε το ίδιο επίπεδο ακρίβειας αναγνώρισης (λόγω της ικανοποιητικής ανάλυσης του ακουστικού χώρου που διατηρήσαμε) καθώς και επιτάχυνση του χρόνου αναγνώρισης (λόγω της αντικατάστασης του υπολογισμού της πιθανότητας εξόδου με ένα απλό *look-up*). Καταλυτικό ρόλο, παίζει το γεγονός ότι το κάθε ακουστικό διάνυσμα 'σπάει' σε κατάλληλα επιλεγμένα μικρότερα διανύσματα (έστω L). Αυτό έχει σαν αποτέλεσμα την ακόμα μεγαλύτερη επιτάχυνση του χρόνου αναγνώρισης, καθώς αντί να χρησιμοποιείται ένα 'τεράστιο' *codebook* κατά τη διάρκεια του κβαντισμού, χρησιμοποιούνται L μικρά *codebooks*.

Βασιζόμενοι λοιπόν στον διανυσματικό κβαντισμό των *MFCC* διανυσμάτων του *front-end* σύμφωνα με τα παραπάνω, η κατανομή εξόδου $b_j(y_t)$, με $y_t = [y_{1t}, y_{2t} \dots y_{Lt}]$ το διάνυσμα που περιέχει τα L υποδιανύσματα είναι :

$$b_j(y_t) = \sum_{m=1}^M c_{jm} \prod_{i=1}^L P_{mi}(vq(y_{it})) \quad (1.6)$$

όπου c_{jm} είναι το βάρος (*weight*) του μείγματος m στη κατάσταση j και $P_{mi}(vq(y_{it}))$ η πιθανότητα του διακριτού συμβόλου $vq(y_{it})$ για το υποδιάνυσμα i του μείγματος m . Με $vq(y_{it})$ αναπαριστούμε το σύμβολο εξόδου του κβαντισμού του υποδιανύσματος i την χρονική στιγμή t , με άλλα λόγια τον αντίστοιχο δείκτη.

Συγκρίνοντας τις εξισώσεις 1.6 και 1.5, βλέπουμε ότι έχουμε αντικατάσταση της γκαουσιανής με ένα γινόμενο πιθανοτήτων, πράγμα που έχει σαν αποτέλεσμα την αντικατάσταση του υπολογισμού των πιθανοτήτων εξόδου των γκαουσιανών με ένα *look-up* και συνεπώς την επιτάχυνση της αναγνώρισης. Για περισσότερες λεπτομέρειες παραπέμπουμε στην διπλωματική εργασία [23].

1.6 Συμπεράσματα

- Τα συστήματα αναγνώρισης φωνής βασίζονται στην ύπαρξη ενός μηχανισμού παραγωγής ‘ακουστικών’ διανυσμάτων (*front-end*), ακουστικών μοντέλων που μοντελοποιούν τα φωνήματα χρησιμοποιώντας κρυφά μαρκοβιανά μοντέλα (*HMMs*) και ενός γλωσσικού μοντέλου. Με χρήση στατιστικών και άλλων μεθόδων, γίνεται τελικά η αποκωδικοποίηση (αναγνώριση) της φωνής.
- Η εισαγωγή του κβαντισμού των παραμέτρων του *front-end* έχει σαν βασικό σκοπό την μέγιστη συμπίεση. Αυτό μας επιτρέπει π.χ. την αναγνώριση σε περίπτωση που το εύρος (*bandwidth*) ενός διαύλου μεταξύ *front-end* και αναγνωριστή είναι περιορισμένο.
- Τα πρώτα συστήματα χρησιμοποιούσαν διακριτά μοντέλα, τελικά όμως επικράτησαν ακουστικά μοντέλα που χρησιμοποιούν γκαουσιανά μείγματα *HMMs*. Για τους σκοπούς αυτής της εργασίας η οποία προτείνει τη χρήση του διανυσματικού κβαντισμού, χρησιμοποιήθηκαν ακουστικά μοντέλα που προκύπτουν με διακριτοποίηση των τελευταίων (*DMHMMs*).
- Πέρα από αυτό όμως, μας δίνεται η δυνατότητα να χρησιμοποιήσουμε το κβαντισμό των παραμέτρων του *front-end* με αναγνωριστή που χρησιμοποιεί τόσο μείγματα συνεχών *HMMs* (στην περίπτωση που ο κβαντισμός υποδιανύσματος δίνει κβαντισμένο υποδιάνυσμα) όσο και μείγματα διακριτών *HMMs* από κβαντοποίηση συνεχών *HMMs* (στην περίπτωση που ο κβαντισμός υποδιανύσματος δίνει τον δείκτη του κβαντισμένου υποδιανύσματος). Στην δεύτερη περίπτωση έχουμε και μεγαλύτερη συμπίεση και ταχύτερη αναγνώριση.

1.7 Παραπομπές

Σε αυτό το κεφάλαιο δόθηκε μια πολύ σύντομη εισαγωγή στο κεφάλαιο που λέγεται αναγνώριση φωνής. Μια πιο πλήρης αλλά και περιεκτική εισαγωγή δίνεται στην εργασία [1], η οποία επίσης περιέχεται και στο βιβλίο [3] για το *HTK* σύστημα. Πιο ειδικά, μια αναλυτική περιγραφή για επεξεργασία φωνής δίνεται στο [2]. Ένα αρκετά αξιόλογο και πλήρες βιβλίο είναι το [4] για το οποίο συνίσταται και το [5]. Στη [4] δίνεται μια ολοκληρωμένη εικόνα του τομέα αναγνώρισης φωνής, από τεχνικές για παραγωγή ακουστικών διανυσμάτων και σύγκρισης προτύπων, μέχρι την εκπαίδευση και προσαρμογή μοντέλων, αλλά και πρακτικές λύσεις για την αποτελεσματικότητα ενός συστήματος αναγνώρισης φωνής.

Κεφάλαιο 2

Κωδικοποίηση φωνής

2.1 Εισαγωγή

Όπως είδαμε στην Εισαγωγή, σκοπός αυτής της εργασίας είναι η εύρεση τρόπου κωδικοποίησης φωνής σε επίπεδα που θα καθιστούν την ανάπτυξη εφαρμογών αναγνώρισης φωνής πιο προσιτή από άποψη κόστους, αφού αυτό είναι συνυφασμένο με τις απαιτήσεις σε *bandwidth*. Αυτό θα επιτρέψει την χρήση του μοντέλου πελάτη - εξυπηρετητή για εφαρμογές σε περιβάλλον όπως αυτό του διαδικτύου ή των ασύρματων δικτύων, όπου ο πελάτης είναι υπεύθυνος για τη κωδικοποίηση της φωνής.

Στο Κεφάλαιο 1 είδαμε ότι το *front-end* παράγει τις παραμέτρους της φωνής με μορφή ακουστικών διανυσμάτων, τα οποία χρησιμοποιούνται από τα ακουστικά μοντέλα για να βρεθεί η βέλτιστη ακολουθία λέξεων από τον αναγνωριστή. Αναφέρθηκε ότι η χρήση διανυσματικού κβαντισμού επιτυγχάνεται με το κβαντισμό των διανυσμάτων εισόδου του *front-end* στα πρότυπα ακουστικά διανύσματα αναφοράς που περιέχονται στο *codebook*. Ειπώθηκε επίσης, ότι η χρήση του διανυσματικού κβαντισμού επιτρέπει τη χρησιμοποίηση ακουστικών μοντέλων που προκύπτουν από διακριτοποίηση μοντέλων με μείγματα συνεχών κατανομών, τα οποία δημιουργήθηκαν ειδικά για το σκοπό αυτής της εργασίας. Ο συνδυασμός του διανυσματικού κβαντισμού με χρήση των παραπάνω μοντέλων, έχει σαν αποτέλεσμα τη συμπίεση του σήματος της φωνής, αλλά και την επιτάχυνση της αναγνώρισης.

Στο πρώτο μέρος αυτού του κεφαλαίου δίνεται κάποιο θεωρητικό υπόβαθρο, παρουσιάζοντας τεχνικές, όπως την κωδικοποίηση πηγής, τον βαθμωτό και διανυσματικό κβαντισμό και τέλος τις τεχνικές κωδικοποίησης φωνής. Στο δεύτερο μέρος του κεφαλαίου, περιγράφεται αναλυτικά το σχήμα κωδικοποίησης που εφαρμόσαμε το οποίο συνδυάζει τεχνικές κωδικοποίησης φωνής και

κωδικοποίησης πηγής. Αναλύονται οι βασικές έννοιες, οι ορολογίες και οι αλγόριθμοι που χρησιμοποιήθηκαν και περιγράφονται οι διαδικασίες εκπαίδευσης για τη δημιουργία των *codebooks* αλλά και του διανυσματικού κβαντισμού με τη χρήση τους.

Πρώτα όμως πρέπει να γίνει ξεκάθαρο, ότι σε αντίθεση με τις συνήθεις τεχνικές κωδικοποίησης που σαν σκοπό έχουν την πιστή αναπαραγωγή του σήματος της φωνής, η κωδικοποίηση που εφαρμόζουμε, υπηρετεί ένα διαφορετικό σκοπό : Την μέγιστη συμπίεση των παραμέτρων της φωνής με σκοπό όχι την πιστή αναπαραγωγή του σήματος, αλλά την καλλίτερη δυνατή αναγνώριση.

2.2 Πηγές πληροφορίας και κωδικοποίηση

Τα συστήματα επικοινωνιών σχεδιάζονται για να μεταδίδουν πληροφορία. Σε όλα αυτά τα συστήματα υπάρχει μια πηγή που παράγει την πληροφορία και σκοπός του συστήματος είναι να μεταδώσει την έξοδο της πηγής στον προορισμό της. Όλοι έχουμε εμπειρία από αυτό το μοντέλο : εκπομπές ραδιοφωνικού σήματος, τηλεόρασης, μετάδοση φωνής κ.τ.λ. Προκειμένου να μπορούμε να κάνουμε ανάλυση απόδοσης ενός τέτοιου μοντέλου όμως, χρειάζονται τα κατάλληλα εργαλεία : ποσοτικές μετρικές και κατάλληλα μαθηματικά μοντέλα. Στην ανάπτυξη των τελευταίων βοήθησαν καταλυτικά οι *Hartley, Nyquist & Shannon*.

Η πηγή πληροφορίας παράγει έξοδο που ο προορισμός της δεν την γνωρίζει από πριν - η πληροφορία έχει να κάνει με τι γνώση για κάτι. Η έξοδος της πηγής είναι μια μη-προβλέψιμη, χρονικά μεταβαλλόμενη διαδικασία, άρα μπορεί να μοντελοποιηθεί σαν μια τυχαία(στοχαστική) διαδικασία. Το χαρακτηριστικό των πηγών αυτών, είναι ότι μπορούν να μοντελοποιηθούν σαν *bandlimited* διαδικασίες. Για παράδειγμα, το σήμα της φωνής έχει σχεδόν όλη την ισχύ του στην μπάντα 300-3400Hz. Εκμεταλλευόμενοι αυτό το γεγονός, μπορούμε να κάνουμε δειγματοληψία σε συχνότητα *Nyquist* και πλέον θα έχουμε να κάνουμε με διακριτού χρόνου στοχαστικές διαδικασίες.

2.2.1 Μοντέλο πληροφορίας

Ένα απλό μοντέλο πληροφορίας είναι η πηγή να μοντελοποιείται από μια διακριτού χρόνου στοχαστική διαδικασία $\{X_i\}_{i=-\infty}^{\infty}$. Το αλφάβητο στο οποίο ορίζονται οι τυχαίες μεταβλητές X_i μπορεί να είναι διακριτές ή συνεχείς (π.χ. δειγματοληπτημένη φωνή), ενώ οι στατιστικές ιδιότητες της διαδικασίας καθορίζονται από την ίδια τη πηγή. Για τη φωνή, ένα ικανοποιητικό τέτοιο μοντέλο,

είναι μια διακριτού χρόνου και πλάτους τυχαία διαδικασία, όπου όλα τα X_i παράγονται ανεξάρτητα (διαδικασία χωρίς μνήμη-*memoryless process*) και με την ίδια κατανομή.

Ας υποθέσουμε ότι η X παίρνει τιμές από ένα σύνολο $A = \{a_1, a_2, \dots, a_M\}$ με πιθανότητες $p_i = p(X = a_i)$, $i = 1, 2, \dots, M$. Μια παρατήρηση για την ‘ποιοτική’ μέτρηση της πληροφορίας, είναι ότι αυτή πρέπει να μειώνεται όσο αυξάνει η πιθανότητα παρατήρησης του συμβόλου εξόδου. Για παράδειγμα, η είδηση ότι μέσα στον Ιούλιο έχουμε βροχή περιέχει περισσότερη πληροφορία από την είδηση ότι έχουμε ηλιοφάνεια. Μια άλλη παρατήρηση, είναι ότι μια μικρή αλλαγή στη πιθανότητα, έχει μικρή επίπτωση στο ποσό της πληροφορίας. Με άλλα λόγια, μια μετρική πληροφορίας θα πρέπει να είναι μια συνεχής και μειούμενη συνάρτηση της πιθανότητας της εξόδου της πηγής.

Μια τέτοια ‘συμπεριφορά’ παρέχεται από την : $-\log(p_i)$ που ονομάζεται *self-information*. Η εντροπία μιας πηγής αποτελεί μια καλή μετρική σύμφωνα με τις παραπάνω παρατηρήσεις και ορίζεται από τη σχέση : $H(X) = -\sum_{i=1}^N p_i \log p_i$. Η εντροπία της πηγής, μας δίνει ένα αρκετά ακριβές όριο του ρυθμού στον οποίο μπορεί να συμπιεστεί η έξοδος της πηγής, για ικανή ανακατασκευή του σήματος χωρίς απώλειες. Αυτό αποδεικνύεται από το θεώρημα κωδικοποίησης πηγής του *Shannon*, σύμφωνα με το οποίο για να έχουμε ικανή ανακατασκευή του σήματος χωρίς απώλειες, θα πρέπει να ισχύει : $R > H$, με R το ρυθμό μετάδοσης (συμπίεση).

Τεχνικές που ‘σέβονται’ το παραπάνω κριτήριο, δηλαδή κωδικοποίηση χωρίς απώλειες, (ονομάζονται *lossless coding* ή και *entropy coding*) είναι η κωδικοποίηση *Huffman* και *Ziv-Lempel*. Συχνά όμως, αποφασίζουμε ότι θέλουμε να συμπίεσουμε την έξοδο της πηγής περισσότερο, με κόστος κάποια μικρή απώλεια πληροφορίας (παραμόρφωση-*distortion*). Σε αυτή τη περίπτωση, λέμε ότι έχουμε κβαντισμό (*quantization*) ή κωδικοποίηση με απώλειες (*lossy coding*). Στην επόμενη παράγραφο αναλύονται δυο τέτοιες τεχνικές.

2.3 Τεχνικές κβαντισμού

Ένα σύστημα κβαντισμού απεικονίζει μια είσοδο σε μια έξοδο, χρησιμοποιώντας κάποιο κατάλληλο μηχανισμό απεικόνισης. Για παράδειγμα, απεικονίζοντας ένα πραγματικό αριθμό σε μια *float* μεταβλητή, έχουμε ήδη κβαντίσει τον πραγματικό αριθμό σε μια από τις διαθέσιμες τιμές (στάθμες) και έχουμε ήδη εισαγάγει θόρυβο. Απλά, επειδή ο αριθμός διαθέσιμων τέτοιων τιμών είναι αρκετά μεγάλος, ο θόρυβος είναι πολύ μικρός και δεχόμαστε ότι δεν έχουμε πρακτικά απώλεια αναπαράστασης κατά την απεικόνιση αυτή. Το ότι με χρήση π.χ. 32 μόνο *bits* πληροφορίας, μπορούμε να αναπαραστήσουμε ένα πραγ-

ματικό αριθμό, σημαίνει ότι έχουμε σημαντικά οφέλη από άποψη απαιτούμενης πληροφορίας(συμπύεση). Συνοψίζοντας, τα δυο σημαντικά χαρακτηριστικά που εισαγάγει ο κβαντισμός, είναι η συμπύεση από τη μια αλλά και η εισαγωγή θορύβου από την άλλη. Στο σχεδιασμό ενός συστήματος κβαντισμού, το παραπάνω *trade-off* παίζει καθοριστικό ρόλο.

Το ερώτημα που τίθεται από το παραπάνω *trade-off* είναι το εξής : ποιός είναι ο ελάχιστος ρυθμός μετάδοσης για να μπορέσουμε να αναπαριστήσουμε την πηγή πληροφορίας, δεχόμενοι ένα συγκεκριμένο επίπεδο απώλειας. Προκειμένου να απαντήσουμε στο παραπάνω ερώτημα, μπορούμε να χρησιμοποιήσουμε τη θεωρία ρυθμού-απώλειας (*rate-distortion theory*) και την αντίστοιχη συνάρτηση που παρέχει (*rate-distortion function*). Αυτή είναι αρκετά χρήσιμη στον σχεδιασμό συστημάτων κβαντισμού σε θεωρητικό όμως κυρίως επίπεδο. Και αυτό, γιατί εξαρτάται από τον ορισμό της απώλειας, αλλά και υποθέσεις για την κατανομή της πηγής πληροφορίας. Για παράδειγμα, εάν θεωρήσουμε μια πηγή με γκαουσιανή κατανομή μηδενικού μέσου και μεταβλητότητας σ^2 , αποδεικνύεται ότι η συνάρτηση ρυθμού-απώλειας, δίδεται από τη σχέση $\frac{1}{2} \log \frac{\sigma^2}{D}$ (με απώλεια D τετραγωνικού σφάλματος). Στην πράξη, η υπόθεση για την κατανομή μιας πηγής είναι αρκετά απλοποιημένη, αλλά επίσης δύσκολη είναι και η επιλογή μιας κατάλληλης μετρικής.

Στο παραπάνω παράδειγμα όπου έχουμε απεικόνιση ενός μόνο συμβόλου εξόδου, μιλάμε για βαθμωτό κβαντισμό (*scalar quantization*). Στην περίπτωση που η μονάδα απεικόνισης είναι ένα σύνολο(μπλοκ) συμβόλων εξόδου, έχουμε διανυσματικό κβαντισμό (*vector quantization*). Οι δυο αυτές μεθοδολογίες έχουν μελετηθεί σε βάθος εδώ και χρόνια, και έχουν προταθεί διάφορες τεχνικές για την αποτελεσματικότερη χρήση τους. Εδώ θα περιγράψουμε πολύ σύντομα τις δυο αυτές μεθοδολογίες και θα δούμε πως χρησιμοποιούνται στις αμέσως επόμενες παραγράφους.

2.4 Βαθμωτός κβαντισμός

Στο βαθμωτό κβαντισμό, έχουμε απεικόνιση του συμβόλου εισόδου σε μια μόνο, από ένα σύνολο προκαθορισμένων τιμών που ονομάζονται στάθμες και μπορούμε να τις φανταστούμε σαν προεπιλεγμένα σημεία στον άξονα των πραγματικών αριθμών. Θεωρούμε ότι έχουμε M περιοχές που χωρίζουν τον άξονα των πραγματικών αριθμών. Κάθε τέτοια περιοχή $R_k, 1 \leq k \leq M$ αντιπροσωπεύεται από μια στάθμη \hat{x}_k . Αν η τιμή εισόδου x ανήκει στη περιοχή R_k , τότε απεικονίζουμε την είσοδο στην στάθμη της περιοχής. Αυτή η απεικόνιση (συνάρτηση κβαντισμού Q) ορίζεται ως : $Q(x) = \hat{x}_k \quad \forall x \in R_k$. Η συνάρτηση κβαντισμού είναι μη-γραμμική και μη-αναστρέψιμη, αφού όλα τα σημεία στην

R_k απεικονίζονται στην τιμή \hat{x}_k . Επειδή η απεικόνιση είναι μη-αναστρέψιμη, κάποια πληροφορία χάνεται για πάντα.

Οι στάθμες αναπαρίστανται ως δυαδική ακολουθία. Αφού έχουμε συνολικά M περιοχές ο ρυθμός πληροφορίας θα είναι $\log M$. Υπάρχουν 2 κατηγορίες βαθμωτού κβαντισμού : ομοιόμορφος και μη-ομοιόμορφος κβαντισμός. Στη πρώτη κατηγορία οι περιοχές έχουν ίδιο μήκος ενώ στη δεύτερη όχι. Η μη ύπαρξη αυτού του περιορισμού, επιτρέπει την καλλίτερη επιλογή σταθμών στη δεύτερη περίπτωση, με αποτέλεσμα η δεύτερη κατηγορία να είναι πιο αποτελεσματική σε σχέση με τη πρώτη, για ίδιο αριθμό σταθμών. Τα κριτήρια βέλτιστης απόδοσης ενός κβαντιστή είναι γνωστά και ως συνθήκες *Lloyd-Max* και είναι τα εξής :

- Τα όρια των περιοχών είναι τα ισαπέχοντα σημεία από 2 γειτονικές στάθμες . Αυτή η συνθήκη ονομάζεται κανόνας *nearest-neighbour*.
- Οι στάθμες είναι τα *centroids* (*conditional expected value*) των περιοχών.

Επειδή οι κανόνες αυτοί δεν παρέχουν αναλυτικές λύσεις, η διαδικασία που ακολουθείται είναι το ξεκίνημα με κάποιο σύνολο περιοχών και η εύρεση των *centroids* με βάση το 2ο κριτήριο. Για τα νέα αυτά *centroids*, δημιουργούνται νέες περιοχές με βάση το 1ο κριτήριο. Η διαδικασία συνεχίζεται, εναλλάσσοντας τα κριτήρια, μέχρι η απώλεια να φτάσει κάτω από τα επιθυμητά επίπεδα. Ο επαναληπτικός αυτός αλγόριθμος, αποδεικνύεται ότι συγκλίνει σε κάποιο τοπικό ελάχιστο και οι αρχικές συνθήκες (επιλογή περιοχών), είναι αυτές που καθορίζουν το που θα συγκλίνει ο αλγόριθμος.

2.5 Διανυσματικός κβαντισμός - VQ

Ο διανυσματικός κβαντισμός (VQ από εδώ και πέρα) αποτελεί μια γενίκευση του βαθμωτού (ή και αντίστροφα, ο βαθμωτός είναι ειδική περίπτωση του διανυσματικού). Πηγαίνοντας από την μια διάσταση στη πολυδιάστατη περίπτωση, νέες ιδέες, έννοιες και τεχνικές εμφανίζονται, που επιτρέπουν την δημιουργία αποτελεσματικών και επιτηδευμένων εφαρμογών συμπίεσης. Ένα διάνυσμα μπορεί να περιγράψει οποιοδήποτε πρότυπο (*pattern*) και από αυτή την άποψη ο VQ μπορεί να θεωρηθεί σαν μια μορφή αναγνώρισης προτύπων, όπου το διάνυσμα εισόδου 'προσεγγίζεται' με ένα προκαθορισμένο από ένα σύνολο τέτοιων προτύπων διανυσμάτων. Τα πρότυπα αυτά διανύσματα ονομάζονται *codewords* ή και *centroids* και το σύνολο αυτών *codebook*.

Μια μαθηματική αντιστοιχία των παραπάνω εκφράζεται με τα εξής : Θεωρούμε ότι έχουμε M περιοχές στον N -διάστατο χώρο $R_k, 1 \leq k \leq M$ και

μονάδα απεικόνισης είναι μπλοκ N συμβόλων εξόδου (ή αλλιώς διάνυσμα μήκους N). Αν $\mathbf{x} \in \mathbf{R}^N$ και $\mathbf{x} \in R_k$ τότε αυτό προσεγγίζεται (κβαντίζεται) σύμφωνα με την $Q(\mathbf{x}) = \hat{\mathbf{x}}_k$. Έχουμε δηλαδή μια απεικόνιση: $Q: \mathbf{R}^N \rightarrow C$, με $C = (\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_k, \dots, \hat{\mathbf{x}}_M)$ και $\hat{\mathbf{x}}_k \in \mathbf{R}^N$. Αφού έχουμε M τέτοιες τιμές κβαντισμού (codebook C), απαιτείται ρυθμός μετάδοσης $R = \frac{\log M}{N} \text{bits/σύμβολο}$ εξόδου.

2.5.1 Ανάλυση απόδοσης και πλεονεκτήματα VQ

Σύμφωνα με τις θεμελιώδεις αρχές της θεωρίας 'ρυθμού-απώλειας', αποδεικνύεται ότι με χρήση VQ , μπορεί να επιτευχθεί πάντα καλλίτερη απόδοση, από τη περίπτωση χρήσης βαθμωτού κβαντισμού. Στην πραγματικότητα, για πολύ καιρό η θεωρία αυτή είχε περιορισμένη απήχηση, γιατί από τη μια, η θεωρία του *Shannon* δεν παρείχε μεθοδολογίες σχεδιασμού VQ συστημάτων και από την άλλη οι βαθμωτοί κβαντιστές παρείχαν σχετικά ικανοποιητική απόδοση. Στο τέλος της δεκαετίας του '70, με την εισαγωγή του απλού επαναληπτικού αλγορίθμου για σχεδιασμό βαθμωτών κβαντιστών που βασίζεται στα κριτήρια βελτιστοποίησης (όπως είδαμε στην προηγούμενη παράγραφο - *Lloyd*), έγινε αντιληπτό, ότι αυτός είναι εύκολα προσαρμόσιμος στην περίπτωση του VQ , φέρνοντας έτσι στο προσκήνιο το ενδιαφέρον για VQ μεθοδολογίες. Από τότε μέχρι σήμερα υπάρχει έντονο ερευνητικό ενδιαφέρον και έχουν αναπτυχθεί πολλές αποτελεσματικές τεχνικές VQ , βασιζόμενες σε διάφορες παραλλαγές του επαναληπτικού αλγορίθμου, με εφαρμογές σε πολλά είδη πηγών πληροφορίας, όπως ομιλίας και εικόνας.

Είδαμε ότι η συνάρτηση ρυθμού απώλειας εξαρτάται από τη γνώση της κατανομής των συμβόλων εξόδου της πηγής και την επιλογή μιας μετρικής για τον υπολογισμό της απώλειας. Επειδή η κατανομή εξαρτάται από τα χαρακτηριστικά της πηγής, χρησιμοποιούμε για την εκτίμησή της, ένα μεγάλο αριθμό διανυσμάτων εκπαίδευσης (*train set*) που για την περίπτωση που έχουμε πηγή ομιλίας δεν είναι άλλα από τα ακουστικά διανύσματα που παράγει το *front-end*. Οι πιο συχνές επιλογές για την μετρική απώλειας στη περίπτωση πηγών φωνής, είναι η απώλεια τετραγωνικού σφάλματος με ή χωρίς χρήση βαρών και η απώλεια *Itakura-Saito*. Η τελευταία είναι ιδιαίτερα χρήσιμη για ακουστικά διανύσματα που παράγονται από τεχνικές κωδικοποίησης βασισμένες στο μοντέλο παραγωγής φωνής, που θα δούμε σε επόμενη παράγραφο. Στην τεχνική που αναπτύξαμε χρησιμοποιούμε την απώλεια τετραγωνικού σφάλματος με βάρη και συγκεκριμένα την *Mahalanobis* απώλεια που ορίζεται ως εξής :

$$d(\mathbf{x}, \hat{\mathbf{x}}) = (\mathbf{x} - \hat{\mathbf{x}})^T W (\mathbf{x} - \hat{\mathbf{x}}) \quad (2.1)$$

με πίνακα βαρών τον αντίστροφο πίνακα συνδιακύμανσης $W = E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} -$

$\bar{\mathbf{x}})^T]^{-1}$ και $\bar{\mathbf{x}} = E[\mathbf{x}]$.

Ο επαναληπτικός αλγόριθμος που χρησιμοποιούμε, περιγράφεται στη παράγραφο 2.9.2. Στόχος του αλγορίθμου είναι η ελαχιστοποίηση της μέσης απώλειας, δηλαδή του όρου $Ed(\mathbf{x}, \hat{\mathbf{x}})$ που στην πράξη μπορεί να αντικατασταθεί με τον επί μακρόν δειγματοληπτικό μέσο (*long term sample average*) : $\lim_{n \rightarrow \infty} \sum_{i=0}^{n-1} d(\mathbf{x}_i, \hat{\mathbf{x}}_i)$, υπό την προϋπόθεση ότι η ακολουθία είναι εργοδική (*ergodic*) και στάσιμη (*stationary*). Ακόμα όμως και να μην ισχύει αυτή η συνθήκη, στην πράξη είναι αρκετό η πηγή να είναι ασυμπτωτικά στάσιμη στο μέσο (*mean stationary*), κάτι που όντως ισχύει στη περίπτωση της φωνής (*long and short term stationary*).

Είδαμε στην αρχή της παραγράφου, ότι ο ρυθμός μετάδοσης για VQ, είναι $R = \frac{\log M}{N} \text{bits/σύμβολο εξόδου}$. Αν συγκρίνουμε αυτό το ρυθμό με το ρυθμό στη περίπτωση του βαθμωτού κβαντισμού ($\log M$) βλέπουμε ότι ο ίδιος αριθμός *bits* αυτή τη φορά μοιράζεται μεταξύ πολλών συμβόλων εξόδου. Αυτό βέβαια δεν σημαίνει ότι απαιτείται ίδια ποσότητα πληροφορίας στις 2 περιπτώσεις, αλλά ότι στην 2η περίπτωση κατανέμεται σε περισσότερα σύμβολα εξόδου, με μικρότερη πιθανότητα σπατάλης *bits*. Στην περίπτωση αυτή, ένα σύμβολο εξόδου μπορεί να απαιτεί κλασματικό (*fractional*) ποσό πληροφορίας όπως π.χ. 0.38 *bits* και αυτό δεν είναι το μοναδικό πλεονέκτημα του VQ. Αποδεικνύεται ότι οι τεχνικές VQ δίνουν καλλίτερη απόδοση, καθώς εκμεταλλεύονται καλλίτερα την ύπαρξη γραμμικής (*correlation*) και μη γραμμικής εξάρτησης μεταξύ των συμβόλων εξόδου, αλλά αποδίδουν καλλίτερα ακόμα και στη περίπτωση που αυτή δεν υπάρχει. Τέλος εκμεταλλεύονται καλλίτερα την κατανομή της πηγής και επιτρέπουν τον σχηματισμό περιοχών με σχήμα χωρίς περιορισμούς στον N -διάστατο χώρο.

2.5.2 Είδη VQ, παράμετροι σχεδιασμού και τεχνικές

Οι τεχνικές VQ χωρίζονται σε 2 κατηγορίες : κβαντιστές με μνήμη και χωρίς μνήμη. Οι δεύτεροι αποτελούν μια εναλλακτική λύση σε περίπτωση που οι πρώτοι έχουν μεγάλες απαιτήσεις σε πόρους, όπως μνήμη και υπολογιστική ισχύς (μεγάλα διανύσματα και μεγάλα *codebooks*) και εκμεταλλεύονται το γεγονός ύπαρξης συσχέτισης μεταξύ διαδοχικών διανυσμάτων. Όμως με προσεκτικό σχεδιασμό των πρώτων, μπορούμε να έχουμε αρκετά αποδοτικό κβαντισμό, καθιστώντας τη χρήση των δεύτερων μη υποχρεωτική.

Ο αριθμός τεχνικών VQ είναι αρκετά μεγάλος για κβαντιστές χωρίς μνήμη και ακόμα μεγαλύτερος για αυτούς με μνήμη. Επίσης υπάρχουν παράμετροι που μπορούν να επηρεάσουν τον τρόπο σχεδιασμού για αυτές τις τεχνικές. Με αυτή τη πληθώρα διαφορετικών προσεγγίσεων, κανείς πρέπει να διαλέξει προσεκτικά την κατάλληλη, προκειμένου να έχει τα μέγιστα ωφέλη. Στη συνέχεια,

θα επικεντρωθούμε σε κάποιες τέτοιες παραμέτρους και τεχνικές που επιλέχθηκαν από την κωδικοποίηση που κάναμε για αυτή την εργασία, με δεδομένη την επιλογή κβαντιστή χωρίς μνήμη.

Επιλογή αρχικού codebook

Μια τέτοια παράμετρος, είναι το μέγεθος του αρχικού *codebook* (δηλαδή ο αριθμός *centroids* σε αυτό) που θα χρησιμοποιήσουμε στον επαναληπτικό αλγόριθμο. Δυο προσεγγίσεις υπάρχουν : στην πρώτη, το μέγεθος είναι όσο και το τελικό (το οποίο καθορίζουμε με τον αριθμό των *bits* που έχουμε επιλέξει να διαθέσουμε) και τα αρχικά *centroids* επιλέγονται π.χ. τυχαία από το σετ εκπαίδευσης. Στη δεύτερη το μέγεθος είναι μικρό (π.χ. ένα : ο μέσος όρος όλων των υποδιανυσμάτων στο σετ εκπαίδευσης) και διπλασιάζεται από επανάληψη σε επανάληψη του αλγορίθμου, μέσα από μια διαδικασία διαχωρισμού (*splitting*) των παρόντων *centroids*, μέχρι την ικανοποίηση κάποιου κριτηρίου (π.χ. επίπεδο απώλειας, επίδοση αναγνώρισης). Η απόφαση για την παράμετρο αυτή, εξαρτάται από την παραλλαγή του αλγορίθμου που χρησιμοποιείται και εμείς επιλέξαμε την δεύτερη.

Product VQ

Μια άλλη επιλογή που έχουμε, είναι να χωρίσουμε το αρχικό διάνυσμα σε μικρότερα και να εφαρμόσουμε VQ για καθένα από αυτά. Αυτό προκύπτει, γιατί πολλές φορές μπορούμε να διακρίνουμε τέτοια μικρότερα γκρουπ μέσα στο διάνυσμα, βάσει κάποιων κοινών χαρακτηριστικών. Στη περίπτωση της φωνής, η μέθοδος αυτή επιλέγεται κυρίως για ακουστικά διανύσματα που παράγονται από *model-based* τεχνικές (βλέπε 2.8). Για *front-ends* όπως αυτά που είδαμε στο Κεφάλαιο 1, ο τρόπος αυτός χρησιμοποιείται σχετικά σπάνια, με πιο κοινό χαρακτηριστικό διαχωρισμού, την ποικιλία τιμών (*dynamic range*) των στοιχείων του ακουστικού διανύσματος. Σύμφωνα με το τελευταίο, ο πιο εμφανής διαχωρισμός είναι ανάμεσα στην ενέργεια και τις υπόλοιπες 12 *cepstral* παραμέτρους.

Το βασικό πλεονέκτημα του διαχωρισμού σε μικρότερα διανύσματα, είναι η μείωση του κόστους αποθήκευσης των *codebooks* και του υπολογισμού που απαιτείται για τον υπολογισμό της απώλειας, αφού τώρα το διάνυσμα είναι μικρότερο. Το μειονέκτημα είναι ότι αυτός ο διαχωρισμός εμποδίζει τον κβαντιστή να εκμεταλλευτεί π.χ. την συσχέτιση που υπάρχει μεταξύ στοιχείων, που τώρα βρίσκονται σε διαφορετικά διανύσματα (σύμφωνα με τη θεωρία, όσο μεγαλύτερο το διάνυσμα, τόσο μεγαλύτερα τα ωφέλη από άποψη συμπίεσης). Φαίνεται ότι στη πράξη, προηγούμενες σχεδιάσεις VQ απέφευγαν τη χρήση

αυτής της μεθόδου, παρασυρόμενοι από το παραπάνω μειονέκτημα, ή από το γεγονός ότι ο διαμοιρασμός των διανυσμάτων γινόταν με λάθος κριτήρια, δίνοντας μη ικανοποιητικά αποτελέσματα έως τώρα.

Εμείς επιμείναμε σε αυτή τη μέθοδο, και καταφέραμε να την εκμεταλλευτούμε όσο περισσότερο γινόταν. Μάλιστα τα τελικά εντυπωσιακά αποτελέσματα της κωδικοποίησης που χρησιμοποιήσαμε, δείχνουν ότι αυτή η μέθοδος μπορεί να βελτιώσει πολύ ένα σχήμα κωδικοποίησης, με το μειονέκτημά της να επηρεάζει ελάχιστα, σε σχέση με τα ωφέλη που μπορεί να δημιουργήσει, αρκεί να χρησιμοποιηθεί κατάλληλα. Σχεδιάζοντας ένα κβαντιστή, πρέπει κανείς να σταθμίσει σωστά τα συν και τα πλην. Για παράδειγμα, η χρήση ολόκληρου του διανύσματος μπορεί θεωρητικά να είναι η καλύτερη επιλογή, πρακτικά όμως δημιουργεί πολύ μεγάλες απαιτήσεις σε μνήμη και υπολογιστική ισχύ. Προκειμένου να δούμε τα ωφέλη της μεθόδου, θα πρέπει να επιλεχθούν κατάλληλα τα μικρότερα διανύσματα (υποδιανύσματα) αλλά και οι συντελεστές που τα αποτελούν. Στο παράδειγμα που δώσαμε παραπάνω (12 συντελεστές, ενέργεια) τα ωφέλη είναι πολύ μικρά, ίσως γιατί το κριτήριο που επιλέχθηκε δεν είναι τόσο μεγάλης σημασίας.

Ενας καλλίτερος διαχωρισμός είναι αυτός που θα μοιράσει το διάνυσμα των 13 στοιχείων σε ένα μεγαλύτερο αριθμό υποδιανυσμάτων (π.χ. 5) καθένα από τα οποία απαιτεί περίπου ίδιο αριθμό *bits*. Ας υποθέσουμε ότι οι δυο μέθοδοι απαιτούν τον ίδιο αριθμό *bits* (αφού αντιπροσωπεύουν το ίδιο διάνυσμα με 2 διαφορετικές μορφές), έστω 20 και ότι στη δεύτερη περίπτωση τα 5 διανύσματα απαιτούν 5, 5, 4, 4 και 2 *bits* αντίστοιχα. Χρησιμοποιώντας την πρώτη προσέγγιση απαιτούνται 2^{20} , δηλαδή πάνω από ένα εκατομμύριο *centroids* στο μοναδικό *codebook*, ενώ στη δεύτερη 5 *codebooks* με $2^5 + 2^5 + 2^4 + 2^4 + 2^2$, δηλαδή 32, 32, 16, 16 και 4 *centroids* αντίστοιχα (σύνολο 100). Τα ωφέλη είναι απίστευτα, από άποψη χρόνου κβαντισμού και αποθήκευσης, ακόμα και αν έχουμε μικρή απώλεια ως προς τη βέλτιστη θεωρητική συμπίεση (δηλαδή για την ίδια συμπίεση η πρώτη μέθοδος να απαιτούσε τελικά 18 ή ακόμα και 15 *bits*, κάτι που δεν αλλάζει όμως την εικόνα).

Τεχνικές ψαξίματος

Τέλος, η τελευταία επιλογή που θα εξετάσουμε, είναι ο τρόπος που είναι δομημένα τα *centroids* μέσα στα *codebooks*, ο οποίος καθορίζει και τον αριθμό των συγκρίσεων που απαιτούνται για να βρεθεί το ‘πλησιέστερο’ στο διάνυσμα εισόδου *centroid* (*nearest-neighbour search*). Ο πιο απλός τρόπος, είναι να χρησιμοποιηθεί πλήρης ψάξιμο, δηλαδή όλα τα *centroids*. Αν και εξαιρετικά απλός, για μεγάλα *codebooks* αυτός ο τρόπος είναι πολύ αργός. Η άλλη επιλογή είναι να δομηθούν τα *centroids* με ένα ιεραρχικό τρόπο (π.χ. δυαδικό δέντρο).

Με αυτό τον τρόπο, για M *centroids* απαιτούνται μόλις $\log M$ συγκρίσεις αντί για M που απαιτούνται στην προηγούμενη περίπτωση. Αν και πολύ γρηγορότερος, αυτός ο τρόπος απαιτεί αλλαγές στον αλγόριθμο εκπαίδευσης, αλλά είναι και υποβέλτιστος σε σχέση με τον πρώτο, αφού τα διανύσματα που χρησιμοποιούνται για την εκπαίδευση των *centroid* σε κάθε επανάληψη, δεν είναι το σύνολο αυτών του σετ εκπαίδευσης, αλλά ένα όλο και μικρότερο γκρουπ αυτών, τα οποία σε προηγούμενη επανάληψη ομαδοποιήθηκαν στην περιοχή του παρόντος *centroid*. Με δεδομένη την χρήση υποδιανυσμάτων, τα *codebooks* είναι αρκετά μικρά, άρα δεν έχει τόσο μεγάλη σημασία η χρήση της δεύτερης μεθόδου αφού επιπλέον είναι και μη βέλτιστη.

2.6 Τεχνικές κωδικοποίησης φωνής

Οι τεχνικές κωδικοποίησης χωρίζονται σε 3 κατηγορίες :

- *Waveform*
- *Model-based*
- *Hybrid*

Σε κάθε μια από τις παραπάνω κατηγορίες συναντάμε τεχνικές που παρέχουν διαφορετικούς ρυθμούς μετάδοσης, ακρίβεια αναπαράστασης της πληροφορίας, απόδοση παρουσία θορύβου και πολυπλοκότητα υλοποίησης. Κάθε κατηγορία εξυπηρετεί λοιπόν διαφορετικές ανάγκες.

2.7 Waveform Coding τεχνικές

Οι τεχνικές αυτής της κατηγορίας, είναι στη πλειοψηφία τους σχετικά απλές. Σκοπός είναι η πιστή αναπαραγωγή του σήματος της φωνής στο δέκτη. Κωδικοποιητές (*coders*) κυματομορφής έχουν σχεδιαστεί τόσο στο πεδίο του χρόνου όσο και στο πεδίο της συχνότητας και είναι αρκετά γρήγοροι λόγω ακριβώς της σχετικά απλής λειτουργίας τους.

2.7.1 Τεχνικές στο πεδίο του χρόνου

Εισάγουν από ελάχιστες έως καθόλου υποθέσεις για το σήμα που κωδικοποιείται και χρησιμοποιούνται σε υψηλούς σχετικά ρυθμούς μετάδοσης. Κάνουν χρήση βαθμωτού κβαντισμού συνήθως και μερικά τέτοια παραδείγματα των τεχνικών αυτών ακολουθούν ευθύς αμέσως :

Pulse Code Modulation (PCM)

Αποτελεί την πιο απλή μέθοδο. Το σήμα δειγματοληπτείται στη συχνότητα *Nyquist* και κάθε δείγμα κβαντίζεται στην κοντινότερη στάθμη και στη συνέχεια μεταδίδεται σαν δυαδικός αριθμός. Όσα περισσότερα *bits* διαθέσουμε, τόσες περισσότερες στάθμες έχουμε, με αποτέλεσμα να έχουμε πιο πιστή αναπαράσταση του σήματος. Το σήμα της φωνής χαρακτηρίζεται από το γεγονός ότι μικρά πλάτη (*amplitudes*) συμβαίνουν πολύ συχνότερα από τα μεγάλα. Μια βελτίωση είναι επομένως, οι στάθμες να απέχουν λιγότερο μεταξύ τους στα μικρά πλάτη και περισσότερο στα μεγάλα. Αυτό οδηγεί στη χρήση μη ομοιόμορφου βαθμωτού κβαντιστή, ο οποίος συμπιέζει το σήμα χρησιμοποιώντας κάποια λογαριθμική συνάρτηση και στη συνέχεια το κβαντίζει χρησιμοποιώντας ομοιόμορφο κβαντιστή. Τέτοιοι λογαριθμικοί συμπιεστές χρησιμοποιούνται για τη μετάδοση φωνής στα τηλεφωνικά δίκτυα και είναι γνωστοί σαν *μ-law & A-law compressors*.

Differential PCM (DPCM)

Μια πρώτη βελτίωση στην *PCM* μέθοδο είναι ο κβαντισμός να γίνεται στην διαφορά μεταξύ 2 διαφορετικών δειγμάτων. Αυτή είναι μια σημαντική βελτίωση, αφού στο σήμα της φωνής, διαδοχικά δείγματα έχουν μεγάλο βαθμό συσχέτισης και επομένως η διαφορά μεταξύ 2 διαφορετικών δειγμάτων είναι μικρή. Μια βασική επέκταση της παραπάνω ιδέας είναι να κάνουμε κάποια πρόβλεψη για το τρέχον δείγμα, και να κβαντίζουμε τη διαφορά μεταξύ του πραγματικού δείγματος και της προβλεπόμενης τιμής του. Έτσι περιμένουμε να μειώσουμε ακόμα περισσότερο το απαιτούμενο *bandwidth*.

Σημείωση : Για την πρόβλεψη του επόμενου δείγματος χρησιμοποιείται ένας γραμμικός προβλέπτης της μορφής : $\hat{x}_n = \sum_{i=1}^p a_i x_{n-i}$, όπου p είναι τα προηγούμενα δείγματα που χρησιμοποιούμε και a_i είναι το βάρος για το αντίστοιχο προηγούμενο δείγμα.

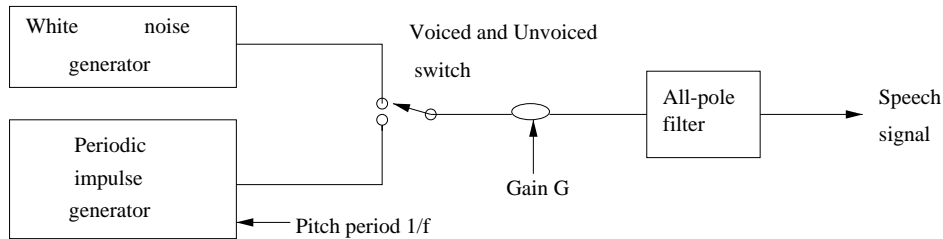
Adaptive DPCM (ADPCM)

Οι παραπάνω τεχνικές βασίζονται στην υπόθεση ότι το σήμα εισόδου είναι στάσιμο (*stationary*), δηλαδή οι στατιστικές του ιδιότητες (π.χ. μεταβλητότητα, αυτοσυσχέτιση) είναι σταθερές με το χρόνο. Στην περίπτωση του σήματος της φωνής οι παραπάνω ιδιότητες μεταβάλλονται αργά με το χρόνο (*quasi-stationary*). Για να εκμεταλλευτούμε και αυτό το χαρακτηριστικό του σήματος της φωνής οι κωδικοποιητές πρέπει να μπορούν να προσαρμοστούν στις χρονικά μεταβαλλόμενες αλλαγές των στατιστικών μεταβολών του σήματος. Έτσι έχουν αναπτυχθεί προσαρμοστικές εκδόσεις του *DPCM*, που

χρησιμοποιούν προσαρμοστικό κβαντιστή ή/και προβλέπτη. Ο κβαντιστής γίνεται προσαρμοστικός αλλάζοντας τις στάθμες κβαντισμού ανάλογα με την ισχύ του σήματος, και ο προβλέπτης αλλάζοντας τους συντελεστές ανάλογα με το φάσμα του σήματος. Τέλος υπάρχουν και άλλες ανάλογες τεχνικές όπως π.χ. *Delta Modulation*.

2.7.2 Τεχνικές στο πεδίο της συχνότητας

Παραδείγματα τεχνικών που χρησιμοποιούνται στο πεδίο της συχνότητας είναι οι : *Sub-band coding (SBC)*, *Adaptive transform coding (ATC)* & *Filterbank Spectrum Analyzer*. Οι τεχνικές αυτές στηρίζονται στην ιδέα του φιλτραρίσματος του σήματος σε διαφορετικές μπάντες στο πεδίο της συχνότητας και τη κωδικοποίηση σε ένα από τα δυο πεδία στη συνέχεια. Μάλιστα, η περιγραφή του *front-end* μηχανισμού που είδαμε στο Κεφάλαιο 1, ανήκει στη τρίτη μέθοδο και στηρίζεται στη χρήση μιας σειράς από φίλτρα σε διαφορετικές μπάντες συχνοτήτων (*filterbanks*). Στη παράγραφο 2 του Κεφαλαίου 3 θα δούμε αναλυτικότερα πως χρησιμοποιείται αυτή η τεχνική για να παράγουμε τελικά τα ακουστικά διανύσματα. Ένα εναλλακτικό τύπο *front-end* μηχανισμού θα δούμε επίσης στην αμέσως επόμενη παράγραφο.



Σχήμα 2.1: Μοντέλο παραγωγής φωνής.

2.8 Model based τεχνικές (Vocoders)

Σε αντίθεση με τις τεχνικές της προηγούμενης κατηγορίας που βασίζονται σε μία δείγμα προς δείγμα ή *frames* ανά *frames* αναπαράσταση της κυματομορφής (πεδίο της χρόνου/συχνότητας), οι τεχνικές αυτής της κατηγορίας βασίζονται στο μοντέλο παραγωγής φωνής. Έτσι χρησιμοποιώντας τη γνώση μας για την διαδικασία παραγωγής φωνής, οι τεχνικές αυτές καταφέρνουν να αναπαραστήσουν το σήμα της φωνής με πολύ μικρότερο *bandwidth* και τέτοιες είναι και οι εφαρμογές στις οποίες χρησιμοποιούνται. Ουσιαστικά γίνεται εξαγωγή

παραμέτρων που αντιστοιχούν στις χρονικά μεταβαλλόμενες παραμέτρους του φωνητικού σωλήνα.

Το μοντέλο φαίνεται στο σχήμα 2.1. Όπως φαίνεται, υπάρχουν 2 δυνατές πηγές. Η πρώτη πηγή αντιστοιχεί στον ήχο χωρίς φωνή και η δεύτερη σε αυτόν με φωνή (*unvoiced & voiced sounds*). Στην 2η πηγή η φωνή παράγεται όταν οι φωνητικές χορδές πάλλονται σε κάποια βασική συχνότητα f_0 . Μια πηγή κάθε χρονική στιγμή διεγείρει το φίλτρο που αναπαριστάει το φωνητικό σωλήνα και έτσι παράγεται το σήμα της φωνής. Η παραγωγή των παραμέτρων του φωνητικού σωλήνα γίνεται με χρήση γραμμικής πρόβλεψης (*linear prediction coding*). Το σύνολο των παραμέτρων που προκύπτουν περιλαμβάνουν τους συντελεστές πρόβλεψης, το κέρδος (*gain*) του σήματος, το είδος εισόδου που έχουμε και τη συχνότητα f_0 , εφόσον η είσοδος είναι *voiced*. Οι παραπάνω παράμετροι μεταδίδονται στον δέκτη όπου χρησιμοποιώντας το ίδιο μοντέλο ανακατασκευάζεται το αρχικό σήμα (για αυτό το λόγο οι τεχνικές αυτές ονομάζονται επίσης *analysis by synthesis* τεχνικές). Για κάθε 20-30 msec που το σήμα της φωνής μπορεί να θεωρηθεί *stationary* υπολογίζονται οι συντελεστές του φίλτρου σύμφωνα με την εξίσωση :

$$X_n = \sum_{i=1}^p a_i X_{n-1} + G w_n \quad (2.2)$$

όπου G είναι το κέρδος, w_n η ακολουθία εισόδου, a_i οι συντελεστές του φίλτρου και p ο αριθμός πόλων του φίλτρου.

Από αυτές τις παραμέτρους και με κατάλληλους μετασχηματισμούς μπορούν να προκύψουν ακουστικά διανύσματα ισοδύναμα με αυτά που είδαμε στο Κεφάλαιο 1, τα οποία βέβαια μπορούν επίσης να χρησιμοποιηθούν για αναγνώριση. Άρα η μέθοδος αυτή αποτελεί ουσιαστικά ένα εναλλακτικό *front-end* μηχανισμό. Αποτελεί φυσικά και μέθοδο κωδικοποίησης φωνής, αφού από το αρχικό σήμα φωνής έχει ήδη προκύψει μια ακολουθία ακουστικών διανυσμάτων.

Βασικό χαρακτηριστικό των *vocoders* είναι οι χαμηλοί ρυθμοί μετάδοσης αλλά και η χαμηλή ποιότητα. Υπάρχουν πολλές υλοποιήσεις *vocoders* με πιο διαδεδομένους όσους στηρίζονται σε τεχνικές γραμμικής πρόβλεψης, σαν αυτή που μόλις περιγράφηκε. Οι υβριδικοί κωδικοποιητές (επίσης *vocoders*), προσπαθούν να χρησιμοποιήσουν μια πιο σύνθετη διέγερση από τους *LPC vocoders*, καταφέροντας έτσι καλή ποιότητα για χαμηλούς-μέσους ρυθμούς μετάδοσης. Υλοποιήσεις *vocoders* (υβριδικών και μη) είναι οι *channel*, *phase*, *cepstral & formant vocoders*, καθώς και οι *REL*P, *CEL*P & *VSEL*P *vocoders*. Μάλιστα η κωδικοποίηση GSM στη κινητή τηλεφωνία δευτέρας γενεάς, βασίζεται σε *RPE-LTP* υβριδικό κωδικοποιητή, ενώ οι *CEL*P αποτελούν το πρότυπο για κινητή τηλεφωνία στην Βόρειο Αμερική.

2.9 Το σχήμα κωδικοποίησης που χρησιμοποιήσαμε

Στις προηγούμενες παραγράφους είδαμε αναλυτικά τεχνικές κβαντισμού και κωδικοποίησης της φωνής. Σημειώθηκε ότι η χρήση του *Filterbank Spectrum Analyzer* σαν μηχανισμού *front-end*, καθώς και η χρήση διανυσματικού κβαντισμού βάσει μικρότερων υποδιανυσμάτων (*product VQ*), παίζουν σημαντικό ρόλο στην επιτυχία του σχήματος κωδικοποίησης αυτής της εργασίας, την οποία και παρουσιάζουμε σε αυτό το δεύτερο μέρος.

Ο *front-end* μηχανισμός παράγει τα ακουστικά διανύσματα με τους 13 ή 39 *MFCC* συντελεστές για κάθε *frame* (δηλαδή διάστημα 10 *msecs*), με αποτέλεσμα ο ρυθμός πληροφορίας να είναι 1300 (ή 3900) συντελεστές αντίστοιχα ανά *second*. Αφού κάθε συντελεστής αντιπροσωπεύεται από μια *float* μεταβλητή απαιτούνται 1300 ή 3900 * 32 *bits/sec*, δηλαδή 41,6 ή 124.8 *kbps* αντίστοιχα. Αυτός ο ρυθμός είναι μικρότερος σε σχέση με αυτό της αρχικής πηγής πληροφορίας (δειγματοληπτημένη κυματομορφή σε 8 ή 16 *Khz* και με 16 *bits* ανά σύμβολο) και σκοπός μας είναι να τον μειώσουμε ακόμα περισσότερο, χρησιμοποιώντας τεχνικές κβαντισμού.

Θυμίζουμε ότι με τον κβαντισμό, ουσιαστικά περιορίζουμε τον αριθμό συμβόλων εξόδου, σε ένα μικρό σύνολο προτύπων συμβόλων εξόδου, καταφέροντας να συμπίεσουμε από τη μια το σήμα αλλά και δυστυχώς να εισάγουμε θόρυβο από την άλλη. Στην εισαγωγή αυτού του κεφαλαίου, τονίσαμε ότι επειδή στόχος μας δεν είναι η όσο πιο πιστή αναπαραγωγή του σήματος, αλλά η καλλίτερη δυνατή αναγνώριση, την απώλεια πληροφορίας που έχουμε με τον κβαντισμό, τη μετράμε με βάση τα αποτελέσματα της αναγνώρισης. Αυτό μας βοηθάει πρακτικά στο να αξιολογήσουμε τα διάφορα σχήματα κωδικοποίησης που θα αναπτύξουμε. Τα σημαντικότερα αποτελέσματα από τα πειράματα που διεξάχθηκαν προκειμένου να καταλήξουμε στο καλλίτερο μέχρι στιγμής αποτέλεσμα, με ρυθμό μετάδοσης μόλις 2 *kbps*, παρατίθενται στο τέλος του κεφαλαίου. Στο Κεφάλαιο 4, παραθέτουμε κάποια άλλα ενδεικτικά πειράματα που έγιναν, πριν καταλήξουμε στα αποτελέσματα που παρουσιάζουμε σε αυτό το κεφάλαιο.

Σε αυτή την ενότητα, θα θυμίσουμε κάποιες βασικές έννοιες-ορολογίες, όπως αυτές των *codebooks*, *centroids* και θα ορίσουμε κάποιες άλλες. Στη συνέχεια θα εξετάσουμε τις διαδικασίες εκπαίδευσης (δημιουργία *codebooks* με βάση τον επαναληπτικό αλγόριθμο *k-means*) και κβαντισμού, στην περίπτωση της (*product VQ*) τεχνικής, η οποία αποτελεί και την ‘καρδιά’ της κωδικοποίησης. Τα παραπάνω πλαισιώνονται με αρκετά σχήματα, προκειμένου να γίνουν ακόμα πιο ξεκάθαρες οι παραπάνω έννοιες και διαδικασίες που περιγράφονται.

2.9.1 Βασικές έννοιες

Εδώ θα θυμίσουμε κάποιες βασικές έννοιες και θα ορίσουμε κάποιες νέες, όπως αυτές θα χρησιμοποιούνται στη συνέχεια, προσαρμοσμένες στην περίπτωση του VQ με υποδιανύσματα.

Είσοδος συστήματος κβαντισμού.

Η είσοδος στο σύστημα κβαντισμού είναι το $MFCC$ διάνυσμα, όχι όμως απαραίτητα με τη μορφή μονάχα του ενός διανύσματος. Μπορούμε να το αναπαραστήσουμε με περισσότερα (υπο-)διανύσματα, αρκεί αυτά συνδυαζόμενα να μας δίνουν την το αρχικό διάνυσμα. Έτσι, αν υποθέσουμε ότι το $MFCC$ διάνυσμα έχει μέγεθος 13 (*high quality* - χωρίς χρήση των παραγώγων), τότε έχουμε επιλογές όπως :

1. Να χρησιμοποιήσουμε το διάνυσμα ως έχει : $[1 \ 2 \ 3 \ \dots \ 13]$
2. Να χρησιμοποιήσουμε 13 βαθμωτούς : 1, 2, 3, ..., 13
3. Να χρησιμοποιήσουμε L ($1 \leq L \leq 13$) υποδιανύσματα, όπως π.χ. : $[1 \ 2 \ \dots \ 7]$, $[8 \ \dots \ 12]$, $[13]$ ή $[1 \ 3 \ 5 \ 7 \ \dots \ 13]$ $[2 \ 4 \ 6 \ \dots \ 12]$ κ.λ.π.
4. Οποιαδήποτε άλλη επιλογή κρίνεται κατάλληλη. Θα δούμε στη συνέχεια το τρόπο επιλογής υποδιανυσμάτων

Και οι 3 παραπάνω επιλογές είναι περιπτώσεις του διανυσματικού κβαντισμού (*Vector quantization*). Η 2η είναι περίπτωση βαθμωτού κβαντισμού, ενώ η 3η είναι ο *code product* κβαντισμός. Στην 1η περίπτωση χρησιμοποιούμε ένα μόνο κβαντιστή, στη 2η δεκατρείς & στη 3η τρεις και δυο κβαντιστές αντίστοιχα.

Αναπαράσταση της εξόδου του συστήματος κβαντισμού.

Αφού εξετάσαμε τις δυνατές εισόδους, ας κάνουμε το ίδιο για την έξοδο. Πρώτα όμως να θυμίσουμε ότι στη διαδικασία κβαντισμού απεικονίζουμε ένα δείγμα εισόδου σε ένα μόνο δείγμα, από ένα πεπερασμένο αριθμό δειγμάτων εξόδου (έστω M), τα οποία προκύπτουν κατάλληλα από την διαδικασία εκπαίδευσης των δεδομένων εκπαίδευσης. Το γεγονός ότι τα δείγματα εξόδου είναι πεπερασμένα, μας δίνει ένα επιπλέον τρόπο απεικόνισης. Έτσι έχουμε 2 τρόπους απεικόνισης, αν κβαντίσουμε το δείγμα εισόδου στο i -οστό από τα πεπερασμένα δείγματα :

το i -οστό διάνυσμα (*codeword*) ή απλά τον αριθμό i (*index*). Το είδος της απεικόνισης είναι αυτό που καθορίζει και το είδος ακουστικών μοντέλων στον αναγνωριστή. Στην 1η περίπτωση όπου απεικονίζουμε ένα δείγμα-διάνυσμα μεγέθους N σε ένα επίσης διάνυσμα μεγέθους N

μιλάμε για *Centroid Coded* κβαντισμό. Στη 2η περίπτωση, όπου απεικονίζουμε το δείγμα-διάνυσμα μεγέθους N στον ακέραιο που χαρακτηρίζει το i -οστό αυτό διάνυσμα (δηλ. το i) μιλάμε για *Index Coded* κβαντισμό.

Centroid και Codebook

Χωρίζοντας τον N -διάστατο χώρο σε M περιοχές, κάθε μια από αυτές τις περιοχές έχει ένα σημείο αναφοράς, το *centroid*. Όλα τα σημεία που ανήκουν σε μια περιοχή λέμε ότι έχουν κβαντιστεί στο *centroid* αυτής της περιοχής. Η απόσταση¹ των σημείων που ανήκουν σε μια περιοχή με το *centroid* της ίδιας περιοχής, είναι μικρότερη από την απόσταση με όποιο άλλο *centroid* του χώρου : αυτό είναι και το κριτήριο (*nearest neighbour*), με βάση το οποίο τα σημεία του χώρου κβαντίζονται σε κάποιο *centroid* και σχηματίζουν κάποια περιοχή. Το *codebook*, εμπεριέχει τα M *centroids* του N -διάστατου χώρου (βλέπε σχήμα 2.2).

Σχήμα κωδικοποίησης, εκπαίδευσης και κβαντισμού

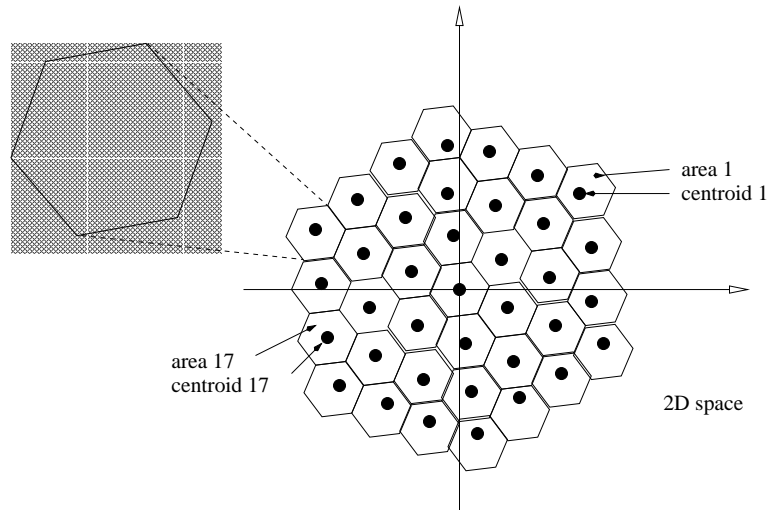
Το σχήμα εκπαίδευσης καθορίζει το πόσα και ποια υποδιανύσματα έχουμε καθώς και το μέγεθος των *codebooks* που αντιστοιχούν σε αυτά τα υποδιανύσματα. Το σχήμα κβαντισμού καθορίζει το ποιά από τα διαθέσιμα *codebooks* θα χρησιμοποιήσουμε σε ένα κβαντισμό. Τέλος, ο όρος σχήμα κωδικοποίησης χρησιμοποιείται από εδώ και πέρα σαν ο συνδυασμός των δυο παραπάνω.

2.9.2 Διαδικασία εκπαίδευσης

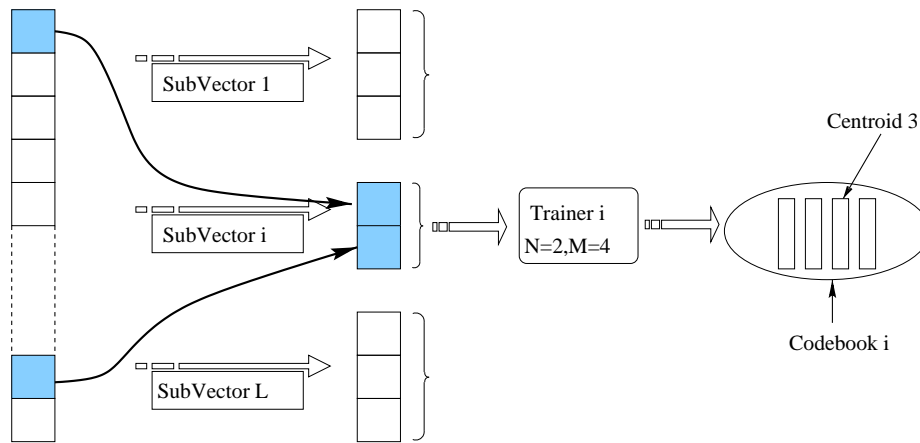
Η διαδικασία εκπαίδευσης αποσκοπεί στην δημιουργία των *codebooks*, τα οποία στη συνέχεια θα χρησιμοποιηθούν από τον κβαντιστή. Για κάθε *frame* στο *train set*, χωρίζουμε το *MFCC* διάνυσμα σε μικρότερα έστω L υποδιανύσματα, σύμφωνα με το σχήμα εκπαίδευσης που έχουμε επιλέξει. Αν συμβολίσουμε με N_i το μέγεθος του υποδιανύσματος i και M_i το αριθμό των *centroids* στο i *codebook*, τότε με το τέλος της διαδικασίας εκπαίδευσης, (μέσω των L εκπαιδευτών) θα έχουν παραχθεί τα αντίστοιχα L *codebooks* μεγέθους M_i και διάστασης N_i το καθένα (βλέπε σχήμα 2.3).

Οι *trainers* χρησιμοποιούν τον *k-means* αλγόριθμο τον οποίο και θα περιγράψουμε στη συνέχεια. Τα σύμβολα N , M , B που εμφανίζονται στον αλγόριθμο (θεωρούμε ότι $M = 2^B$), αντιστοιχούν στα N_i και M_i για το κάθε υποδιάνυσμα (και κατ' επέκταση για τον αντίστοιχο από τους L *trainers*).

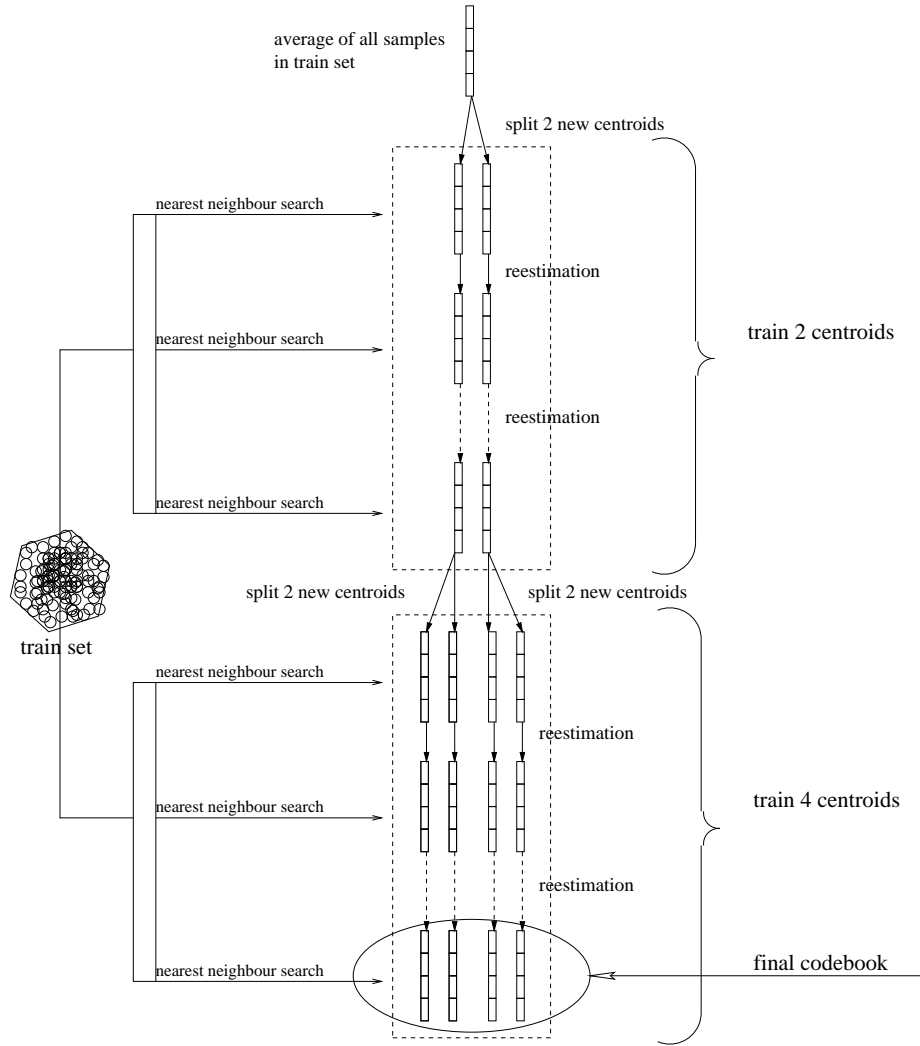
¹ Λέγοντας απόσταση εννοούμε κάποια κατάλληλα επιλεγμένη μετρική. Συνήθως κάποια *weighted Euclidean* μετρική όπως την *Mahalanobis*. Έτσι, χρησιμοποιώντας αργότερα τον όρο 'μοιάζει' εννοούμε ότι ελαχιστοποιείται η παραπάνω απόσταση



Σχήμα 2.2: **Codebook & Centroids.** Παράδειγμα για $N=2$ και $M=36$ όπου φαίνεται ο διδιάστατος χώρος με 36 περιοχές. Κάθε περιοχή έχει κι ένα *centroid*, το οποίο έχει την ιδιότητα να αντιπροσωπεύει καλλίτερα όλα τα σημεία της περιοχής. Το σύνολο των 36 αυτών *centroids* αποτελεί το *codebook* δηλαδή τα 36 διδιάστατα διανύσματα



Σχήμα 2.3: **Διαδικασία εκπαίδευσης.** Χωρίζουμε κάθε *MFCC* διάνυσμα σε L υποδιανύσματα με μέγεθος N_i το καθένα. Για κάθε υποδιάνυσμα, αποφασίζουμε πόσους και ποιους συντελεστές θα περιέχει, καθώς και πόσα *centroids* στο *codebook* (στη συγκεκριμένη περίπτωση 4).



Σχήμα 2.4: **Εκπαίδευση και δημιουργία codebook** Παράδειγμα δημιουργίας 4 *centroids* στο *codebook*. Ξεκινώντας από την αρχική τιμή και εφαρμόζοντας διαδοχικά διαδικασίες *splitting* και επανεκπαίδευσης των *centroids*, παίρνουμε το τελικό *codebook* (με 4 *centroids*)

Αλγόριθμος εκπαίδευσης :

Ας υποθέσουμε ότι θέλουμε να χωρίσουμε τον N -διάστατο χώρο σε $M = 2^B$ περιοχές. Το N είναι το μέγεθος του υποδιανύσματος & το B είναι ο αριθ-

μός των *bits* που θέλουμε να χρησιμοποιήσουμε. Αν π.χ. έχουμε $B=5$, αυτό σημαίνει ότι τελικά θα έχουμε 32 *centroids* στο *codebook*. Ο αλγόριθμος εκπαίδευσης που χρησιμοποιούμε είναι ο *k-means* (σε κάποιες παραλλαγές είναι γνωστός και ως *generalized Lloyd*, *LBG* ή και *ISODATA* αλγόριθμος), εφαρμόζεται για κάθε ένα από τα υποδιανύσματα που έχουμε ορίσει και δίνεται παρακάτω :

Αρχικοποίηση (Initialization):

Θέσε αρχικό μοναδικό *centroid*, το υποδιάνυσμα που αντιστοιχεί στη μέση τιμή όλων των τιμών του υποδιανύσματος στο *train set* (σετ εκπαίδευσης). Η επιλογή αυτή είναι αρκετά αντιπροσωπευτική και είμαστε τώρα έτοιμοι να περάσουμε στον επαναληπτικό αλγόριθμο.

Σε κάθε βήμα $i : (0 \leq i < B)$

Βήμα 1 : Χώρισε κάθε ένα από τα n *centroids* ($1 \leq n \leq 2^i$) σε 2 νέα, εφαρμόζοντας δυαδικό χωρισμό (*binary splitting*).

Αυτό σημαίνει ότι από κάθε *centroid* y_n παίρνουμε 2 νέα, τα : $(y_n^+ = y_n + e)$ & $(y_n^- = y_n - e)$ τα οποία όμως δεν είναι αρκετά αντιπροσωπευτικά και χρειάζονται εκπαίδευση για να γίνουν. Η τιμή του e είναι της τάξης του 0.05

Βήμα 2 : Για κάθε επανάληψη j (Ο αριθμός των επαναλήψεων αυξάνει γραμμικά με το βήμα : π.χ. $j=i+5$)

1. Για κάθε υποδιάνυσμα του σετ εκπαίδευσης (*Clustering - Nearest Neighbour search*):

Υπολόγισε την απόστασή του από όλα τα υπάρχοντα *centroids* και ομαδοποίησέ το στο *centroid* με το οποίο το μέτρο της μεταξύ τους απόστασης είναι ελάχιστο. Το μέτρο της απόστασης που χρησιμοποιείται παίζει μεγάλο ρόλο. Συνήθως χρησιμοποιείται η *Mahalanobis* απόσταση.

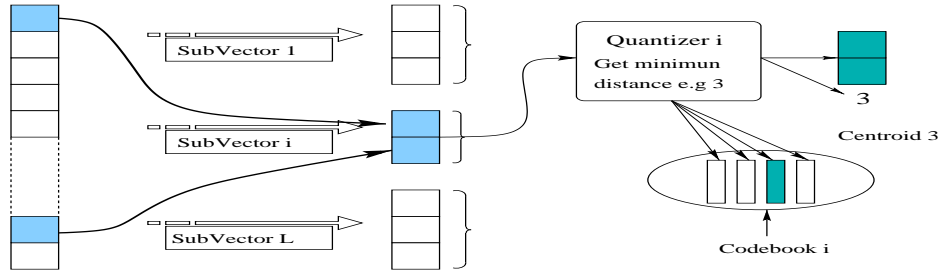
2. Υπολόγισε τις νέες τιμές των *centroids* (*Reestimation - Centroid Update*):

Η νέα τιμή για κάθε *centroid*, είναι η μέση τιμή όλων των υποδιανυσμάτων που έχουν ομαδοποιηθεί στην προηγούμενη τιμή του *centroid*

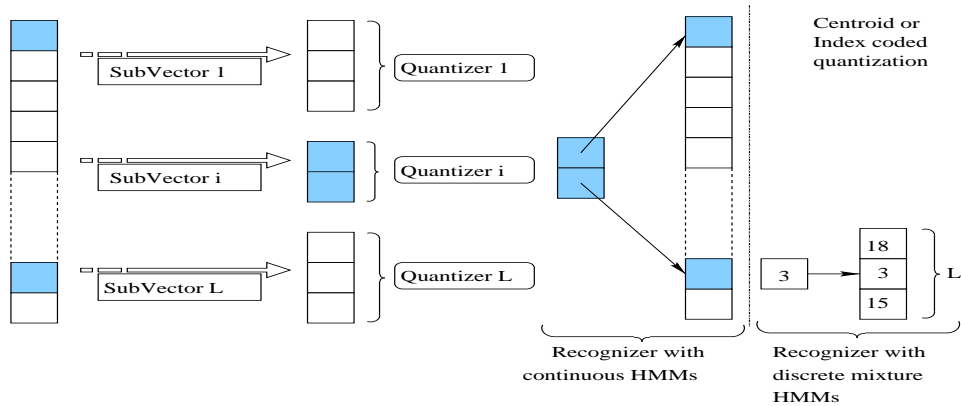
Ο αλγόριθμος φαίνεται καλλίτερα στο σχήμα 2.4

2.9.3 Διαδικασία κβαντισμού

Η διαδικασία κβαντισμού είναι πιο απλή. Η ιδέα στην οποία βασίζεται, είναι ότι κάθε υποδιάνυσμα θα πρέπει να κβαντιστεί στο *centroid* εκείνο του *codebook*, με το οποίο μοιάζει περισσότερο ή αλλιώς με το *centroid* με το οποίο η μεταξύ τους απόσταση είναι μικρότερη. Περισσότερα σχόλια στα σχήματα 2.5 και 2.6.



Σχήμα 2.5: Διαδικασία κβαντισμού. Για κάθε υποδιάνυσμα i , ο κβαντιστής βρίσκει το *centroid* που το αντιπροσωπεύει καλλίτερα, δηλαδή το *centroid* με το οποίο η απόσταση είναι μικρότερη. Όσο για την έξοδο του κβαντιστή, αυτή μπορεί να είναι ή το ίδιο το *centroid* ή το *index* του. Στη συγκεκριμένη περίπτωση, το υποδιάνυσμα i λέμε ότι κβαντίστηκε στο 3ο *centroid*.



Σχήμα 2.6: Έξοδος συστήματος κβαντιστών και αναγνώριση. Αν έχουμε *centroid coded* κβαντισμό, από το σύστημα των κβαντιστών προκύπτει ένα ανακατασκευασμένο *MFCC* διάνυσμα, αφού τα *elements* κάθε *centroid* παίρνουν την θέση τους στο αρχικό διάνυσμα. Η αναγνώριση για *centroid coded* κβαντισμό, γίνεται από ένα *Recognizer* με συνεχή *HMM* μοντέλα. Αν έχουμε *index coded* κβαντισμό, η έξοδος των κβαντιστών είναι ένα διάνυσμα από τα αντίστοιχα των *centroid indices*. Προκύπτει έτσι ένα διάνυσμα μήκους L , όσο και ο αριθμός των υποδιανυσμάτων. Η αναγνώριση για *index coded* κβαντισμό γίνεται από ένα *Recognizer* με μείγματα διακριτών *HMMs*.

2.10 Αξιολόγηση κωδικοποίησης

Στις δυο προηγούμενες παραγράφους μιλήσαμε για την διαδικασία εκπαίδευσης και κβαντισμού. Επισημάνθηκε επίσης προηγούμενα ότι η κωδικοποίηση που χρησιμοποιούμε δεν έχει στόχο την ακριβή ανακατασκευή του σήματος της φωνής (ακουστικών διανυσμάτων), αλλά την καλλίτερη δυνατή αναγνώριση (κωδικοποίηση φωνής για αναγνώριση). Εύλογα λοιπόν η αξιολόγηση ενός σχήματος κωδικοποίησης γίνεται με την αναγνώριση των κβαντισμένων ακουστικών διανυσμάτων.

2.11 Συμπεράσματα

- Οι τεχνικές κωδικοποίησης χωρίζονται σε δυο κατηγορίες ανάλογα με το αν τηρούν το θεώρημα κωδικοποίησης πηγής του *Shannon*. Αυτές που το τηρούν εξασφαλίζουν κωδικοποίηση χωρίς απώλεια πληροφορίας (*lossless*), ενώ οι *lossy* τεχνικές ή αλλιώς τεχνικές κβαντισμού, προσπαθούν να συμπίσουν ακόμα περισσότερο τα σύμβολα εξόδου της πηγής, με τίμημα την απώλεια κάποιου ποσού πληροφορίας.
- Πιο γνωστές τεχνικές κβαντισμού είναι ο βαθμωτός (ομοιόμορφος και μη) και ο διανυσματικός ο οποίος υπερέχει του πρώτου. Τεχνικές κωδικοποίησης φωνής κωδικοποιούν απ'ευθείας τη κυματομορφή ή παράγουν τις παραμέτρους του φωνητικού σωλήνα, του μοντέλου της φωνής.
- Σε αντίθεση με τις συνήθεις τεχνικές κωδικοποίησης, που σαν σκοπό έχουν την πιστή αναπαραγωγή του σήματος της φωνής, η κωδικοποίηση που εφαρμόζουμε υπηρετεί ένα διαφορετικό σκοπό : Την μέγιστη συμπίεση με σκοπό την καλλίτερη δυνατή αναγνώριση .
- Το σχήμα κωδικοποίησης που χρησιμοποιήσαμε, πηγαίνοντας ένα βήμα παραπέρα, κβαντίζει τα ακουστικά διανύσματα του *front-end*, προκειμένου να μειωθεί ακόμα περισσότερο το *bandwidth* που απαιτείται για την μετάδοσή τους.
- Για τη συμπίεση, χρησιμοποιείται σύστημα διανυσματικού κβαντισμού βασισμένο στην χρήση υποδιανυσμάτων (*product VQ* τεχνική). Στην διάρκεια της εκπαίδευσης κατασκευάζονται πρότυπα διανύσματα με τη βοήθεια του *k-means* αλγορίθμου. Κατά τη διαδικασία του κβαντισμού τα υποδιανύσματα κβαντίζονται στο καταλληλότερο πρότυπο υποδιάνυσμα. Στο Κεφάλαιο 4 θα δούμε πως υλοποιήθηκε η κωδικοποίηση με το

σύστημα *Yarrow* καθώς και τη χρήση του για την εκπαίδευση και τον κβαντισμό.

- Η αξιολόγηση της κωδικοποίησης γίνεται με την αναγνώριση του κωδικοποιημένου σήματος. Τα πειράματα για την αξιολόγηση της κωδικοποίησης αναλύονται στο Κεφάλαιο 4.

2.12 Παραπομπές

Δυο κλασικές εργασίες στη χρήση διανυσματικού κβαντισμού παρουσιάζονται στα [6] και [7]. Στη πρώτη δίνεται έμφαση στις διαφορές βαθμωτού και διανυσματικού κβαντισμού και παρουσιάζεται η αποτελεσματικότητα του δεύτερου, καθώς βελτιστοποιεί την παρουσία γραμμικής και μη γραμμικής εξάρτησης, σχήματος συνάρτησης πυκνότητας πιθανότητας (*probability density function*) και της διάστασης του διανύσματος. Στη δεύτερη παρουσιάζεται μια σειρά διαφορετικών τεχνικών βασισμένων σε διανυσματικό κβαντισμό. Για πιο ολοκληρωμένη εικόνα το [8] είναι ένα βιβλίο αφιερωμένο σε τέτοιες τεχνικές. Τεχνικές κωδικοποίησης φωνής παρουσιάζονται επίσης στο βιβλίο [9], ενώ στο [5] δίνεται έμφαση στις τεχνικές κωδικοποίησης πηγής.

Κεφάλαιο 3

Υλοποίηση της κωδικοποίησης - το σύστημα YARROW

3.1 Εισαγωγή

Η υλοποίηση της κωδικοποίησης που παρουσιάσαμε προηγούμενα έγινε επεκτείνοντας το σύστημα φωνής *Yarrow*. Το *Yarrow* σύστημα άρχισε να αναπτύσσεται στο *S.R.I* από το *Leonardo Neumeyer* το 1997. Σκοπός του *project* είναι να δημιουργηθεί ένα πλαίσιο από γενικές βιβλιοθήκες (*Application Programming Interface - API*) σε τομείς όπως ψηφιακή επεξεργασία σήματος, αναγνώριση προτύπων & στατιστική μοντελοποίηση, για χρήση τόσο σε έρευνα όσο και σε γενικότερες εφαρμογές. Η δημιουργία του βασίστηκε στην ανάγκη για την ύπαρξη ενός *front-end* μηχανισμού επεξεργασίας φωνής που να είναι εύκολα μεταφέρσιμος σε διάφορες πλατφόρμες, εύκολος στη χρήση και να έχει μικρές απαιτήσεις σε πόρους κάτι που οδήγησε στην επιλογή της γλώσσα προγραμματισμού *Java* (www.java.sun.com) για την υλοποίησή του.

Σε αυτή την ενότητα περιγράφεται η βασική δομή του συστήματος και αναλύεται ο τρόπος επέκτασής του, με σκοπό τη δημιουργία μιας πλατφόρμας που να επιτρέπει την ερευνητική διαδικασία πάνω στο θέμα του διανυσματικού κβαντισμού των παραμέτρων της φωνής. Αρχικά περιγράφεται εκτενώς η αρχιτεκτονική του κώδικα που αναπτύχθηκε για την υποστήριξη διανυσματικού κβαντισμού και παρουσιάζεται η διασύνδεση με το σύστημα αναγνώρισης φωνής *Decipher*, το οποίο προσαρμόστηκε κατάλληλα για αυτό το σκοπό. Επίσης αναλύεται η χρήση του νέου συστήματος *Yarrow* για την διεξαγωγή πειραμάτων κωδικοποίησης φωνής. Στόχος είναι να χρησιμοποιηθεί το *Yarrow* στον πελάτη και το *Decipher* στον εξυπηρετητή σύμφωνα με το σενάριο πελάτη-εξυπηρετητή που αναφέρθηκε στο εισαγωγικό κεφάλαιο.

3.2 Βιβλιοθήκες του αρχικού συστήματος

Οι βιβλιοθήκες (*packages*) του αρχικού συστήματος *Yarrow* είναι οι εξής:

- **sri.star.dsp**

Αυτή είναι η βασική βιβλιοθήκη του *front-end*. Ο σχεδιασμός βασίζεται σε αυτή του *Decipher*. Το *front-end* χωρίζει τη κυματομορφή σε σταθερού μήκους τμήματα (*frames*), και υπολογίζει διάφορες αναπαραστάσεις της αρχικής κυματομορφής που ονομάζονται *features*. Το *front-end* υλοποιείται σαν μια συλλογή από τέτοια *feature objects*. Η συλλογή αυτή αποτελείται από ένα γράφο με καθορισμένη τοπολογία. Στο γράφο αυτό κάθε *feature* είναι ένας κόμβος (*node*) με ένα σύνολο από *arcs* σαν είσοδο, από άλλα *features*. Για κάθε *frame*, προκειμένου να υπολογιστεί κάποιο *feature*, πρέπει να υπολογιστούν όλα τα άλλα από τα οποία εξαρτάται σύμφωνα με το γράφο. Η χρήση σχήματος κρυφής μνήμης επιταχύνει τη παραπάνω διαδικασία.

Η παραγωγή *features* βασίζεται στη χρήση μιας σειράς φίλτρων (*bank of filters front-end processor*) η οποία είναι παρούσα στις περισσότερες υλοποιήσεις συστημάτων φωνής. Σκοπός είναι η εκτίμηση της ενέργειας του σήματος φωνής στις διάφορες μπάντες συχνότητας (συνήθως *Mel frequency*). Έτσι τελικά σχηματίζεται το *MFCC* διάνυσμα που είδαμε σε προηγούμενο κεφάλαιο. Για υψηλής ποιότητας φωνή, χρησιμοποιούνται 25 τέτοια φίλτρα. Κάθε ένας από τους 12 *cepstral* συντελεστές προκύπτει από το γινόμενο ενός διανύσματος βαρών για κάθε από τις 25 ενέργειες με το διάνυσμα των 25 ενεργειών. Στο διάνυσμα των 12 συντελεστών προστίθεται και το 'άθροισμά' τους, αποκτώντας έτσι το τελικό διάνυσμα με τους 13 συντελεστές (*cepstrum feature*). Υπάρχουν και άλλες μορφές *cepstrum* όπως το *rasta* και *zero mean* το οποίο και χρησιμοποιήσαμε. Από εδώ και πέρα λέγοντας *cepstrum* εννοούμε το τελευταίο. Το άλλο *MFCC* διάνυσμα που χρησιμοποιούμε είναι το *Single* και αποτελείται από το *cepstrum* μαζί με τις πρώτες και δεύτερες παραγωγούς (σύνολο 39 συντελεστές).

- **sri.star.util**

Αυτή η βιβλιοθήκη περιέχει αρκετές βοηθητικές κλάσεις όπως στατιστικές μεθόδους για διανύσματα & πίνακες, και κλάσεις για διάβασμα - γράψιμο κυματομορφών, με *NIST Sphere* τύπο αρχείων.

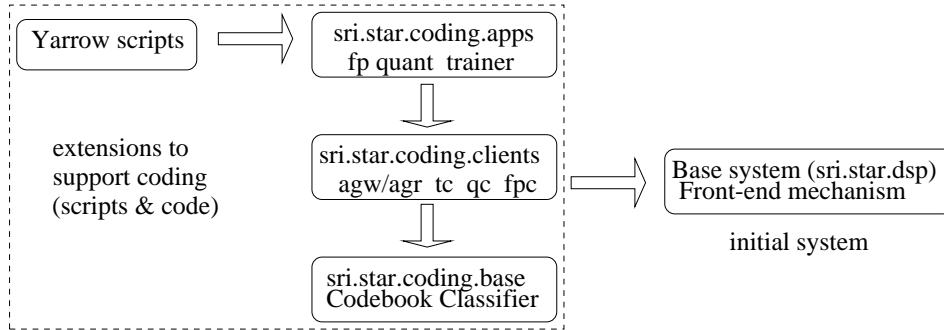
- **sri.star.wavsource**

Γενική βιβλιοθήκη με σκοπό την υποστήριξη περισσότερων ειδών εισόδου εκτός από κυματομορφές *NIST*.

3.3 Επέκταση του συστήματος για υποστήριξη VQ

3.3.1 Σκοπός

Σκοπός της επέκτασης του συστήματος *Yarrow* είναι να μπορεί να εφαρμοστεί διανυσματικός κβαντισμός στα παραγόμενα από το *front-end features* (κάτι το οποίο παρέχεται ήδη από την αρχική μορφή του *Yarrow*), ώστε να είναι δυνατή η μέγιστη συμπίεση του σήματος της φωνής, εξασφαλίζοντας παράλληλα υψηλή ακουστική ανάλυση. Για την διεξαγωγή των πειραμάτων, θα πρέπει να είναι δυνατή η επιλογή διαφόρων σχημάτων κωδικοποίησης. Από τη μεριά του *Yarrow* θα πρέπει να παρέχεται η δυνατότητα εκπαίδευσης των *codebooks* για το *train set*, καθώς και η χρήση τους για την κβαντοποίηση του *test set*. Από την άλλη πλευρά, τα κβαντισμένα αρχεία θα δίνονται σαν είσοδος στον αναγνωριστή του *Decipher*, για την αξιολόγηση του σχήματος κωδικοποίησης με το οποίο γίνονται κάθε φορά τα πειράματα. Η τρόπος επέκτασης του *Yarrow* για την υλοποίηση του σχήματος κωδικοποίησης αυτής της εργασίας φαίνεται στο Σχήμα 3.1 όπου παρουσιάζεται η αρχιτεκτονική του κώδικα, ο τρόπος χρήσης του από τα πειράματα μέσω των *scripts* που αναπτύχθηκαν καθώς και η διασύνδεση με το αρχικό σύστημα *Yarrow*.



Σχήμα 3.1: **Αρχιτεκτονική Yarrow συστήματος** Η επέκταση αφορά στην προσθήκη των *sri.star.coding packages* και των τελικών *scripts*. Όπως θα δούμε στη συνέχεια το *fpc package* αποτελεί το συνδετικό κρίκο ανάμεσα στα 2 συστήματα.

3.4 Αρχιτεκτονική συστήματος και υλοποίηση

Η πλατφόρμα του *Yarrow* χρησιμοποιήθηκε για την υλοποίηση του σχήματος κωδικοποίησης. Οι 3 παραπάνω βιβλιοθήκες αποτέλεσαν μια πραγματική βάση

για την περαιτέρω επέκταση. Οι νέες βιβλιοθήκες που προστέθηκαν έχουν επεκτείνει το όνομα `sri.star` με το `sri.star.coding` για να είναι φανερό ότι ανήκουν στην βιβλιοθήκη κωδικοποίησης. Η βιβλιοθήκη αποτελείται από τις βασικές (`core`) κλάσεις (`sri.star.coding.base`), τις βοηθητικές (`sri.star.coding.utils`), τους πελάτες (`sri.star.coding.clients.*`) & τις εφαρμογές (`sri.star.coding.apps.*`). Αναλυτικότερα τα `packages` που αναπτύχθηκαν είναι τα ακόλουθα :

- `sri.star.coding.base`
- `sri.star.coding.clients.agw`
- `sri.star.coding.clients.agr`
- `sri.star.coding.clients.fpc`
- `sri.star.coding.clients.qc`
- `sri.star.coding.clients.net`
- `sri.star.coding.clients.tc`
- `sri.star.coding.clients.utils`
- `sri.star.coding.utils`
- `sri.star.coding.apps.fp`
- `sri.star.coding.apps.quant`
- `sri.star.coding.apps.trainer`
- `sri.star.coding.apps.utils`

3.5 Βιβλιοθήκη `sri.star.coding.base`

Αποτελεί πράγματι τη βάση του συστήματος. Περιέχει τις κλάσεις που αποτελούν το συνδετικό κρίκο ανάμεσα στην εκπαίδευση & τον κβαντισμό. Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο ο συνδετικός αυτός κρίκος δεν είναι άλλος από το `codebook`. Έτσι, αυτή η βιβλιοθήκη ορίζει και εκπαιδεύει τα `codebooks`. Οι βασικές κλάσεις είναι :

TrainerInformation

Κλάση υπεύθυνη για τη συλλογή των στατιστικών ιδιοτήτων του υποδιαγύσματος στο `train set`, απαραίτητες για την εκπαίδευση των `codebooks`

Codebook

Το *codebook* δεν είναι παρά μια συλλογή από 2^B διανύσματα μήκους L (*centroids*), αυτά που αντιπροσωπεύουν καλλίτερα με B *bits* πληροφορίας το υποδιάνυσμα στο *train set*. Έτσι αυτή η κλάση εκτός από τον ορισμό, υλοποιεί μεθόδους όπως :

- τον διαμοιρασμό (*split*) ενός *centroid* σε 2 νέα (που χρησιμοποιείται κατά την εκπαίδευση),
- τον κβαντισμό ενός υποδιανύσματος εισόδου στο πλησιέστερο από τα *centroids* του *codebook*,
- το διάβασμα/γράψιμο του *codebook*

Classifier

Είναι το αντικείμενο που υλοποιεί τον *k-means* αλγόριθμο για την εκπαίδευση του *codebook*.

3.6 Βιβλιοθήκη πελατών (sri.star.coding.clients.*)

Οι πελάτες είναι κλάσεις που βρίσκονται ένα επίπεδο πάνω από τις βασικές κλάσεις και ένα επίπεδο κάτω από τις εφαρμογές στο *abstraction layer* του *coding package* (βλέπε Σχήμα 3.1). Με λίγα λόγια υλοποιούν όλες τις εξειδικευμένες λειτουργίες που χρειάζονται οι εφαρμογές.

3.6.1 Πελάτες αρχείων

Το *Yarrow* σύστημα, στη βασική του μορφή δεν παράγει αρχεία, απλώς κάνει *front-end* επεξεργασία σε αρχεία κυματομορφών. Παρ' όλα αυτά, καθορίζει κάποιο τύπο αρχείων που όμως δεν το υλοποιεί. Το φορματ καθορίζει *ascii* αρχεία, τα οποία είναι σε συμπιεσμένη μορφή. Οι δυο επόμενες κατηγορίες υλοποιούν το διάβασμα & γράψιμο *features* από/σε αυτό το φορματ. Λέγοντας *feature*, εννοούμε κβαντισμένα ή όχι *features*, όπως π.χ. *cepstrum*, *centroid coded* ή *index coded cepstrum*. Έτσι υπάρχει ένα μοναδικό καλά ορισμένο και απλό σχετικά φορματ, για όλα τα *features* του *Yarrow* συστήματος.

Το φορματ καθορίζει κάποιο *header* και τα δεδομένα. Το *header* δίνει πληροφορία για τον τύπο του *feature* (*integer/float*), το όνομα (*cepstrum/Single*), το μέγεθος, και τον αριθμό των *frames*. Στο πρώτο παράδειγμα, έχουμε 175 *frames* από *cepstrum*. Το *Compute Deltas* δηλώνει ότι πρέπει να υπολογισθούν οι πρώτες και δεύτερες παράγωγοι, πριν γίνει η αναγνώριση. Παράδειγμα αρχείου με *cepstrum* φαίνεται στη συνέχεια :


```

VectorList
Type float
Name cepstrum
VectorSize 13
NumVectors 175
ComputeDeltas true
EndHeader
-600.9399 1.2876966 0.5145594 -2.0349026 22.101751 -5.7347784 -0.022707298
15.5569315 -26.187925 8.510943 1.3709744 -1.8681848 7.1525745
...
< τα υπόλοιπα 174 frames>

```

Στο 2ο παράδειγμα που ακολουθεί, έχουμε 381 *frames* από *Single* που έχουν κβαντιστεί-διακριτοποιηθεί σε 9 υποδιανύσματα :

```

VectorList
Type int
Name Single
VectorSize 9
NumVectors 381
ComputeDeltas false
EndHeader
16 113 52 1 8 18 10 3 2
5 108 48 32 65 48 10 3 2
6 120 16 8 3 20 65 2 33
4 96 16 4 108 36 64 65 33
...
< τα υπόλοιπα 377 frames >

```

Τα αρχεία αυτά έχουν ονόματα με τις εξής καταλήξεις :

| | |
|-------------------------|---------|
| cepstrum | .cep |
| Single | .Single |
| Centroid Coded Cepstrum | .ccc |
| Centroid Coded Single | .ccs |
| Index Coded Cepstrum | .icc |
| Index Coded Single | .ics |

Πίνακας 3.1: Παραγόμενα features και καταλήξεις αντίστοιχων αρχείων

Οι 2 βιβλιοθήκες που σχετίζονται με το διάβασμα και γράψιμο των αρχείων είναι οι :

- **sri.star.coding.clients.agw** *Ascii Gzipped Writer* κλάσεις. Υλοποιούν το γράψιμο των παραγόμενων αρχείων στο παραπάνω μορματ.
- **sri.star.coding.clients.agr** *Ascii Gzipped Reader* κλάσεις. Υλοποιούν το διάβασμα από αρχεία στο παραπάνω μορματ.

3.6.2 Πελάτες front-end

Πρόκειται για το *package* **sri.star.coding.clients.fpc**.

feature Produce Client (fpc) κλάσεις, όπως :

CepstrumProducerClient,

SingleProducerClient.

Με άλλα λόγια κλάσεις που παράγουν τα μη-κβαντισμένα *features* όπως *cepstrum* και *Single*. Οι κλάσεις αυτές είναι οι μόνες που χρησιμοποιούν άμεσα το *front-end* και αποτελούν τον ενδιαμέσο κρίκο ανάμεσα στο *front-end* & το *coding* υποσύστημα. Έτσι απομονώνεται το ένα υποσύστημα από το άλλο. Με αυτό το τρόπο όλες οι υπόλοιπες κλάσεις του *coding package*, δεν χρειάζονται να ξέρουν απολύτως τίποτα για το *front-end*.

3.6.3 Πελάτες κβαντισμού

Πρόκειται για το *package* **sri.star.coding.clients.qc**.

Quantizer Client (qc) κλάσεις, όπως :

QuantizerClient,

Είναι μια *abstract* κλάση που επεκτείνεται από τις υπόλοιπες του *package* και υλοποιεί τα ακόλουθα μεταξύ άλλων :

- το διάβασμα του *configuration*, που καθορίζει πόσα και ποια *codebooks* θα χρησιμοποιηθούν για τον κβαντισμό (σχήμα κβαντισμού, βλέπε παράγραφο 2.9.1).
- το φόρτωμα των *codebooks*, των προτάσεων του test-set και ότι άλλη αρχικοποίηση, είναι απαραίτητη.

CentroidQuantizerClient, *IndexQuantizerClient*.

Οι δυο αυτές κλάσεις υλοποιούν τα δυο είδη κβαντισμού, δηλαδή την απεικόνιση του κάθε υποδιανύσματος στο πλησιέστερο *centroid* του *codebook*, με έξοδο είτε το ίδιο το *centroid* (*centroid coded*), είτε στο δείκτη του *centroid* (*index coded*).

3.6.4 Πελάτες εκπαίδευσης

Πρόκειται για το **sri.star.coding.clients.tc**.

Κλάσεις όπως :

TrainerInfoClient,

Ο *TrainerInfoClient* συλλέγει τις στατιστικές ιδιότητες του υποδιανύσματος στο *train set*. Οι πληροφορίες αυτές είναι απαραίτητες για την εκπαίδευση.

TrainerClient,

Οι λειτουργίες του *TrainerClient* είναι :

- αρχικοποίηση πελάτη,
- καθορισμός σχήματος εκπαίδευσης (πόσα και ποια υποδιάνυσμα, μέγεθος *codebooks*, βλέπε 2.9.1)
- διάβασμα στατιστικών ιδιοτήτων του υποδιανύσματος,
- διάβασμα των δεδομένων εκπαίδευσης,
- δημιουργία των *centroids* (*training*),
- γράψιμο παραγόμενων *codebooks*, κ.α.

IndexTrainerClient.

Λειτουργεί σαν *wrapper* του *TrainerClient* επιτρέποντας την επιλογή των διανυσμάτων που θέλουμε να εκπαιδεύσουμε

Τέλος, υπάρχουν αρκετές κλάσεις που επεκτείνουν τον *TrainerClient*, αυξάνοντας την λειτουργικότητά του.

3.6.5 Υπόλοιποι πελάτες

Για τις υπόλοιπες λειτουργίες υπάρχουν πελάτες όπως αυτοί που ανήκουν στα πακέτα :

- **sri.star.coding.utils**
πελάτες που περιέχουν πιο γενικές βοηθητικές λειτουργίες.
- **sri.star.coding.net**
πελάτες υπεύθυνοι για την επικοινωνία *applet*-αναγνωριστή σε δίκτυο.
- **sri.star.coding.plugin**
πελάτες που χρησιμεύουν στη λειτουργία του *applet*.

3.7 Βιβλιοθήκη εφαρμογών (sri.star.coding.apps*)

Είδαμε παραπάνω ότι υπάρχουν πολλά επίπεδα στα οποία είναι οργανωμένος ο κώδικας. Όπως οι πελάτες βρίσκονται ένα επίπεδο πάνω από το *base package*, παρόμοια το *apps package* βρίσκεται ένα επίπεδο πάνω από τους πελάτες και οργανώνονται σε αντίστοιχα *packages* όπως εφαρμογές *front-end*, κβαντισμού, εκπαίδευσης (βλέπε σχήμα 3.1).

Κρύβοντας όλη την δύσκολη υλοποίηση στους πελάτες, οι εφαρμογές απλά συνδυάζουν ένα ή περισσότερους πελάτες για να υλοποιήσουν μια σχετικά πολύπλοκη διαδικασία, απλά με μερικές γραμμές κώδικα !

- **sri.star.coding.apps.fp**

Εφαρμογές που χρησιμοποιούν τα *sri.star.coding.clients.fpc* & *sri.star.coding.clients.agw packages*, δηλαδή εφαρμογές που παράγουν & γράφουν μη-κβαντισμένα *features* όπως *Single* & *cepstrum*. Κλάσεις όπως *SingleProducer* & *CepstrumProducer*.

- **sri.star.coding.apps.qc**

Εφαρμογές που υλοποιούν κβαντισμό. Υπάρχουν όλοι οι δυνατοί συνδυασμοί, ανάλογα με την είσοδο (κυματομορφή ή μη-κβαντισμένα *feature*) & το είδος της κβαντοποίησης (δείκτης ή διάλυσμα). π.χ. :

WaveformToCentroidCodedSingleQuantizer :

Διαβάζει από κυματομορφή, παράγει *Single feature* και μετά το κβαντίζει σε *centroid*. Χρησιμοποιεί πελάτες όπως :

SingleProducerClient

CentroidQuantizerClient

SingleWriterClient

CepstrumToIndexCodedCepstrumQuantizer :

Διαβάζει από *cepstrum* αρχείο τα *frames* και τα κβαντίζει σε δείκτες. Χρησιμοποιεί πελάτες όπως :

CepstrumReaderClient

IndexQuantizerClient

IndexCodedCepstrumWriterClient

- **sri.star.coding.apps.utils**

Χρήσιμες κλάσεις όπως : *CepstrumToSingle*(μετατροπéας), *CodebookInfo*, *ShowTrainerInfo*(τυπώνει πληροφορίες για αντικείμενα).

- **sri.star.coding.apps.trainer**

Κλάσεις όπως *Trainer*, *InitTrainer*, κ.α. Βασίζονται στους αντίστοιχους πελάτες *sri.star.coding.clients.tc*. Για παράδειγμα, ο *Trainer* το μόνο που κάνει, είναι να δημιουργεί ένα αντικείμενο τύπου *IndexTrainerClient* και να καλεί την μέθοδο *produceCodebooks()* (όπως φαίνεται και στο παράδειγμα που ακολουθεί).

```
package sri.star.coding.apps.trainer;

import sri.star.coding.clients.trainer.*;
import sri.star.util.*;

/**
 * Produces the desired codebooks for all subVectors
 *
 * @see sri.star.coding.clients.trainer.IndexTrainerClient
 * @author Perakakis Manolis
 * @version coding (1.0)
 */
public class Trainer {

    public static void main(String args[]) {
        String config
        String tmp "";
        int index;

        if(args.length == 2) {
            config = args[0];
            tmp = args[1];
        }
        else
            ErrorHandler.fatalError(null,
                "USAGE: java Trainer <config_file> <index> ");
        try {
            index = (new Integer(tmp)).intValue();
            IndexTrainerClient itc = new IndexTrainerClient(config,
index);
            itc.produceCodebooks();
        }catch (Exception e) {
            e.printStackTrace();}
```

```
}
}
```

3.8 Πειράματα VQ με το Yarrow

Μέχρι τώρα εξετάστηκε το *Yarrow* σε επίπεδο κώδικα. Όμως για να τρέξουν τα πειράματα χρειάζεται συχνά αρκετή προετοιμασία. Έτσι, πέρα από το *API*, γράφτηκαν τα απαραίτητα *scripts* τα οποία βρίσκονται ένα επίπεδο πάνω από τις εφαρμογές και απευθύνονται στον χρήστη, σε αντίθεση με το *API* που απευθύνεται στον *developer*. Τα πειράματα που θέλει να τρέξει κάποιος για να ελέγξει την επίδοση της κωδικοποίησης για κάποιο σχήμα, βασίζονται σε μια αλληλουχία ενεργειών που περιγράφεται στις 5 επόμενες παραγράφους :

3.8.1 Παραγωγή επιθυμητού feature από αρχεία κυματομορφών

Τόσο για το *train* όσο & για το *test set*. Αυτό γίνεται προκειμένου να αποφεύγεται η χρήση του *front-end* κάθε φορά που θέλουμε να δοκιμάσουμε ένα νέο σχήμα κωδικοποίησης¹ (θα χρησιμοποιήσουμε για το κάθε διάνυσμα). Αντίθετα έχοντας παράγει το επιθυμητό *feature* (π.χ. *Single*) τόσο για το *test set* όσο και για το *train set*, το μόνο που χρειάζεται να γίνεται κάθε φορά είναι μόνο η λειτουργία του κβαντισμού/εκπαίδευσης και όχι επιπλέον και η λειτουργία της παραγωγής του *feature* μέσω του *front-end* μια διαδικασία σχετικά χρονοβόρα. Με αυτό το τρόπο σε λιγότερο χρόνο μπορούμε να κάνουμε περισσότερα πειράματα. Το επιθυμητό *feature* μπορεί να είναι *cepstrum* ή *Single*. Το *script* που κάνει τα παραπάνω είναι το **go.produce.Single** για το *Single feature* και το **go.produce.cepstrum** για το *cepstrum*. Απαιτεί **configuration** αρχείο της μορφής :

```
input_files in.wavs
output_files out.cepstrum
```

δηλαδή απλά περιέχει τις λίστες με τα αρχεία κυματομορφών και τα παραγόμενα αρχεία.

¹Εδώ καλό είναι να θυμίσουμε τους όρους σχήμα κωδικοποίησης, εκπαίδευσης και κβαντισμού. Ο 1ος είναι πιο ευρύς και περιλαμβάνει τον συνδυασμό των επόμενων. Το σχήμα εκπαίδευσης καθορίζει τις επιλογές για την εκπαίδευση των *codebooks* (πώς θα ορίσουμε τα υποδιανύσματα, το μέγεθος των *codebooks*), ενώ το σχήμα κβαντισμού καθορίζει ποιά από τα διαθέσιμα για κάθε υποδιάνυσμα *codebooks* θα χρησιμοποιήσουμε στο κβαντισμό.

3.8.2 Προετοιμασία εκπαίδευσης

Και αυτό το βήμα αρκεί να γίνει μια φορά. Εδώ χρειάζεται να παραχθούν οι στατιστικές ιδιότητες του *feature* στο *train set*. Αφού τρέξει το *script* θα παραχθεί το καθορισμένο αρχείο στατιστικών ιδιοτήτων το οποίο θα διαβαστεί αργότερα από τον εκπαιδευτή. Το *script* για *cepstrum feature* είναι το **go.InitCepstrumTrainer**, ενώ για το *Single feature* είναι το **go.InitSingleTrainer**.

3.8.3 Διαδικασία εκπαίδευσης

Για κάθε σχήμα εκπαίδευσης που θέλουμε να δοκιμάσουμε, πρέπει να τρέξουμε τόσους *trainers*, όσος και ο αριθμός των υποδιανυσμάτων, παράγοντας τα αντίστοιχα *codebooks*.

Αφού καθοριστούν τα υποδιανύσματα, θα πρέπει να καθοριστεί και το μέγεθος του *codebook*, δηλαδή ο αριθμός των *centroids* που περιέχει το *codebook*. Αν για παράδειγμα χρειάζονται 64 *centroids*, θα γραφτούν για αυτό το υποδιάνυσμα τα *codebooks* με 1 έως 6 *bits*, έτσι ώστε αργότερα στο κβαντισμό να έχουμε διαθέσιμα όλα τα δυνατά *codebooks* και να μη ξαναχρειαστεί να παράγουμε το *codebook* με π.χ. 16 *centroids* αν αυτά χρειαστούν.

Η ονοματολογία των *codebooks* που γράφονται είναι η εξής :
`< codebook.id.numberOfBits >`, όπου *id* είναι το *id* του υποδιανύσματος όπως καθορίζεται στο *Coding* αρχείο. Έτσι για παράδειγμα, εάν πρόκειται να παραχθούν *codebooks* με 6 *bits* για το 3ο υποδιάνυσμα, τα αρχεία που θα παραχθούν θα είναι τα:

```
codebook.3.1
codebook.3.2
codebook.3.3
codebook.3.4
codebook.3.5
codebook.3.6
```

Το **Coding** αρχείο είναι αυτό που καθορίζει το σχήμα εκπαίδευσης και έχει την εξής μορφή :

```
2 1 2 3
2 3 4 3
3 5 6 7 3
3 8 9 10 3
3 11 12 13 3
```

το φORMατ είναι : <μέγεθος υποδιανύσματος> <στοιχεία του *feature* διανύσματος που το αποτελούν> <αριθμός *bits* > . Η γραμμή καθορίζει και το *id* του υποδιανύσματος. Έτσι π.χ. η γραμμή < 3 8 9 10 3 > δηλώνει ότι το υποδιάνυσμα 4 αποτελείται από τα εξής 3 στοιχεία του *feature* διανύσματος : 8 9 10, και το *codebook* που θα παραχθεί θα είναι 3 *bits*, δηλαδή θα περιέχει 8 *centroids*.

Εκτός από το παραπάνω αρχείο απαιτείται και κάποιο *configuration* αρχείο της μορφής :

| | |
|----------------------------------------|---------------------------------------------|
| <code>feature_type</code> | <code>cepstrum</code> |
| <code>input_files</code> | <code>trainerFiles</code> |
| <code>trainer_info_file</code> | <code>../.. /InitTrainer/ti.cepstrum</code> |
| <code>coding_configuration</code> | <code>Coding</code> |
| <code>codebook_path</code> | <code>codebooks/cepstrum/</code> |
| <code>e</code> | <code>0.01</code> |
| <code>repetitions</code> | <code>4</code> |
| <code>total_number_of_codebooks</code> | <code>5</code> |

Το *feature_type* καθορίζει τι είδους *feature* έχουμε(*Single, cepstrum*).

Το *input_files* είναι η λίστα με τα αρχεία του *train set*.

Το *trainer_info_file* είναι το αρχείο με τις στατιστικές ιδιότητες που αναφέρθηκε παραπάνω.

Το *coding_configuration* είναι το παραπάνω *Coding* αρχείο.

Το *codebook_path* καθορίζει που θα γραφτούν τα *codebooks*.

Το *e* είναι μια σταθερά που καθορίζει πόσο θα διαφέρουν τα 2 νέα *centroids* μετά το *splitting* του αρχικού.

Το *repetitions* είναι ο αριθμός επαναλήψεων στον αλγόριθμο εκπαίδευσης.

Το *total_number_of_codebooks* είναι ο συνολικός αριθμός των υποδιανυσμάτων.

Τα *scripts* εκπαίδευσης είναι τα **go.SingleTrainer** για *Single features* και το **go.CepstrumTrainer** για *cepstrum features*.

3.8.4 Διαδικασία κβαντισμού

Αφού παράγουμε τα *codebooks* σύμφωνα με τη παραπάνω διαδικασία, είμαστε έτοιμοι να προχωρήσουμε στο επόμενο βήμα του κβαντισμού. Ο κβαντιστής θα πρέπει να φορτώσει τα κατάλληλα *codebooks*. Αυτό καθορίζεται σε *configuration* αρχείο (καθορίζει το σχήμα κβαντισμού), που έχει τη μορφή :

| | |
|----------------------------------|---------------------------------------------|
| <code>input_files</code> | <code>in.cepstrums</code> |
| <code>output_files</code> | <code>out.coded.cepstrums</code> |
| <code>codebook_path</code> | <code>../trainer/codebooks/cepstrum/</code> |
| <code>number_of_codebooks</code> | <code>5</code> |
| <code>codebook1</code> | <code>3</code> |
| <code>codebook2</code> | <code>4</code> |
| <code>codebook3</code> | <code>5</code> |
| <code>codebook4</code> | <code>4</code> |
| <code>codebook5</code> | <code>4</code> |

Το `input_files` είναι η λίστα με τα αρχεία που θα κβαντιστούν (αυτά που φτιάξαμε στο 1ο βήμα)

Το `output_files` είναι η λίστα με τα ονόματα των κβαντισμένων αρχείων που θα παραχθούν

Το `codebook_path` είναι η τοποθεσία απ'όπου θα φορτωθούν τα `codebooks`

Οι επόμενες 5 γραμμές καθορίζουν τα `codebooks` που θα χρησιμοποιηθούν. Αυτά είναι με βάση το φορματ που καθορίστηκε παραπάνω τα :

```
codebook.1.3
codebook.2.4
codebook.3.5
codebook.4.4
codebook.5.4
```

Το `script` του κβαντισμού είναι το `go.Quantizer`

3.8.5 Έλεγχος με Αναγνώριση

Τα αρχεία που παράγονται από τον κβαντισμό δίνονται σαν είσοδος στον αναγνωριστή για να ελέγξουμε την επίδοση του σχήματος κωδικοποίησης που χρησιμοποιήσαμε. Συνήθως φροντίζουμε να παράγουμε `codebooks` με περισσότερα *bits*, απ'όσα συνήθως πιστεύουμε ότι ίσως είναι αρκετά. Με αυτό το τρόπο εξασφαλίζουμε ότι η επίδοση της αναγνώρισης θα είναι σε υψηλά επίπεδα. Έχοντας εξασφαλίσει το παραπάνω, πρέπει να αφαιρέσουμε όση πλεονάζουσα πληροφορία υπάρχει, χωρίς όμως να επιτρέπουμε μείωση της επίδοσης του αναγνωριστή. Αυτός άλλωστε είναι και ο στόχος μας. Για να αφαιρέσουμε τη πλεονάζουσα πληροφορία επιλέγουμε `codebooks` με λιγότερα *bits*. Έτσι για κάθε σχήμα κωδικοποίησης (σχήμα εκπαίδευσης + σχήμα κβαντισμού), εκτελούμε τόσους κβαντισμούς και αναγνωρίσεις όσους χρειάζονται για να επιτύχουμε το στόχο που τέθηκε παραπάνω.

Κάποια σχήματα κωδικοποίησης δίνουν μη ικανοποιητικά αποτελέσματα. Αυτό που πρέπει να γίνει τότε, είναι να καθοριστεί ένα νέο σχήμα κωδικοποίησης και να επαναληφθεί η διαδικασία κβαντισμού και αναγνώρισης για αυτό το σχήμα. Αν χρειαστεί να γίνουν πειράματα σε άλλα *train/test sets*, θα πρέπει να επαναληφθούν τα 2 πρώτα βήματα.

3.9 Αναγνώριση με το Decipher

Το σύστημα αναγνώρισης που χρησιμοποιήθηκε είναι αυτό του *Decipher*. Προκειμένου να γίνουν τα πειράματα έγιναν αλλαγές για να υποστηρίξουν τα νέα κβαντισμένα *features* που δημιουργήθηκαν με το *Yarrow*. Επίσης αφού όλα τα αρχεία που παράγονται με το *Yarrow* έχουν καθορισμένο φορματ, θα πρέπει το *Decipher* να μπορεί να τα διαβάσει και να τα αποκωδικοποιήσει. Επιπρόσθετα, στην περίπτωση που χρησιμοποιείται *index coded* κβαντισμός, θα πρέπει να γίνει διακριτοποίηση των μειγμάτων των κατανομών εξόδου των HMM μοντέλων. Σε αυτή τη περίπτωση, για κάθε σχήμα κωδικοποίησης χρειάζεται να ξαναγίνει η διακριτοποίηση των μειγμάτων έτσι ώστε να υπάρχει πάντα συνέπεια ανάμεσα στην είσοδο του συστήματος (*features*) και τα μοντέλα που χρησιμοποιούνται. Η διακριτοποίηση έγινε σε άλλη διπλωματική εργασία [23], ενώ στη παρούσα έγινε το απαραίτητο *interface* με το *Yarrow*. Για να υποστηρίζονται όλα τα παραπάνω, έγιναν οι απαραίτητες αλλαγές στην *front-end* βιβλιοθήκη του *Decipher*. Αυτές περιλαμβάνουν τα εξής :

- Επέκταση της *I/O* βιβλιοθήκης του *Decipher* για είσοδο από τα αρχεία που παράχθηκαν με το *Yarrow*.
- Δημιουργία ενός νέου *feature* για τη περίπτωση που έχουμε *index coded* κβαντισμό.

3.10 Συμπεράσματα

Σε αυτή την ενότητα παρουσιάστηκε η δουλειά που έγινε πάνω στο σύστημα *Yarrow*, τόσο σε επίπεδο κώδικα, όσο και πειραμάτων, καθώς και το *interface* με το σύστημα του *Decipher*. Δημιουργήθηκε τελικά μια πλατφόρμα που επιτρέπει την ερευνητική διαδικασία πάνω στο θέμα του Διανυσματικού Κβαντισμού των παραμέτρων της φωνής.

- Επεκτάθηκε το σύστημα ώστε να υποστηρίζει διανυσματικό κβαντισμό.
- Σε επίπεδο κώδικα, αναπτύχθηκε το *package sri.star.coding*, σαν μια κατάλληλα σχεδιασμένη ιεραρχία με πολλά επίπεδα το καθένα από τα

οποία επιτρέπει στον *developer* να κάνει αλλαγές χωρίς να επηρεάζεται η εικόνα του συστήματος από τον τελικό χρήστη. Το σύστημα είναι αρκετά ευέλικτο και επεκτάσιμο για μελλοντικές βελτιώσεις και προσθήκες.

- Σε επίπεδο πειραμάτων, σχεδιάστηκαν τα *configuration* αρχεία και τα *scripts* με τέτοιο τρόπο που να είναι εύκολο για τελικό χρήστη να τρέξει οποιοδήποτε πείραμα και να πειραματιστεί απλά αλλάζοντας κάποιες παραμέτρους.
- Έγιναν οι απαραίτητες αλλαγές στο *Decipher* σύστημα (επίσης τόσο σε επίπεδο κώδικα, όσο και πειραμάτων) για να επιτελεί αναγνώριση με τα παραγώμενα από το *Yarrow features*.
- Το *API* που αναπτύχθηκε, συνοδεύεται από τεκμηρίωση με πολλά σχόλια για κάθε κλάση χωριστά (σε μορφή *.html* σύμφωνα με τις συμβάσεις της *Java*), αλλά και μια σειρά από παραδείγματα πειραμάτων με επίσης όλα τα απαραίτητα σχόλια. Έτσι γίνεται εύκολο τόσο για τον *developer* να κατανοήσει και να επεκτείνει αν χρειαστεί τον κώδικα, αλλά και τον απλό χρήστη να τρέξει όποια πειράματα απαιτείται.

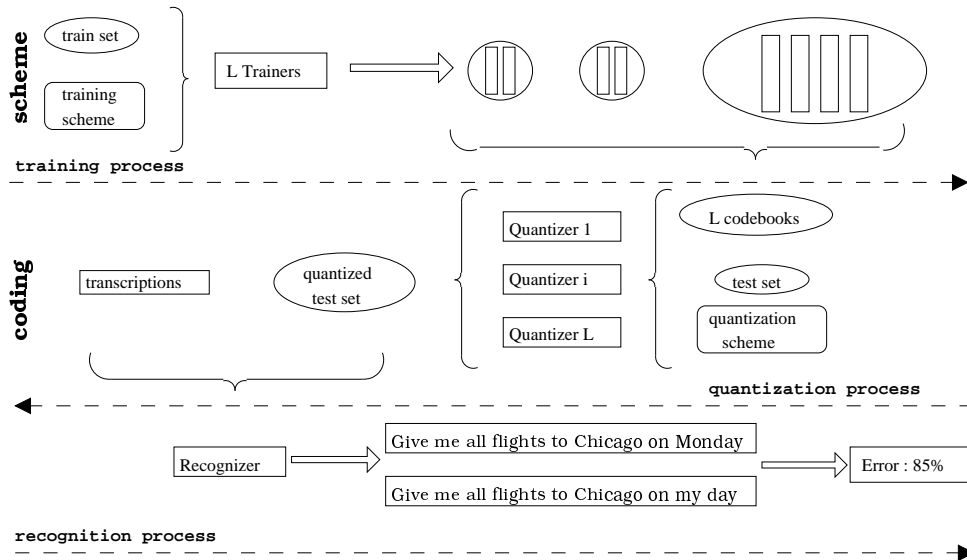
Κεφάλαιο 4

Πειράματα κωδικοποίησης

Στο Κεφάλαιο 2 αναλύσαμε το σχήμα κωδικοποίησης και τονίσαμε πως αποτελεί μέθοδο κωδικοποίησης φωνής προς αναγνώριση. Επισημάνθηκε επίσης ότι κάνουμε συνδυασμό τεχνικών κωδικοποίησης φωνής (επεξεργασία *front-end*) και τεχνικών κωδικοποίησης πηγής (διανυσματικός κβαντισμός) προκειμένου να μειώσουμε ακόμα περισσότερο τον ρυθμό μετάδοσης (ή ισοδύναμα να αυξήσουμε στο μέγιστο το βαθμό συμπίεσης), προσπαθώντας παράλληλα να κρατήσουμε υψηλά επίπεδα αναγνώρισης. Αναλύσαμε τεχνικές κβαντισμού με έμφαση στην τεχνική *product VQ* και είδαμε τις επιλογές που μπορεί να γίνουν κατά τον σχεδιασμό ενός συστήματος διανυσματικού κβαντισμού και τέλος παρουσιάσαμε τις διαδικασίες εκπαίδευσης και κβαντισμού για την κωδικοποίηση που προτείνουμε.

Στο Κεφάλαιο 3 μελετήσαμε την υλοποίηση της κωδικοποίησης που έγινε για το σύστημα *Yarrow* και είδαμε πως μπορούμε να το χρησιμοποιήσουμε για την εκπαίδευση και τον κβαντισμό. Είδαμε επίσης πως μπορούμε να το χρησιμοποιήσουμε με το σύστημα αναγνώρισης φωνής *Decipher* προκειμένου να αξιολογήσουμε ένα σχήμα κωδικοποίησης.

Σε αυτό το Κεφάλαιο θα δούμε αναλυτικά την διαδικασία των πειραμάτων και θα παρουσιάσουμε τα πειράματα που έγιναν, τα οποία αφορούν πειράματα τόσο βαθμωτού όσο και διανυσματικού κβαντισμού. Θα παραθέσουμε τον *bit allocation* αλγόριθμο που μας βοηθά στην εύρεση του επιτρεπόμενου ορίου συμπίεσης χωρίς απώλειες στην απόδοση αναγνώρισης και τέλος θα εξετάσουμε την επίδραση του θορύβου στην κωδικοποίηση. Αξίζει να σημειωθεί ότι στο Κεφάλαιο αυτό παρουσιάζονται τα 'τελικά' αποτελέσματα για τα οποία χρειάστηκε να γίνουν και πολλά άλλα πειράματα. Κάποια αυτά που έγιναν με τη χρήση του *bit allocation* αλγορίθμου παρουσιάζονται στο Παράρτημα στο τέλος αυτής της εργασίας.



Σχήμα 4.1: **Διαδικασία πειραμάτων.** Οι *trainers* παράγουν τα *L codebooks* (όσα και τα υποδιανύσματα). Κάθε ένας από τους *L* κβαντιστές χρησιμοποιεί το αντίστοιχο *codebook* για να κβαντίσει το υποδιάνυσμα σε κάθε *frame* στο *test set*. Για κάθε κβαντισμένο αρχείο, ο αναγνωριστής συγκρίνει το *transcription* (περιγραφή πρότασης) με το αποτέλεσμα της αναγνώρισης και υπολογίζει το συνολικό σφάλμα.

4.1 Διαδικασία πειραμάτων

Στις παραγράφους 2.9.2 και 2.9.3 μιλήσαμε για την διαδικασία εκπαίδευσης και κβαντισμού. Σε αυτή τη παράγραφο θα δούμε την συνολική εικόνα της διαδικασίας πειραμάτων, η οποία εκτός από τις δυο προηγούμενες εμπεριέχει και τη διαδικασία αναγνώρισης, με την οποία και αξιολογούμε την αποτελεσματικότητα του σχήματος κωδικοποίησης που εξετάζουμε κάθε φορά. Η γενική διαδικασία των πειραμάτων φαίνεται στο Σχήμα 4.1 και περιλαμβάνει τις διαδικασίες εκπαίδευσης, κβαντισμού και αναγνώρισης σε κάθε ένα από τα 3 επίπεδα που την αποτελούν.

Για την εκπαίδευση και τον κβαντισμό χρησιμοποιήσαμε το *Yarrow*, ενώ για την αναγνώριση χρησιμοποιούμε το σύστημα αναγνώρισης φωνής *Decipher* (τόσο το *Yarrow* όσο και το *Decipher* περιγράφονται στο Κεφάλαιο 4). Για την απόδοση της κωδικοποίησης, χρησιμοποιούμε σαν μέτρο, το σφάλμα της αναγνώρισης (*Word Error Rate* - *WER*), το οποίο ορίζεται ως :

$WER = \frac{INS+DEL+SUB}{TOTAL} \times 100$. Είναι δηλαδή ο αριθμός των λάθος εισαγωγών, Σαντικαταστάσεων και παραλείψεων λέξεων, σε σχέση με το συνολικό αριθμό λέξεων στην πρόταση.

Τα πειράματα έγιναν στο *ATIS corpus*, το οποίο έχει ένα λεξικό 1500 λέξεων και είναι μέτριας δυσκολίας. Το *train set* αποτελείται από 700 προτάσεις και το *test set* από 400 προτάσεις. Στα πειράματα που παρουσιάζουμε γίνεται χρήση *Centroid Coded* κβαντισμού με τα συνεχή γκαουσιανά ακουστικά μοντέλα του *ATIS*. Όπως έχουμε τονίσει και προηγούμενα, αυτός ο τρόπος δεν προσφέρει πραγματική συμπίεση (αφού η έξοδος του κβαντισμού αναπαρίσταται με το διάνυσμα και όχι το δείκτη), αλλά είναι αναγκαία για τη διαδικασία των πειραμάτων και την εύρεση των πιο αποδοτικών σχημάτων κωδικοποίησης. Για τα πιο επιτυχημένα τέτοια σχήματα κωδικοποίησης, προχωρούμε στη διακριτοποίηση των ακουστικών μοντέλων (βλέπε παράγραφο 1.5.3), τα οποία προσφέρουν μεγάλη συμπίεση και γρηγορότερη αναγνώριση και θα χρησιμοποιούνται στη πράξη, σε εφαρμογές στηριζόμενες σε αυτή την εργασία.

Η γενική διαδικασία των πειραμάτων είναι η εξής :

- Στη διαδικασία εκπαίδευσης, χρησιμοποιούμε το *train set* και το δοσμένο σχήμα εκπαίδευσης (*trainig scheme*), με το οποίο καθορίζουμε πόσα και ποια υποδιανύσματα θέλουμε, καθώς και πόσα *bits* θα διαθέσουμε για την εκπαίδευσή τους για να παράγουμε τελικά τα *codebooks*.
- Στο σχήμα κβαντισμού, χρησιμοποιούμε το *test set* και το δοσμένο σχήμα κβαντισμού (*quantization scheme*), με το οποίο δηλώνουμε ποιά από αυτά τα *codebooks* θα χρησιμοποιήσουμε για να κάνουμε το κβαντισμό και ο κβαντιστής παράγει τα κβαντισμένα αρχεία που αντιστοιχούν σε αυτά του *test set*.
- Με τη διαδικασία αναγνώρισης, τα κβαντισμένα αυτά αρχεία περνάνε σαν είσοδος στον αναγνωριστή. Τα αποτελέσματα της αναγνώρισης συγκρίνονται με την περιγραφές των αρχείων και έτσι προκύπτει το ποσοστό λάθους της αναγνώρισης, το οποίο μας δείχνει κατά πόσο το σχήμα κωδικοποίησης (*coding scheme*) ήταν επιτυχές.

Στα πειράματα που ακολουθούν, αρχίζουμε με το *baseline* πείραμα, το οποίο χρησιμοποιεί το *test-set* (μη κβαντισμένα αρχεία) και το οποίο καθορίζει την ελάχιστη απόδοση αναγνώρισης που θα δεχόμαστε για τα πειράματα με κβαντισμό. Στα πειράματα αυτά, χρησιμοποιούμε τεχνικές βαθμωτού (ομοιόμορφου και μη) κβαντισμού, καθώς και τεχνικές *VQ* βασισμένες στη χρήση υποδιανυσμάτων (*product VQ*). Τέλος θα παρουσιάσουμε την χρήση του *bit-allocation* αλγορίθμου και πειράματα με θόρυβο.

4.2 Baseline πείραμα

Το *baseline* είναι το βασικό πείραμα που δείχνει την απόδοση για μη κβαντισμένα αρχεία. Σε όλα τα επόμενα πειράματα κβαντισμού, στόχος μας θα είναι να προσεγγίσουμε επίπεδα αναγνώρισης κοντά σε αυτά του *baseline* πειράματος, κάτι που θα αποδεικνύει ότι ο θόρυβος που εισαγάγει ο κβαντισμός είναι αρκετά μικρός ή δεν θα επηρεάζει αισθητά την διαδικασία της αναγνώρισης. Για τη διεξαγωγή του, χρησιμοποιήθηκαν αρχικά τηλεφωνικής ποιότητας (*toll quality*) δεδομένα, τόσο σε *PCM* κωδικοποίηση, όσο και σε *GSM*. Παρατηρούμε ότι αν και η διαφορά στο *bit-rate* είναι αρκετά μεγάλη, η διαφορά στην ακρίβεια αναγνώρισης είναι σχετικά μικρή. Το *baseline* μας ενδιαφέρει κυρίως για υψηλής ποιότητας (*high quality*) δεδομένα όπου η ακρίβεια αναγνώρισης ήταν **6.55**, για μεγαλύτερο όμως *bit-rate*.

| Type of Encoding | Bit-rate (kbps) | Word-error Rate(%) |
|------------------------|-----------------|--------------------|
| PCM toll quality(Mlaw) | 64 | 12.7 |
| GSM encoding | 13 | 14.5 |
| PCM high quality | 256 | 6.55 |

Πίνακας 4.1: **Baseline πείραμα.** PCM(high & toll quality) & GSM encoding

4.3 Πειράματα βαθμωτού κβαντισμού

Αρχικά κβαντίσαμε τους συντελεστές τηλεφωνικής ποιότητας φωνής και εξετάσαμε την απόδοση για διαφορετικούς αριθμούς *bits/element*. Χρησιμοποιήσαμε τόσο ομοιόμορφο όσο και μη-ομοιόμορφο κβαντισμό. Στη περίπτωση του μη-ομοιόμορφου κβαντισμού, η απόσταση μεταξύ των σταθμών, προέκυψε από την κατανομή των συντελεστών του διανύσματος στο *train set*. Τα αποτελέσματα συνοψίζονται στο πίνακα 4.2.

Παρατηρούμε ότι για 5-8 *bits/element* η απόδοση της αναγνώρισης είναι αρκετά καλή, και στις 2 περιπτώσεις. Από εκεί και πέρα τα αποτελέσματα αλλάζουν αρκετά. Συγκεκριμένα, για τον ομοιόμορφο κβαντισμό βλέπουμε ότι μειώνοντας έστω και κατά 1 *bit/element*, η απόδοση πέφτει αμέσως στο 17.43 που ήδη είναι μη αποδεκτό, από άποψη αναγνώρισης. Αυτή η συμπεριφορά των αποτελεσμάτων σημαίνει ότι :

1. Κανένας συντελεστής δεν χρειάζεται περισσότερα από 5 *bits*.
2. Στον ομοιόμορφο κβαντισμό, χρησιμοποιώντας λιγότερα από 5 *bits/element*,

| Bits/coef. | Bit-Rate (kbps) | Uniform (WER%) | Non-Uniform (WER%) |
|------------|-----------------|----------------|--------------------|
| 8 | 7.2 | 12.55 | 12.82 |
| 7 | 6.3 | 12.65 | 12.87 |
| 6 | 5.4 | 13.08 | 12.65 |
| 5 | 4.5 | 13.14 | 13.62 |
| 4 | 3.6 | 17.43 | 13.19 |
| 3 | 2.7 | 45.47 | 14.64 |
| 2 | 1.8 | 108.9 | 21.07 |

Πίνακας 4.2: **Bit-rates & WER για scalar quantization των συντελεστών του MFCC διανύσματος για τηλεφωνικής ποιότητας φωνή.** (Για τηλεφωνικής ποιότητας φωνή, το *MFCC* διάνυσμα έχει μέγεθος 9 : 8 *cepstral* συντελεστές και 1 συντελεστής για την ενέργεια)

| Bits/coef. | Bit Rate (kbps) | Uniform (WER%) | Non-Uniform (WER%) |
|------------|-----------------|----------------|--------------------|
| 8 | 10.4 | 6.65 | 6.53 |
| 7 | 9.1 | 6.76 | 6.40 |
| 6 | 7.8 | 6.65 | 6.43 |
| 5 | 6.5 | 6.96 | 6.32 |
| 4 | 5.2 | 6.96 | 6.32 |
| 3 | 3.9 | 12.45 | 6.88 |
| 2 | 2.6 | 95.43 | 9.04 |

Πίνακας 4.3: **Bit rates & WER για scalar quantization των συντελεστών του MFCC διανύσματος για υψηλής ποιότητας φωνή.** (Για υψηλής ποιότητας φωνή, το *MFCC* διάνυσμα έχει μέγεθος 13 : 12 *cepstral* συντελεστές και 1 συντελεστής για την ενέργεια)

αφαιρούμε πληροφορία από ποιο σημαντικά *elements*, ενώ ίσως δίνουμε περισσότερα *bits* από ότι χρειάζεται σε άλλα *elements*. Αυτό έχει σαν αποτέλεσμα την ραγδαία αύξηση του *WER*.

3. Αντίθετα, στην περίπτωση του μη-ομοιόμορφου κβαντισμού, βλέπουμε ότι ακόμα και στα 2.7 Kbps η απόδοση δεν πέφτει σημαντικά. Μάλιστα, είναι σχεδόν ίση με την απόδοση του *GSM* συστήματος το οποίο έχει ρυθμό μετάδοσης της τάξης των 13 Kbps.

Στη συνέχεια χρησιμοποιήσαμε υψηλής ποιότητας φωνή και διεξάγαμε ανάλογα πειράματα (βλέπε Πίνακα 4.3). Παρατηρούμε πάλι, ότι στην περίπτωση του μη-ομοιόμορφου κβαντισμού έχουμε καλλίτερα αποτελέσματα. Χρησι-

μοποιώντας 5.2 & 3.9 kbps αντίστοιχα για ομοιόμορφο και μη ομοιόμορφο κβαντισμό, υπερβαίνουμε κατά λίγο το *WER* στο *baseline* πείραμα (6.55%). Συμπερασματικά, βλέπουμε ότι χρησιμοποιώντας ένα σχετικά απλό σχήμα παίρνουμε πολύ καλά αποτελέσματα.

4.4 Πειράματα διανυσματικού κβαντισμού

Είδαμε στην παράγραφο 2.5.2, ότι η *product VQ* τεχνική, μπορεί να αποδειχθεί αρκετά αποτελεσματική, εφόσον επιλεγούν κατάλληλα οι παράμετροι αυτής της τεχνικής, που είναι ο καθορισμός των υποδιανυσμάτων και ο αριθμός *bits* που θα διαθέσουμε σε κάθε ένα από αυτά.

4.4.1 Επιλογή παραμέτρων συστήματος

Ο αριθμός των συνδυασμών που προκύπτει από την επιλογή των υποδιανυσμάτων στα οποία μπορούμε να χωρίσουμε το *MFCC* διάνυσμα και τον αριθμό των *centroids* ανά υποδιάνυσμα, είναι απαγορευτικά μεγάλος. Έτσι, έπρεπε να βρεθεί ένας συστηματικός τρόπος επιλογής των παραμέτρων.

4.4.2 Επιλογή υποδιανυσμάτων

Για την επιλογή των υποδιανυσμάτων υπήρξαν δυο προσεγγίσεις :

Η πρώτη προσέγγιση είναι να ομαδοποιήσουμε τα στοιχεία του *MFCC* διανύσματος με τέτοιο τρόπο, ώστε αυτά με τη μεγαλύτερη στατιστική συσχέτιση, να ομαδοποιηθούν στο ίδιο υποδιάνυσμα. Η προσέγγιση αυτή δεν έδωσε τα αναμενόμενα αποτελέσματα, καθώς αποδείχθηκε ότι οι συσχετίσεις μεταξύ των *elements* κυμαίνονταν σε πολύ μικρές τιμές. Αυτό επιβεβαιώνεται από τη χρήση *DCT* στο *front-end* (βλέπε Σχήμα 1.2).

Η δεύτερη προσέγγιση βασίστηκε στο γεγονός, ότι κάποια στοιχεία του διανύσματος είναι πιο σημαντικά από κάποια άλλα και επομένως ο αριθμός των *bits* που πρέπει να διαθέσουμε κυμαίνεται αρκετά, ανάλογα με τη σημαντικότητα του στοιχείου. Συγκεκριμένα, προχωρώντας από τα πρώτα στα τελευταία στοιχεία του διανύσματος, ο λόγος *bits/elements* μειώνεται δραστικά, κάτι το οποίο οφείλεται στη χρήση της *Mel-frequency* κλίμακας στο *front-end* και έχει σαν αποτέλεσμα την συμπίεση της πληροφορίας στα πρώτα στοιχεία του διανύσματος (βλέπε επίσης Σχήμα 1.2). Αυτό σημαίνει ότι προκειμένου όλα τα υποδιανύσματα να ‘περιέχουν’ ίσο ποσό πληροφορίας, θα πρέπει να ομαδοποιήσουμε τα πιο σημαντικά από αυτά σε μικρά υποδιανύσματα και τα λιγότερο σημαντικά σε μεγαλύτερα υποδιανύσματα. Τα αποτελέσματα για τις δυο προσεγγίσεις φαίνονται στον Πίνακα 4.4.

| Bit Rate (bps) | Correlation based | Knowledge based |
|----------------|-------------------|-----------------|
| 1400 | 18.77 | 11.71 |
| 1600 | 13.36 | 9.30 |
| 1800 | 10.24 | 8.10 |
| 1900 | 8.92 | 6.99 |
| 2000 | 8.38 | 6.63 |
| 2100 | 7.72 | - |
| 2200 | 7.01 | - |

Πίνακας 4.4: Σύγκριση **correlation & knowledge based partitioning**.

Στην περίπτωση χρήσης συσχέτισης για την επιλογή των υποδιανυσμάτων, τα αποτελέσματα είναι σαφώς μη ικανοποιητικά. Τα υποδιανύσματα που χρησιμοποιήθηκαν είναι τα : (1,5), (3,9,12,13), (4,6), (2,7,11), (8,10). Στη δεύτερη περίπτωση αντίθετα, όπου το κριτήριο είναι η σημαντικότητα των στοιχείων, τα αποτελέσματα είναι σαφώς καλλίτερα. Σε αυτή τη περίπτωση, τα υποδιανύσματα που χρησιμοποιήθηκαν είναι τα : (1,2), (3,4), (5,6,7), (8,9,10), (11,12,13).

4.4.3 Επιλογή αριθμού bits ανά υποδιάνυσμα : bit allocation αλγόριθμος

Η δεύτερη παράμετρος που πρέπει να καθοριστεί είναι ο αριθμός των *centroids* ανά υποδιάνυσμα. Η διαδικασία που ακολουθήσαμε βασίζεται στον *bit-allocation* αλγόριθμο και είναι η εξής :

Αρχικοποίηση : Διάθεση ενός αρχικού αριθμού *bits* στα υποδιανύσματα και αξιολόγηση της αναγνώρισης. Αυτό είναι και το αρχικό σχήμα.

Βήμα 1 : Για κάθε υποδιάνυσμα, αύξηση κατά ένα του αριθμού *bits* και νέα αξιολόγηση της αναγνώρισης, κρατώντας τον αριθμό των *bits* που έχουμε διαθέσει στα υπόλοιπα υποδιανύσματα ίσο με αυτό στο τρέχον σχήμα. Το επιπλέον *bit* το διαθέτουμε τελικά στο υποδιάνυσμα το οποίο προσέφερε την μεγαλύτερη αύξηση στην απόδοση της αναγνώρισης και θέτουμε αυτό, σαν το νέο σχήμα.

Βήμα 2 : Αν έχει επιτευχθεί το επιθυμητό αποτέλεσμα αναγνώρισης ή αν φτάσαμε το μέγιστο διαθέσιμο αριθμό *bits* σταματάμε, αλλιώς συνεχίζουμε τη διαδικασία από το Βήμα 1.

Ο ίδιος αυτός αλγόριθμος μπορεί να χρησιμοποιηθεί στην περίπτωση του μη ομοιόμορφου βαθμωτού κβαντισμού, αφού αυτός αποτελεί ειδική περίπτωση του διανυσματικού κβαντισμού, με υποδιανύσματα μήκους ενός συντελεστή. Τα αποτελέσματα φαίνονται στους πίνακες 4.5 και 4.6 αντίστοιχα.

| | Composition of subVectors | | | | | | |
|------------|------------------------------|----------|----------|----------|----------|-------------|-------------|
| | 1,2 | 3,4 | 5,6,7 | 8,9,10 | 11,12,13 | | |
| Total bits | subvector bits per iteration | | | | | Bit Rate | WER(%) |
| 12 | 3 | 3 | 2 | 2 | 2 | 1200 | 16.76 |
| 14 | 5 | 3 | 2 | 2 | 2 | 1400 | 11.71 |
| 16 | 5 | 3 | 4 | 2 | 2 | 1600 | 9.30 |
| 18 | 5 | 3 | 4 | 4 | 2 | 1800 | 8.10 |
| 19 | 5 | 4 | 4 | 4 | 2 | 1900 | 6.99 |
| 20 | 5 | 4 | 4 | 4 | 2 | 2000 | 6.63 |

Πίνακας 4.5: **Bit allocation για τα 5 υποδιανύσματα.** Όπως φαίνεται καθαρά, τα υποδιανύσματα 1 & 2 είναι τα πιο σημαντικά αν και μικρότερα. Ακολουθούν τα 3 & 4, ενώ το υποδιάνυσμα 5 φέρει την λιγότερη πληροφορία. Παρατηρούμε επίσης ότι με μόλις *2kbps* διατηρούμε ίδιο *WER* με την περίπτωση όπου δεν έχουμε καν κβαντισμό!

| | <i>MFCC</i> | | | | | | | | | | | | | | |
|-------------------|----------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------------|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | | |
| <i>Total bits</i> | element bits per iteration | | | | | | | | | | | | | Bit Rate | <i>WER</i> (%) |
| 17 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1700 | 12.78 |
| 18 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1800 | 10.66 |
| 20 | 3 | 3 | 2 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2000 | 8.69 |
| 22 | 3 | 3 | 3 | 3 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2200 | 7.67 |
| 24 | 3 | 3 | 3 | 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2400 | 6.69 |
| 26 | 3 | 3 | 3 | 3 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 2600 | 6.81 |
| 28 | 3 | 3 | 3 | 3 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2800 | 6.71 |
| 30 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3000 | 6.55 |

Πίνακας 4.6: **Bit allocation για τη scalar περίπτωση** Φαίνεται καθαρά πως οι 4 πρώτοι συντελεστές είναι οι πιο σημαντικοί (3 *bits*) ενώ οι υπόλοιποι είναι λιγότερο σημαντικοί και απαιτούν μόνο 2 συντελεστές. Παρατηρούμε πως ήδη από τα 2600 *bits*, έχουμε καταφέρει να προσεγγίσουμε την απόδοση του *baseline* πειράματος. Παρατηρούμε επίσης πως σε σχέση με το προηγούμενο πίνακα, χρειάζονται 10 *bits* παραπάνω (3*kbps*) για να φτάσουμε την ίδια απόδοση στην αναγνώριση.

| Test SNR(db) | Train SNR(db) | VQ Encoding bi-ts/subvector | WER(%) |
|--------------|---------------|-----------------------------|--------|
| Clean | Clean | No encoding | 6.55 |
| Clean | Clean | 5 5 4 4 2 | 6.63 |
| 24 | Clean | No encoding | 8.51 |
| 24 | Clean | 5 5 4 4 2 | 12.19 |
| 24 | 24 | 5 5 4 4 2 | 11.89 |
| 24 | Clean | 5 5 5 4 2 | 11.18 |
| 24 | Clean | 6 5 5 4 2 | 10.49 |
| 24 | Clean | 6 6 5 4 3 | 9.47 |
| 24 | Clean | 7 6 5 4 4 | 9.32 |
| 24 | Clean | 7 6 5 4 5 | 8.94 |

Πίνακας 4.7: Κβαντισμός παρουσία θορύβου.

4.4.4 Πειράματα με θόρυβο.

Στην εφαρμογή της μεθόδου που ακολουθήσαμε έως τώρα, δεν υπολογίσαμε την επίδραση του θορύβου. Για να διαπιστώσουμε την επίδραση του θορύβου, κάναμε το εξής πείραμα :Προστέθηκε θόρυβος στο *test set* έτσι ώστε ο λόγος σήματος/θορύβου (*SNR*) να φτάσει τη τιμή των 24 dB. Ακολουθώντας το *bit allocation* σχήμα για άλλη μια φορά, προσπαθήσαμε να βρούμε το κατάλληλο αριθμό *bits* που απαιτούνται για να φτάσουμε τα επίπεδα αναγνώρισης στη περίπτωση του μη κβαντισμού. Τα αποτελέσματα φαίνονται στον Πίνακα 2.7.

Αναλυτικότερα, στη 1η και 2η γραμμή επαναλαμβάνονται τα αποτελέσματα που πήραμε πριν την εισαγωγή του θορύβου για τα πειράματα χωρίς κβαντισμό (*baseline*) και για το καλλίτερο πείραμα με χρήση κβαντισμού. Όλα τα επόμενα πειράματα είναι παρουσία θορύβου. Στην 3η και 4η γραμμή επαναλαμβάνουμε τα 2 παραπάνω πειράματα . Το σφάλμα αναγνώρισης χωρίς κβαντισμό είναι 8.51% ενώ με κβαντισμό αυξάνει ακόμα περισσότερο φτάνοντας το 12.19%. Αν προσθέσουμε 24 dB στο *train set* έχουμε μικρή μείωση του σφάλματος (11.89%). Στα επόμενα πειράματα εφαρμόζουμε το *bit-allocation* σχήμα με σκοπό να προσεγγίσουμε το 8.51%. Στις υπόλοιπες γραμμές φαίνεται πως μειώνεται το *WER* στο σε 8.94 προσθέτοντας συνολικά 7 *bits* παραπάνω, φτάνοντας έτσι στα 27 *bits*, δηλαδή 2.7kbps.

4.5 Συμπεράσματα

- Η διαδικασία πειραμάτων αποτελείται από τις διαδικασίες εκπαίδευσης, κβαντισμού και αναγνώρισης. Τα αποτελέσματα της αναγνώρισης (έχου-

με κωδικοποίηση φωνής προς αναγνώριση) είναι αυτά που κρίνουν την ποιότητα της κωδικοποίησης.

- Το *baseline* πείραμα μας δίνει αποτελέσματα αναγνώρισης χωρίς κβαντισμό, τα οποία και θα προσπαθήσουμε να προσεγγίσουμε και στα πειράματα της κωδικοποίησης.
- Ο βαθμωτός κβαντισμός αν και απλός στην υλοποίηση δίνει αρκετά καλά αποτελέσματα. Ο μη-ομοιόμορφος υπερέρχει του ομοιόμορφου βαθμωτού κβαντισμού.
- Για τον διανυσματικό κβαντισμό πρέπει να κάνουμε επιλογές σχετικά με το πως θα χωρίσουμε το *MFCC* διάνυσμα σε μικρότερα υποδιανύσματα και με το πόσα *bits* θα διαθέσουμε για την εκπαίδευσή τους. Για την πρώτη επιλογή χρησιμοποιήσαμε την γνώση μας σχετικά με την σημαντικότητα των συντελεστών του *MFCC* διανύσματος ενώ για την δεύτερη τον *bit-allocation* αλγόριθμο.
- Τα αποτελέσματα των πειραμάτων διανυσματικού κβαντισμού είναι εξαιρετικά. Με *bit-rate* μόλις **2Kbps** παίρνουμε ακρίβεια αναγνώρισης ίση, με την περίπτωση του μη κβαντισμού (*baseline* πείραμα), ενώ και στη περίπτωση θορύβου, το σύστημα αποδείχθηκε αρκετά εύρωστο απαιτώντας επιπρόσθετα μόλις 0.7Kbps. Τα αποτελέσματα αυτά δημοσιεύτηκαν στα [10], [11], και [12].

Κεφάλαιο 5

Αναγνώριση σε ασύρματα και ενσύρματα δίκτυα δεδομένων

5.1 Εισαγωγή

Στο Εισαγωγικό κεφάλαιο αναφερθήκαμε στις ραγδαίες εξελίξεις στο χώρο των δικτύων και τον ασύρματων επικοινωνιών και προτείναμε τη χρησιμοποίηση αυτών των νέων τεχνολογιών για την περαιτέρω εξάπλωση εφαρμογών αναγνώρισης φωνής. Στο πρώτο μέρος αυτού του κεφαλαίου θα εξετάσουμε διεξοδικά τις αναμενόμενες εξελίξεις στους τομείς του διαδικτύου και των ασύρματων επικοινωνιών. Θα δούμε πως μπορούμε να χρησιμοποιήσουμε εφαρμογές αναγνώρισης φωνής σε αυτά τα περιβάλλοντα, βασιζόμενοι στην κωδικοποίηση που αναπτύξαμε σε αυτή την εργασία (βλέπε Κεφάλαιο 2). Η κωδικοποίηση που αναπτύξαμε παίζει καταλυτικό παράγοντα, μειώνοντας στο ελάχιστο τις απαιτήσεις εύρους ζώνης και καθιστώντας δυνατή και με μικρό κόστος την ανάπτυξη αντίστοιχων εφαρμογών στα παραπάνω περιβάλλοντα.

Αναφέραμε επίσης στο Εισαγωγικό κεφάλαιο, ότι τέτοιες εφαρμογές εκ των πραγμάτων θα χρησιμοποιούν το μοντέλο πελάτη - εξυπηρετητή για αναγνώριση φωνής. Στο δεύτερο μέρος του κεφαλαίου, περιγράφεται μια τέτοια υλοποίηση του πελάτη (*Speech Recognition Applet Demo*) για αναγνώριση φωνής στο διαδίκτυο μέσα από ένα δίαυλο εύρους *2kbps*, με τον πελάτη να χρησιμοποιεί το σύστημα *Yarrow* που εξετάσαμε στο Κεφάλαιο 3, και τον εξυπηρετητή το σύστημα αναγνώρισης φωνής *Decipher* κατάλληλα αλλαγμένο για συνεργασία με το *Yarrow*. Επίσης Θα εξετάσουμε την αρχιτεκτονική του πελάτη και τον τρόπο επικοινωνίας με τον αναγνωριστή. Θα αρχίσουμε αυτό το κεφάλαιο εξετάζοντας πρώτα τις εφαρμογές αναγνώρισης φωνής που κυριαρχούν σήμερα.

5.2 Εφαρμογές αναγνώρισης φωνής σήμερα

Οι εφαρμογές αναγνώρισης φωνής χωρίζονται σε 5 κατηγορίες :

- Εφαρμογές γραφείου όπως συστήματα υπαγόρευσης, εισαγωγή δεδομένων.
- Κατασκευές. Παρακολούθηση της κατασκευαστικής διαδικασίας και έλεγχος ποιότητας.
- Έλεγχος με φωνή, συστήματα για άτομα με ειδικές ανάγκες, εφαρμογές όπως η ρύθμιση του κλιματισμού ή του συστήματος ήχου σε ένα αυτοκίνητο.
- Εφαρμογές στην ιατρική όπως για παράδειγμα έλεγχος ρομποτικού βραχίονα για επεμβάσεις από απόσταση.
- Εφαρμογές στις τηλεπικοινωνίες και τηλεφωνικά κέντρα. Αυτοματοποίηση χειρισμού τηλεφωνικών κέντρων, εύρεση πληροφοριών τηλεφωνικού καταλόγου, τελεμάρκετινγκ.

Οι εφαρμογές αναγνώρισης φωνής πρέπει να παρέχουν πραγματικά οφέλη στο χρήστη, με τη μορφή αυξημένης παραγωγικότητας, ευκολίας χρήσης και ενός πιο φυσικού τρόπου επικοινωνίας. Ο χρήστης πρέπει να νοιώθει άνετα μέσα από το διάλογο με το σύστημα το οποίο θα πρέπει να είναι αρκετά ευέλικτο, προσφέροντας βοήθεια στο χρήστη και ένα αποτελεσματικό τρόπο επικοινωνίας. Επίσης η υψηλή ακρίβεια αναγνώρισης και η λειτουργία σε πραγματικό χρόνο αποτελούν απαραίτητα συστατικά επιτυχίας μιας εφαρμογής αναγνώρισης φωνής.

Την μεγαλύτερη απήχηση από τις παραπάνω, έχουν οι εφαρμογές αναγνώρισης φωνής για ανάκτηση πληροφοριών με τη χρήση τηλεφώνου. Η χρήση μέσω ενός απλού τηλεφώνου κάνει τις υπηρεσίες αυτές προσβάσιμες σε οποιονδήποτε. Οι πληροφορίες αφορούν εύρεση εισιτηρίων για αεροπορικές πτήσεις, αγορά προϊόντων, χρηματοοικονομικά μεγέθη και άλλες. Στις Ηνωμένες Πολιτείες τέτοιες εφαρμογές δέχονται καθημερινά αρκετές δεκάδες χιλιάδων τηλεφωνικών κλήσεων. Τέτοιες υπηρεσίες είναι βέβαια προσβάσιμες και από κινητά τηλέφωνα κάτι που αποτελεί μεγάλο πλεονέκτημα καθώς κάνει την πρόσβαση σε αυτές τις υπηρεσίες προσβάσιμη από οπουδήποτε.

Προκειμένου να γίνει εκμετάλλευση του πλούτου των πληροφοριών που υπάρχει στο διαδίκτυο, έχει αναπτυχθεί το πρότυπο της *VoiceXML*. Είναι μια τεχνολογία που βασίζεται στην *XML* και που θα επιτρέψει να γίνει προσβάσιμο το περιεχόμενο δικτυακών τόπων μέσα από το τηλέφωνο. Τέτοιο δικτυακοί

τόποι θα ονομάζονται *voice portals* (φωνητικές πύλες) και αναμένεται ότι η νέα αυτή αγορά που δημιουργείται, θα έχει μεγάλη ανάπτυξη τα προσεχή χρόνια.

5.3 Εξελίξεις στο χώρο των επικοινωνιών : δίκτυα δεδομένων

Ο χώρος του διαδικτύου αποτελεί ένα ώριμο πια τεχνολογικά χώρο που πραγματικά έχει αλλάξει τη ζωή μας σε τεχνολογικό, οικονομικό και κοινωνικό επίπεδο. Οι δυνατότητες επικοινωνίας που προσφέρει, καθώς και η γρήγορη πρόσβαση σε πληροφορίες, αποτελούν τα πιο δυνατά χαρακτηριστικά του. Πώς μπορούν οι εφαρμογές αναγνώρισης φωνής να εκμεταλλευτούν αυτή τη τεχνολογία; Είδαμε παραπάνω ένα τέτοιο παράδειγμα με τη χρήση της τεχνολογίας *VoiceXML* μέσω τηλεφωνικού δικτύου (σταθερού ή ασύρματου). Το χαρακτηριστικό αυτών των προσπαθειών, είναι η πρόσβαση μέσω δικτύου φωνής.

Εδώ θα παρουσιάσουμε άλλες προσπάθειες εφαρμογής αναγνώρισης φωνής, που βασίζονται σε δίκτυα δεδομένων, τόσο μέσα από ενσύρματες όσο και ασύρματες συνδέσεις. Ο διαχωρισμός αυτός μεταξύ δικτύων δεδομένων και φωνής συνεχίζει να υφίσταται παρ'όλες τις προσπάθειες για σύγκλιση των δυο αυτών τρόπων επικοινωνίας. Και ενώ τα δίκτυα φωνής, συνεχίζουν να κυριαρχούν στη επικοινωνία μεταξύ ανθρώπων¹, για την επικοινωνία ανθρώπου-μηχανής (παράδειγμα η αναζήτηση πληροφοριών) τα δίκτυα δεδομένων υπερτερούν.

5.4 Ασύρματα δίκτυα νέας γενεάς

Η ταχεία ανάπτυξη των ασύρματων δικτύων και η εξάπλωση τις κινητής τηλεφωνίας, μαζί με την μεγάλη ανάπτυξη που υφίσταται η αγορά *embedded* συστημάτων, *consumer electronics* και μιας πληθώρας άλλων συσκευών από *palmtops* και *PDA's* έως *smartphones*, δείχνει ότι οδεύουμε στην μετά *PC* εποχή. Αν στην εποχή των *mainframes* είχαμε πολλούς ανθρώπους να μοιράζονται ένα υπολογιστικό σύστημα και στην εποχή του *PC* την δυνατότητα ο κάθε άνθρωπος να έχει τον δικό του υπολογιστή, σήμερα οδεύουμε στην εποχή που η αναζήτηση και επεξεργασία πληροφοριών, θα γίνεται από πολλές διαφορετικού τύπου τέτοιες συσκευές, τις οποίες πιθανότατα θα κατέχουν πολλοί από εμάς. Και αν η πρόσβαση στο διαδίκτυο από τον προσωπικό υπολογιστή έχει γίνει τόσο εύκολη και απαραίτητη, γιατί να μην ισχύει το ίδιο και για αυτές τις συσκευές, που ήδη σήμερα θεωρούνται απαραίτητα εργαλεία, όπως

¹Για πόσο ακόμα δεν είναι γνωστό, καθώς με τεχνολογίες όπως *Voice Over IP (VoIP)* σύντομα αναμένεται η χρήση δικτύων δεδομένων ακόμα και για αυτό το τρόπο επικοινωνίας

ένα *PDA* ή ένα κινητό τηλέφωνο; Μέχρι το τέλος αυτής της παραγράφου, θα επικεντρώσουμε την ανάλυσή μας στην περιοχή της κινητής τηλεφωνίας, αφού αυτή αποτελεί το πιο δημοφιλές και οικείο παράδειγμα, χωρίς να δοθεί όμως η εντύπωση ότι οι εξελίξεις αφορούν αποκλειστικά σε αυτή τη περιοχή.

Σύμφωνα λοιπόν με τελευταίες έρευνες της *Forrester Research*, μέχρι το 2003 θα υπάρχουν 1 δις. κινητά τηλέφωνα σε δίκτυα *GSM* 3ης γενεάς, αριθμός πολύ μεγαλύτερος των μέχρι τότε αναμενόμενων προσωπικών υπολογιστών που θα είναι διασυνδεδεμένοι στο δίκτυο! Μιλάμε λοιπόν για το κινητό διαδίκτυο (*mobile internet*) του αύριο, στο οποίο καθοριστικό ρόλο θα παίζει η υποστήριξη ενός γρήγορου και αξιόπιστου ασύρματου δικτύου δεδομένων. Στη σημερινή κινητή τηλεφωνία, ήδη υπάρχει υποστήριξη δεδομένων (σε πολύ μικρές ταχύτητες) και αντίστοιχες υπηρεσίες (*SMS*, *WAP*). Στο δρόμο προς τη κινητή τηλεφωνία 3ης γενεάς θα μας οδηγήσουν πρωτόκολλα όπως τα *GPRS*, *EDGE*, *WCDMA* με βασικό χαρακτηριστικό όλων, την πλήρη υποστήριξη δικτύων δεδομένων *IP* και *X25* σε υψηλές ταχύτητες (*broadband networks*).

5.4.1 Οφέλη για παροχείς δικτύων, εφαρμογών και τελικό χρήστη

Ήδη το *GPRS* (βλέπε [18]), θα τεθεί σε εφαρμογή στις αρχές του 2001 και οι παροχείς κινητής τηλεφωνίας πασχίζουν να κερδίσουν τις σχετικές άδειες καταβάλλοντας υπέρογκα ποσά, ελπίζοντας να εξασφαλίσουν ένα μερίδιο στην νέα αγορά που γεννιέται. Τα κέρδη από αυτή την μετάβαση στα δίκτυα αυτά (τα οποία στην πλειοψηφία τους βασίζονται στα ήδη υπάρχοντα με κύριες αλλαγές σε λογισμικό) θα είναι μεγάλα, τόσο για τους παροχείς αυτών των δικτύων αλλά και για τον χρήστη αυτής της νέας τεχνολογίας.

Για τους παροχείς, δίνεται η ευκαιρία να εκμεταλλευτούν καλλίτερα την χωρητικότητα του δικτύου, καθώς πρόκειται για *packet switched* τεχνολογία (σε αντίθεση με τη σημερινή *circuit switched*). Αυτό σημαίνει ότι τα *time slots* δεσμεύονται δυναμικά για τη μεταφορά δεδομένων και όχι στατικά, ανεξάρτητα από την ανάγκη για *bandwidth*. Αυτό βέβαια αποτελεί μεγάλο πλεονέκτημα και για το χρήστη, αφού η χρέωση θα είναι ανά πακέτο πληροφορίας και όχι ανά χρόνο σύνδεσης (κάτι τέτοιο άλλωστε δεν θα υφίσταται αφού θα είναι συνεχώς συνδεδεμένος στο δίκτυο του παροχέα).

Ο κάτοχος ενός κινητού τηλεφώνου, θα μπορεί να έχει πρόσβαση στο διαδίκτυο ή στο εταιρικό εσωτερικό δίκτυο (*corporate intranet*) από οπουδήποτε. Τα κινητά τηλέφωνα θα εξελίσσονται διαρκώς και θα είναι δυνατό ο χρήστης να εγκαθιστά κατάλληλο λογισμικό για κάποια υπηρεσία. Αυτός είναι και ο λόγος που αναμένεται μεγάλο μερίδιο της μελλοντικής αυτής αγοράς να δεσμευτεί από υπηρεσίες παροχής εφαρμογών (*ASPs*). Παρ'όλες όμως τις δυνατότητες

που θα παρέχουν τα κινητά ή άλλες παρόμοιες συσκευές, η αλληλεπίδραση θα παραμένει ακόμα σε χαμηλά επίπεδα (οθόνη 5 γραμμών ή με μικρή ανάλυση). Σε κάτι τέτοιο μπορεί να βοηθήσει η αναγνώριση φωνής με τα συγκριτικά πλεονεκτήματα που προσφέρει.

5.5 Χρήση κωδικοποίησης φωνής αυτής της εργασίας

Αφού είδαμε στην προηγούμενο παράγραφο το περιβάλλον πιθανής χρησιμοποίησης μιας τεχνολογίας όπως αυτής που αναπτύχθηκε σε αυτή την εργασία (διαδίκτυο, ασύρματες επικοινωνίες), θα δείξουμε πως μπορεί αυτή να χρησιμοποιηθεί στη πράξη, στα δυο παραπάνω περιβάλλοντα ξεκινώντας από το δεύτερο (ασύρματες επικοινωνίες). Πρώτα όμως τονίζουμε ότι σκοπός αυτής της παραγράφου είναι να δείχτει κάτι τέτοιο από τεχνικής άποψης και όχι από άποψη δημιουργίας νέων εφαρμογών ή υπηρεσιών. Γι αυτό το λόγο και βασιζόμενοι στη παράγραφο 5.1, όπου είδαμε ότι η πιο διαδεδομένη εφαρμογή αναγνώρισης φωνής είναι η χρησιμοποίηση του σημερινού ενσύρματου και ασύρματου τηλεφωνικού δικτύου για ανάκτηση πληροφοριών κάθε είδους², θα δώσουμε έμφαση στο πως μπορεί να επεκταθεί η υπηρεσία αυτή στα δυο παραπάνω περιβάλλοντα.

5.6 Χρήση στις ασύρματες επικοινωνίες

Το μεγάλο πλεονέκτημα των δικτύων 3ης γενεάς, είναι η υποστήριξη δικτύου δεδομένων βασισμένο στο *IP* καθώς και οι μεγαλύτερες ταχύτητες μεταφοράς. Αυτό σημαίνει ότι αυτόματα παύει να υφίσταται ο διαχωρισμός δικτύων φωνής και δεδομένων. Ο πλούτος των πληροφοριών στο διαδίκτυο γίνεται αυτόματα διαθέσιμος με ένα *interface* όμως, όχι απόλυτα ικανοποιητικό.

Χρησιμοποιώντας το μοντέλο πελάτη-εξυπηρετητή που αναφέραμε στο Εισαγωγικό Κεφάλαιο προτείνουμε τη κωδικοποίηση της φωνής στο κινητό τηλέφωνο και την αποστολή σε συμπιεσμένη πια μορφή του αιτήματος για ανάκτηση της ζητούμενης πληροφορίας στον αναγνωριστή, μέσα από το δίκτυο δεδομένων. Τα πλεονεκτήματα για αυτή τη προσέγγιση είναι ότι ο χρήστης δεν χρειάζεται να ασχοληθεί ο ίδιος με την απαιτούμενη πλοήγηση μέχρι να βρει τη πληροφορία που επιζητά, γιατί αυτό θα γίνεται αυτόματα με τη χρήση

²Ο χρήστης καλεί από το τηλέφωνό κάποιο καθορισμένο αριθμό ζητώντας κάποια πληροφορία, η φωνή μεταδίδεται σε μορφή *m-law* (σταθερή τηλεφωνία) ή *GSM* (κινητή τηλεφωνία) και αφού αναγνωριστεί η πρόταση, το σύστημα απαντάει χρησιμοποιώντας *text to speech synthesis*

απλά της φωνής. Επιπρόσθετα, το κόστος για κάτι τέτοιο θα είναι πολύ χαμηλό, αφού ο απαιτούμενος αριθμός πακέτων που θα χρειάζεται να σταλούν στον αναγνωριστή είναι πολύ λίγα.

Σε αυτή την ενότητα θα εξετάσουμε ζητήματα που έχουν να κάνουν με μια τέτοια σχεδίαση, όπως τη δυνατότητα χρήσης συσκευών με υποστήριξη ασύρματου δικτύου δεδομένων (π.χ. κινητού) για τη κωδικοποίηση της φωνής και ζητήματα που αφορούν στην επικοινωνία δεδομένων σε τέτοια περιβάλλοντα (π.χ. *GPRS*). Πρώτα όμως θα μιλήσουμε για την δημιουργία ενός προτύπου (*standard*) για παρόμοιες εφαρμογές που προάγεται αυτό το καιρό από το Ινστιτούτο Τηλεπικοινωνιών της Ευρώπης.

5.6.1 Σχετικά πρότυπα : ETSI proposal

Το *ETSI* σύστησε πριν καιρό την ομάδα εργασίας *STQ-Aurora DSR* με σκοπό την δημιουργία ενός προτύπου για ανάπτυξη κατανεμημένων εφαρμογών αναγνώρισης φωνής, με πρόσβαση από κινητές συσκευές. Η πρώτη έκδοση του στάνταρ δημοσιεύτηκε το Φεβρουάριο του 2000 και προτείνει το μοντέλο πελάτη-εξυπηρετητή, όπως άλλωστε και το δικό μας μοντέλο. Η επεξεργασία του *front-end* γίνεται στον πελάτη, ο οποίος παράγει ακουστικά διανύσματα μεγέθους 14 στοιχείων (13 *cepstral* παράμετροι συν μια επιπρόσθετη παράμετρος για ευρωστία παρουσία θορύβου). Χρησιμοποιούν επίσης διανυσματικό κβαντισμό (*split-Vq* τεχνική με 2 υποδιανύσματα, σύμφωνα με πρόταση της *Motorola*). Προκύπτει κωδικοποίηση με ρυθμό δεδομένων *4.8Kbps*, πολύ μεγαλύτερο από το *2Kbps* που πετύχαμε σε αυτή την εργασία. Για περισσότερες πληροφορίες παραπέμπουμε στην [19].

5.6.2 Τεχνολογία ασύρματων συσκευών πληροφορίας

Αρχικά, τα κινητά τηλέφωνα, είχαν μόνο δυνατότητες φωνητικής επικοινωνίας. Αργότερα προστέθηκαν υπηρεσίες αποστολής/λήψης σύντομων μηνυμάτων (*SMS*), ενώ πρόσφατα απέκτησαν πρόσθετες δυνατότητες αναζήτησης πληροφοριών στο διαδίκτυο, με υπηρεσίες όπως *WAP* και *iMode*. Πρόσφατα τέλος, εμφανίστηκαν και τα πρώτα κινητά με υποστήριξη *GPRS*. Κατά γενική ομολογία, οι δυνατότητές τους θα αυξηθούν κατακόρυφα στο σύντομο μέλλον. Θα μιλάμε για ασύρματες συσκευές πληροφορίας (*Wireless Information Devices - WIDs*), με δυο κύριες υποκατηγορίες : τα έξυπνα τηλέφωνα (*smartphones*) φωνο-κεντρικές (*voice-centric*) συσκευές με επιπλέον δυνατότητες ανταλλαγής πληροφορίας και *communicators*, πληροφοριο-κεντρικές (*information-centric*) συσκευές με δυνατότητες φωνής και εισόδου από πληκτρολόγιο, στυλό ή άλλο τρόπο (φωνή;).

Οι συσκευές αυτές, έχουν συνήθως επεξεργαστές με πολύ μικρή κατανάλωση ισχύος και με ταχύτητες που φτάνουν αυτή τη στιγμή έως και τα 200MHz (ARM). Τρέχουν ειδικά σχεδιασμένα λειτουργικά συστήματα σαν το *Symbian* (κοινοπραξία *Ericsson, Nokia, Motorola*) με βιβλιοθήκες ανάπτυξης προγραμμάτων σε γλώσσες όπως *C++*, *Java*. Ειδικά για ανάπτυξη με τη χρήση *Java*, υπάρχει η ειδική έκδοση της πλατφόρμας, με την κωδική ονομασία *J2ME* (*Java 2 Micro Edition*, βλέπε [20]) η οποία παρέχει την ιδεατή μηχανή *KVM*, με απαιτήσεις μνήμης μόλις 100K. Επίσης, υποστηρίζει περιορισμένο αριθμό *APIs* της πλατφόρμας, για μεταφορά απλών σε πόρους και τεχνικές προγραμμάτων. Δυστυχώς όμως, δεν υπάρχει υποστήριξη πραγματικών αριθμών (*floats*), κάτι που κάνει αυτή τη στιγμή δύσκολη τη μεταφορά πακέτων σαν αυτό που αναπτύχθηκε με το *Yarrow* σύστημα. Κάτι τέτοιο όμως είναι πολύ πιθανό να αλλάξει σύντομα.

5.6.3 Δίκτυα 2.5 και 3ης γενεάς : υποσχέσεις, περιορισμοί και συμπεράσματα

Παρά τον ενθουσιασμό που υπάρχει γύρω από τις υπηρεσίες μεγάλου εύρους ζώνης, που θα έρθουν με τα δίκτυα 3ης γενεάς, κανείς πρέπει να κρατάει κάποιες επιφυλάξεις. Πρώτος λόγος, είναι ότι το εύρος ζώνης που ανακοινώνονται για τα πρωτόκολλα αυτά, είναι το θεωρητικό μέγιστο. Για παράδειγμα, για το *GPRS* μέγιστο θεωρητικό εύρος ζώνης θεωρούνται τα 171.2kbps (κάτι που απαιτεί δέσμευση 8 *time slots* και χρήση κωδικοποίησης *CS-4*), στην πράξη όμως δεν αναμένεται να ξεπεράσει τα 20-30kbps. Τα επόμενα πρωτόκολλα μέχρι τα 3ης γενεάς, θα υποστηρίζουν θεωρητικά μέχρι και 2Mbps, ποτέ όμως δεν πρόκειται να δούμε κάτι τέτοιο στη πράξη. Επίσης, όσο πιο εξελιγμένα είναι τα πρωτόκολλα αυτά, τόσο περιορισμένα αναμένεται να είναι από άποψη κάλυψης : για παράδειγμα το *EDGE* δεν αναμένεται να καλύψει περιοχές εκτός πόλεων.

Όμως, ανεξάρτητα από το εύρος ζώνης, η πιο σημαντική καινοτομία αυτών των δικτύων, είναι όπως είπαμε η υποστήριξη δικτύων δεδομένων *IP*, όπως ακριβώς αυτά του διαδικτύου. Φυσικά, αφού το πρωτόκολλο είναι το ίδιο, αναμένει κανείς να είναι εύκολο να μεταφερθούν οι υπάρχουσες εφαρμογές του διαδικτύου, σε συσκευές σε τέτοια δίκτυα. Κάτι τέτοιο δεν ισχύει απόλυτα τουλάχιστον, αφού το ασύρματο κανάλι έχει εντελώς διαφορετική συμπεριφορά από ένα ενσύρματο. Τα δυο βασικότερα χαρακτηριστικά που το διαφοροποιούν, είναι το υψηλότερο επίπεδο θορύβου από τη μια και η πολύ μεγαλύτερη καθυστέρηση (*latency*) των πακέτων από τη άλλη. Έτσι π.χ. από τα συστήματα πελάτη εξυπηρετητή ευνοούνται αυτά με μικρό αριθμό ανταλλαγής πληροφορίας (μικρό αριθμό αιτήσεων/απαντήσεων), όπως η εφαρμογή μας, ενώ αντίθετα

π.χ. η πλοήγηση σε μια σελίδα απαιτεί πολύ μεγάλο αριθμό τέτοιων αιτήσεων.

Βλέπουμε επομένως, ότι η ανάπτυξη εφαρμογών για πρόσβαση από ασύρματο περιβάλλον, απαιτεί να συμπεριληφθούν υπόψη διάφοροι παράγοντες που έχουν να κάνουν τόσο με τα ίδια τα δίκτυα, όσο και με τις συσκευές που παρέχουν πρόσβαση σε αυτά. Προκειμένου να υποστηριχθεί περαιτέρω η ανάπτυξη τέτοιων εφαρμογών, έχει ιδρυθεί ανάλογη πρωτοβουλία (*Mobile Applications Initiative*) (βλέπε [22]). Επίσης πρέπει να γίνει κατανοητό, ότι δεν είναι όλες οι εφαρμογές κατάλληλες για ένα τέτοιο περιβάλλον, άλλες για τεχνικούς λόγους και άλλες για λόγους χρησιμότητας. Αναμένεται τέτοιες εφαρμογές, να είναι απλές από άποψη χρήσης και να παρέχουν γρήγορη και εύκολη ανάκτηση, μικρών σε ποσότητα αλλά σημαντικών πληροφοριών, στο πλαίσιο στο οποίο ανήκει άλλωστε και η δική μας εφαρμογή (βλέπε [22]).

5.7 Χρήση στο διαδίκτυο

Αφού είδαμε πως μπορεί να αναπτυχθούν εφαρμογές αναγνώρισης φωνής στα ασύρματα δίκτυα δεδομένων, θα κάνουμε το ίδιο και για το περιβάλλον του διαδικτύου. Για το περιβάλλον του διαδικτύου, τέτοιες εφαρμογές υπάρχουν ήδη (βλέπε [13],[14],[15],[16],[17]), αν και δεν υπάρχει ακόμα ανάλογη εμπορική εκμετάλλευση. Η πιο ώριμη προσπάθεια είναι το σύστημα *WEB GALAXY* του Μ.Ι.Τ, η οποία επιτρέπει πρόσβαση μέσω διαδικτύου σε συγκεκριμένες βάσεις πληροφοριών (καιρός, αεροπορικά ταξίδια, οδηγός πόλης). Έμφαση σε αυτή τη προσπάθεια, έχει δοθεί περισσότερο στο τμήμα της κατανόησης φυσικής γλώσσας, στο πώς δηλαδή θα ερμηνευθεί η αίτηση του πελάτη, για την εύρεση της κατάλληλης πληροφορίας.

Και σε αυτή την ενότητα, επικεντρώνουμε τη προσοχή μας σε εφαρμογές αναγνώρισης φωνής για ανάκτηση μικρού όγκου πληροφοριών, σαν αυτή που είδαμε στη παράγραφο 5.1 (με μετάδοση φωνής μέσω τηλεφωνικού δικτύου). Αν αντικαταστήσουμε τον τρόπο αυτό επικοινωνίας, με μετάδοση φωνής μέσω του *WWW*, με μοναδικό περιορισμό την πρόσβαση στο διαδίκτυο και τις μικρές απαιτήσεις σε πόρους (εύρος ζώνης, επεξεργαστική ισχύς, μνήμη), έχουμε τα εξής πλεονεκτήματα :

Από τη μεριά του χρήστη :

- Αποφυγή μιας ξεχωριστής τηλεφωνικής κλήσης, συχνά υπεραστικής κατά τη διάρκεια της πλοήγησης.
- Ομοιόμορφη ολοκλήρωση αλληλεπίδρασης της εφαρμογής με την πλοήγηση.
- Αξιόπιστη μετάδοση δεδομένων (δίκτυο δεδομένων αντί δίκτυο φωνής)

Από τη μεριά του παροχέα μιας τέτοιας υπηρεσίας :

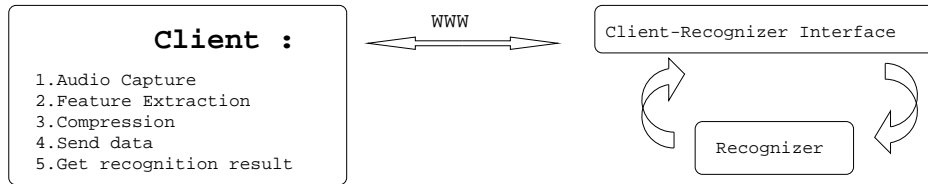
- Καθίσταται μη αναγκαία η ύπαρξη εξοπλισμού τηλεφωνικού κέντρου και διανομής κλήσεων, κάτι που απαιτεί επιπρόσθετο κόστος από τη μια, αλλά και εξειδικευμένες λύσεις σε υλικό και λογισμικό από την άλλη.
- Οι απαιτήσεις για εύρος ζώνης σε *2kbps* είναι πολύ αποτελεσματική, αφού εκτός τη μείωση κόστους, επιτρέπει σε τέτοια συστήματα και υπηρεσίες, να υποστηρίζουν πολύ μεγάλο αριθμό καναλιών εισόδου και αντίστοιχα αναγνωριστών (υποστήριξη μεγάλης κλίμακας -*large scale support*). Αντίθετα κάτι τέτοιο είναι πολύ δύσκολο να γίνει στην περίπτωση χρήσης τηλεφωνικού δικτύου, αφού υπάρχουν περιορισμοί ως προς τον μέγιστο αριθμό τηλεφωνικών καναλιών και ακόμα επιπρόσθετο κόστος.

Σε αυτό το κεφάλαιο θα εξετάσουμε τη δομή του συστήματος που αναπτύξαμε, προκειμένου να επιτευχθεί η αναγνώριση φωνής στο διαδίκτυο. Πρέπει να σημειωθεί σε αυτό το σημείο, ότι η υλοποίηση που παρουσιάζεται εδώ, είναι απλά ένας τέτοιος τρόπος. Η προσέγγιση θα μπορούσε να ήταν διαφορετική κάτω από διαφορετικές συνθήκες, περιορισμούς ή και σκοπούς. Η συγκεκριμένη υλοποίηση απλά στοχεύει στην ανάπτυξη ενός αρχικού πρότυπου τέτοιου συστήματος.

5.8 Γενική εικόνα του συστήματος

Σε αντίθεση με το κλασικό μονολιθικό σύστημα αναγνώρισης φωνής, η εφαρμογή μας ακολουθεί εκ των πραγμάτων το μοντέλο πελάτη - εξυπηρετητή (*client - server*), όπως άλλωστε κάνουν οι περισσότερες ιντερνετικές εφαρμογές. Ο πελάτης είναι ένα εφαρμογίδιο (*applet*)³, που αιχμαλωτίζει το σήμα της φωνής και το στέλνει μέσω δικτύου στον εξυπηρετητή προς αναγνώριση. Αυτό γίνεται αφού πρώτα εφαρμόσει ένα κατάλληλο σχήμα κωδικοποίησης της φωνής, με στόχο να ελαττώσει όσο γίνεται το απαιτούμενο εύρος ζώνης (*bandwidth*). Ένα μέσο μηχανήμα, είναι ικανό να εκτελέσει όλες αυτές τις λειτουργίες σε πραγματικό χρόνο (*real time*). Ο αναγνωριστής (*Recognizer Server*) επιτελεί όλο το δύσκολο έργο της αναγνώρισης και ενημερώνει τον πελάτη για το αποτέλεσμα. Η γενική μορφή του συστήματος φαίνεται στο Σχήμα 5.1.

³Η διαφορά, *Java applet* και *application* απλά έχει να κάνει με το περιβάλλον στο οποίο τρέχει ο κώδικας, αλλά η λειτουργικότητα είναι η ίδια. Το *demo* υπάρχει και στις 2 μορφές αλλά σαν σύμβαση από εδώ και πέρα θα χρησιμοποιείται ο όρος *applet*



Σχήμα 5.1: Επικοινωνία πελάτη - αναγνωριστή μέσω Web

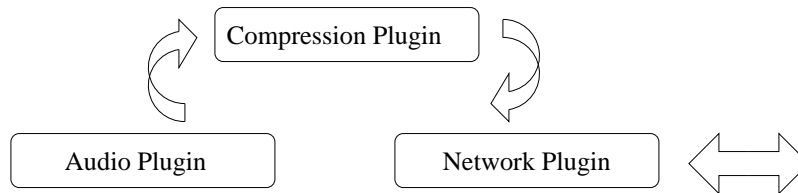
Όπως φαίνεται στο σχήμα, το σύστημα αποτελείται από 3 κομμάτια : Τον πελάτη(*applet*), τον αναγνωριστή καθώς και από το συνδετικό τους κρίκο, που κάνει τη διασύνδεση των δυο (*Client Recognizer Interface - CRI*).

5.9 Αρχιτεκτονική του πελάτη

Ο πελάτης είναι ένα *Java applet* που εκτελεί τις ακόλουθες λειτουργίες :

- Ηχογράφηση και αναπαραγωγή της προς αναγνώριση πρότασης
- Εξαγωγή και συμπίεση των παραμέτρων της φωνής
- Αποστολή της πρότασης στον αναγνωριστή και ενημέρωση για το αποτέλεσμα της αναγνώρισης

Για κάθε μια από τις παραπάνω λειτουργίες υπάρχει ένα *software module (Plugin)* που είναι υπεύθυνο για να τις εκτελέσει. Άρα έχουμε 3 *plugins* : το *Audio (AP)*, το *Compression (CP)* και το *Network (NP)*.



Σχήμα 5.2: Το applet με τα 3 plugins που το αποτελούν.

5.9.1 Audio Plugin

Η *Java* είναι μια γλώσσα προγραμματισμού ανεξάρτητη πλατφόρμας. Σαν τέτοια, δεν παρέχει τη δυνατότητα στον προγραμματιστή να επιτελέσει λειτουργίες εξαρτώμενες από κάποια συγκεκριμένη πλατφόρμα (*platform depended*).

Για να κάνει κανείς κάτι τέτοιο θα πρέπει να κάνει τον προγραμματισμό για αυτές τις λειτουργίες σε κάποια άλλη γλώσσα προγραμματισμού, και στη συνέχεια να κάνει σύνδεση του κώδικα αυτού, με την ιδεατή μηχανή (*virtual machine*) της *Java* μέσω ενός αρκετά πολύπλοκου τρόπου. Για τέτοιες λειτουργίες η *Java* διαθέτει τη τεχνολογία *Java Native Interface (JNI)*. Ο *platform dependent* κώδικας γράφτηκε σε *C* και μεταγλωττίστηκε σε βιβλιοθήκη, η οποία φορτώνεται από τον *java* κώδικα του *audio plugin* σε *run-time* για να επιτελέσει τις ζητούμενες λειτουργίες (*recording/rendering*). Η πλατφόρμα στην οποία έγινε η υλοποίηση είναι το λειτουργικό σύστημα *Linux*. Παρόλα αυτά, ο *audio driver*⁴ που χρησιμοποιήθηκε, υπάρχει διαθέσιμος σε όλες τις πλατφόρμες *UNIX*. Σε επερχόμενη έκδοση του *Java 2 SDK* αναμένεται η υποστήριξη των παραπάνω λειτουργιών μέσω ενός νέου *API* το οποίο θα αντικαταστήσει την παρούσα υλοποίηση του *Audio Plugin*.

5.9.2 Compression Plugin

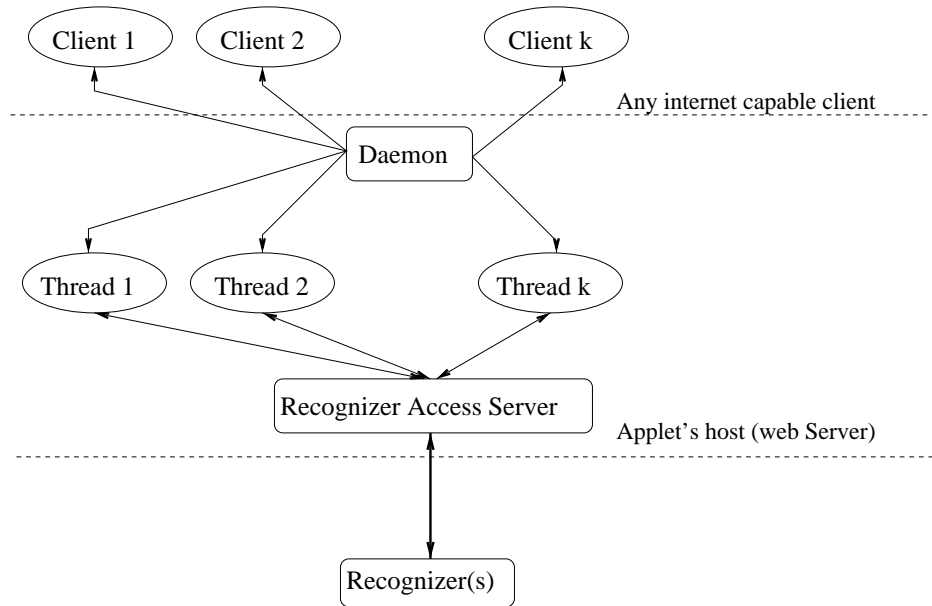
Το *plugin* αυτό επιτελεί 3 λειτουργίες χρησιμοποιώντας *clients* που είδαμε στο προηγούμενο κεφάλαιο (*package sri.star.coding.clients*). Είναι το πλέον βασικό *plugin*, που χρησιμοποιεί όλη τη τεχνολογία που αναπτύχθηκε για αυτή την εργασία.

- Ο *feature producer client (fpc)* είναι υπεύθυνος για να εφαρμόσει *front-end* ανάλυση στο σήμα της κυματομορφής που παίρνει από το *AudioPlugin*. Το αποτέλεσμα είναι ένας πίνακας $M \times N$ όπου N είναι το μέγεθος του *feature* που παράγουμε (π.χ. *Single*) και M ο αριθμός των *frames*.
- Ο *quantizer client (qc)* χβαντίζει το *feature* χρησιμοποιώντας το δοσμένο σχήμα κωδικοποίησης όπως είδαμε στο προηγούμενο κεφάλαιο.
- Ο *ascii gzipped writer (agw)* θα πάρει τον χβαντισμένο *feature* και θα δημιουργήσει μια συμπιεσμένη αναπαράστασή του, την οποία αργότερα θα κλιθεί να γράψει στο *socket* 'προς' τον αναγνώριστη (το οποίο θα δημιουργήσει το *Network Plugin*).

5.9.3 Network Plugin

Αυτό το *plugin* είναι υπεύθυνο για την δικτυακή επικοινωνία με τον αναγνώριστη. Παίρνει σαν είσοδο το συμπιεσμένο *feature* και το μεταδίδει στον αναγνώριστη. Επίσης ανακτά το αποτέλεσμα της αναγνώρισης και ενημερώνει το χρήστη για τυχόν προβλήματα.

⁴Συγκεκριμένα ο *Open Sound System (OSS 3.6.0)* www.4front-tech.com



Σχήμα 5.3: Επικοινωνία πελάτη αναγνωριστή μέσω *daemon*, *daemon threads* & *RAS* για μια συγκεκριμένη εφαρμογή-υπηρεσία. Για κάθε τέτοια εφαρμογή, ο *RAS* μπορεί να χρησιμοποιεί ένα γκρουπ αναγνωριστών, δρώντας έτσι σαν μηχανισμός εξισορρόπησης φορτίου (*load balancing mechanism*).

5.10 Επικοινωνία πελάτη αναγνωριστή

Η επικοινωνία των πελατών με τον αναγνωριστή, πρέπει να υποστηρίζει την εξυπηρέτηση πολλών πελατών ταυτόχρονα. Η πιο λογική υλοποίηση, θα ήταν να ενσωματωθεί αυτή η ικανότητα απευθείας στον κώδικα του αναγνωριστή. Δηλαδή για κάθε πελάτη, ο αναγνωριστής θα δημιουργούσε ένα καινούριο *process*, το οποίο θα διάβαζε από το *socket* με το πελάτη τα συμπιεσμένα δεδομένα και θα του έστελνε το αποτέλεσμα της αναγνώρισης. Για λόγους όμως υλοποίησης κυρίως, αποφασίστηκε να ενσωματωθεί αυτή η ικανότητα σε ένα σύνολο αντικειμένων, που αποτελούν τελικά το συνδετικό κρίκο μεταξύ των πελατών και του αναγνωριστή. Η δομή του συστήματος όπως έχει αυτή τη στιγμή, είναι αυτή που φαίνεται στο Σχήμα 5.3. Οι πελάτες επικοινωνούν με τον αναγνωριστή μέσω του *daemon*. Ο *daemon* είναι ένας δαίμονας που το μόνο που κάνει είναι να δημιουργεί ένα νέο *thread* για κάθε νέο πελάτη π.χ. για τον πελάτη *i* δημιουργείται το *thread i*, το οποίο είναι υπεύθυνο για να εξυπηρετήσει τον πελάτη. Το *thread* θα διαβάσει τα συμπιεσμένα δεδομένα

από το *socket* με τον πελάτη. Στη συνέχεια θα ζητήσει κάποιο *id* από τον *Recognizer Access Server (RAS)* το οποίο καθορίζει τη σειρά προτεραιότητας στην αναγνώριση. Στην συνέχεια το *thread* θα προσπαθήσει να κερδίσει πρόσβαση στον αναγνωριστή με βάση το *id* του. Όταν έρθει η σειρά του, ο *RAS* θα επικοινωνήσει με τον αναγνωριστή. Ο αναγνωριστής θα διαβάσει τα δεδομένα και θα επιστρέψει την πρόταση που αναγνώρισε στο *thread*, το οποίο με τη σειρά του θα την επιστρέψει στον πελάτη.

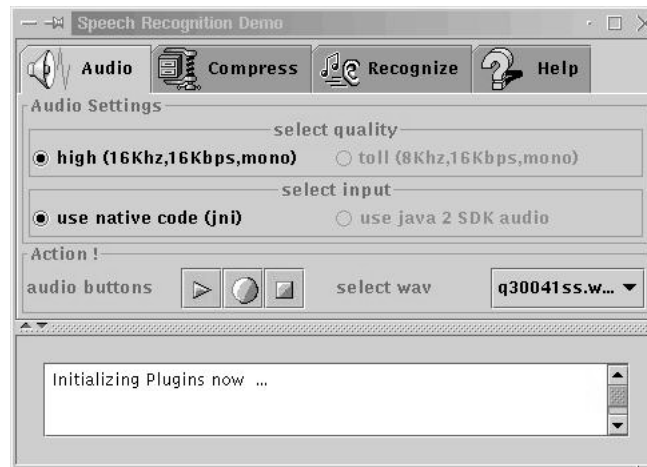
5.10.1 Εναλλακτικές προσεγγίσεις

Η υλοποίηση αυτή, υποθέτει ότι υπάρχει διαθέσιμος ένας μόνο αναγνωριστής, ο οποίος μάλιστα εξυπηρετεί σειριακά τις αιτήσεις για αναγνώριση. Σε ένα άλλο σενάριο (π.χ. μια πραγματική υπηρεσία), θα μπορούσε να υπάρχει μια φάρμα τέτοιων αναγνωριστών χωρισμένη σε γκρουπς, καθένα από τα οποία αντιστοιχεί σε μια καθορισμένη υπηρεσία ή εφαρμογή π.χ. ένα γκρουπ με αναγνωριστές για κράτηση αεροπορικών εισιτηρίων, ένα άλλο για πληροφορίες κίνησης μετοχών κ.ο.κ. Σε αυτή τη περίπτωση σε κάθε *Recognizer Access Server (RAS)*, αντιστοιχεί ένα τέτοιο γκρουπ αναγνωριστών και τότε ο *RAS* ενεργεί και σαν μηχανισμός εξισορρόπησης φορτίου (*load balancing mechanism*). Σε αυτή τη περίπτωση για κάθε τέτοια υπηρεσία, θα υπάρχει και ένας διαφορετικός *daemon* στον οποίο και θα απευθύνεται ο πελάτης.

Φυσικά, υπάρχουν ακόμα ένα σωρό τέτοιες λεπτομέρειες. Σε κάθε περίπτωση, αυτό που πρέπει να σημειωθεί είναι ότι στο *demo* αυτό, έχουν γίνει πολλές απλοποιήσεις όπως για παράδειγμα στην υλοποίηση της επικοινωνίας πελάτη-αναγνωριστή και σε καμιά περίπτωση δεν αντιπροσωπεύει μια 'πραγματική' εφαρμογή.

5.11 Speech Recognition Applet Demo

Όπως φαίνεται στις επόμενες εικόνες του *applet*, υπάρχουν 4 *panels* (πάνελς) με τα 3 πρώτα να αντιστοιχούν στα 3 *plugins* και το 4ο στο σύστημα βοήθειας του χρήστη. Σε κάθε ένα από τα 3 αυτά πάνελς, υπάρχουν διαθέσιμες 2 περιοχές. Η πρώτη (περιοχή επιλογών - *settings area*), επιτρέπει στο χρήστη να καθορίσει ποιες επιλογές θα χρησιμοποιήσει, ενώ η δεύτερη, (περιοχή ενεργειών - *action area*), να επιτελέσει την αντίστοιχη λειτουργία του παρόντος πάνελ (π.χ. συμπίεση για το *Compress Panel*). Τέλος, κοινή για όλα τα *panels* είναι η περιοχή μηνυμάτων (*Message Console*), η οποία δίνει πληροφορίες κατά τη διάρκεια της λειτουργίας του *demo*. Στη συνέχεια θα εξεταστεί η λειτουργία του κάθε πάνελ χωριστά.



Σχήμα 5.4: AudioPanel

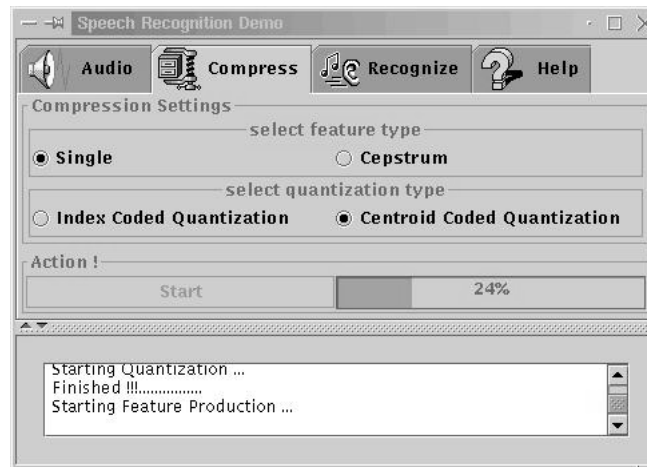
5.11.1 AudioPanel

Το *AudioPanel* (βλέπε Σχήμα 5.4), περιλαμβάνει επιλογές που έχουν να κάνουν με την ποιότητα του συστήματος ήχου, καθώς αυτή έχει να κάνει με το *front-end*. Αυτόματα ελέγχεται τι είδος *codebooks* υπάρχουν. Αν αυτά έχουν παραχθεί για παράδειγμα, μόνο για υψηλής ποιότητας ήχο, τότε η επιλογή για χαμηλή ποιότητα ήχου απενεργοποιείται, όπως φαίνεται και στο Σχήμα 5.4. Εκτός από την ποιότητα ήχου, υπάρχει επιλογή και για την χρησιμοποίηση *JNI (java SDK 1)* ή του *java SDK 2* κώδικα.

Στην περιοχή ενεργειών, ο χρήστης καλείται να καθορίσει μια πρόταση για είσοδο στο *front-end*. Μπορεί να ηχογραφήσει μια νέα κυματομορφή ή και να επιλέξει μια από τις διαθέσιμες. Για τη πρώτη ενέργεια, ο χρήστης πρέπει να επιλέξει το δεύτερο κουμπί (το τρίτο καθορίζει το τέλος της ηχογράφησης, ενώ το πρώτο την αναπαραγάγει) από τα 3 κουμπιά ήχου, ενώ για τη δεύτερη επιλογή, χρησιμοποιείται η μπάρα επιλογής που βρίσκεται στα δεξιά. Ενδιάμεσα γίνεται η επιλογή του τρόπου που επιθυμεί ο χρήστης.

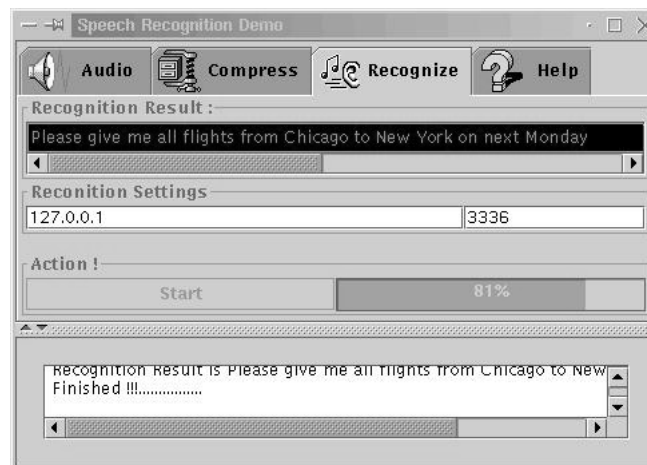
5.11.2 CompressPanel

Εδώ (βλέπε Σχήμα 5.5), ο χρήστης επιλέγει το είδος του *feature* που θα παραχθεί (*Single* ή *Csptrum*) και το είδος κωδικοποίησης (*Index* ή *Centroid Coded* κβαντισμό). Στην περιοχή ενεργειών, υπάρχουν το κουμπί έναρξης (*Start button*) και η μπάρα προόδου (*progress bar*) που δείχνει το ποσοστό της εκτελούμενης εργασίας που έχει γίνει έως τώρα. Σε αυτό το πάνελ επιτελούνται



Σχήμα 5.5: CompressPanel

δυο εργασίες : Η παραγωγή του *feature* από το *front-end* και στη συνέχεια η κωδικοποίηση με βάση το επιλεγμένο σχήμα κωδικοποίησης.

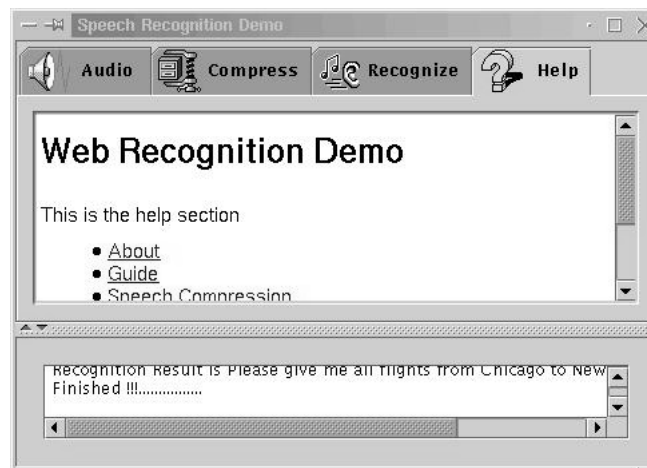


Σχήμα 5.6: RecognizePanel

5.11.3 RecognizePanel

Φτάνοντας στο *Recognize panel*, είμαστε έτοιμοι να στείλουμε το συμπιεσμένο πια μήνυμα στο αναγνωριστή, αρκεί να επιλέξουμε το κουμπί *start*. Και

εδώ όπως και στη περιοχή ενεργειών του προηγούμενου πάνελ, υπάρχει μπάρα προόδου που ενημερώνει το χρήστη για τη πρόοδο της αναγνώρισης. Στη περιοχή επιλογών, ο χρήστης μπορεί να καθορίσει την διεύθυνση *IP* και το *port* του μηχανήματος στο οποίο τρέχει ο *daemon*. Στην εικόνα 5.5 βλέπουμε ότι ο *daemon* τρέχει στο τοπικό μηχάνημα, αλλά φυσικά οποιαδήποτε έγκυρη διεύθυνση μπορεί να καθορισθεί (π.χ. 147.27.8.126). Το *Network plugin* θα αναλάβει την επικοινωνία με τον αναγνωριστή και μόλις ανακτηθεί το αποτέλεσμα της αναγνώρισης, θα εμφανιστεί στη μπάρα κειμένου που βρίσκεται στο πάνω μέρος του πάνελ (*Recognition Result text field*).



Σχήμα 5.7: HelpPanel

5.11.4 HelpPanel

Η βοήθεια για τη λειτουργία του *applet* χωρίζεται στα εξής μέρη :

- Αναλυτικό οδηγό για όλες τις λειτουργίες του *applet*
- Τεχνική ανάλυση που περιλαμβάνει την τεχνολογία συμπίεσης φωνής που χρησιμοποιεί το *applet*

5.12 Συμπεράσματα

- Τα αποτελέσματα του ερευνητικού κομματιού αυτής της εργασίας, καθιστούν δυνατή την δημιουργία εφαρμογών αναγνώρισης φωνής σε συστήματα πελάτη-εξυπηρετητή, για δίκτυα δεδομένων (διαδίκτυο, ασύρματα

δίκτυα όπως κινητή τηλεφωνία 2.5 και 3ης γενεάς), ακόμα και στις περιπτώσεις που ο μεταξύ τους διάυλος επικοινωνίας είναι αρκετά μικρού εύρους (της τάξης των *2kbps*).

- Τέτοιες εφαρμογές μπορεί να είναι η πρόσβαση σε πληροφορίες στο παγκόσμιο δίκτυο, τόσο από προσωπικούς υπολογιστές, όσο και άλλες συσκευές με δυνατότητες ασύρματης επικοινωνίας με το διαδίκτυο (*WIDs*, όπως κινητά τηλέφωνα επόμενης γενεάς, *smartphones*, *communicators* κ.α.). Παρουσιάστηκαν τα πλεονεκτήματα χρήσης δικτύου δεδομένων σε σχέση με χρήση δικτύων φωνής για τέτοιες εφαρμογές, τόσο από πλευράς χρήστη, όσο και παροχέων τέτοιων εφαρμογών.
- Όσο αφορά στο χώρο των ασύρματων δικτύων, τονίσαμε την επερχόμενη ραγδαία εξάπλωσή τους και τονίσαμε το πλεονέκτημα εφαρμογών αναγνώρισης φωνής σε αυτά τα περιβάλλοντα. Ερευνήσαμε τις διαφορές που υπάρχουν στα ασύρματα δίκτυα σε σχέση με τα ενσύρματα, τις δυνατότητες ύπαρξης εφαρμογών βασισμένες στο μοντέλο που προτείνουμε και τις προοπτικές αυτών.
- Όσο αφορά στο χώρο του διαδικτύου, δείξαμε πως μπορεί να χτιστεί μια απλή τέτοια εφαρμογή αναγνώριση φωνής, χρησιμοποιώντας το *Yarrow* σύστημα. Εξετάσαμε την λειτουργικότητα του πελάτη και τον τρόπο επικοινωνίας με τον αναγνωριστή και δείξαμε πως μπορεί να χτιστεί μια πραγματική εφαρμογή/υπηρεσία πάνω σε αυτό το μοντέλο.

Παράρτημα Α1

A-1.1 Πειράματα bit allocation

Όπως είδαμε στα Κεφάλαια 2 και 4, ο διανυσματικός κβαντισμός με χρήση υποδιανυσμάτων (*product VQ*), μπορεί να δώσει πολύ καλά αποτελέσματα, αν επιλεγθούν κατάλληλα τα υποδιανύσματα (επιλογή συντελεστών και ποσότητα πληροφορίας που απαιτούν). Δείξαμε στο Κεφάλαιο 4 ότι η καλλίτερη επίδοση που επιτύχαμε ήταν τα *2kbps*. Εδώ παρατίθενται μερικά από τα πειράματα που έγιναν κατά τη διάρκεια του *bit-allocation* (βλέπε παράγραφο 4.4.3), με 3-6 υποδιανύσματα (τα πειράματα 3, 3+, 4, 5, 5+ και 6). Τα πειράματα αυτά έγιναν τον Οκτώβριο του 1997 και έδωσαν πολύ καλά αποτελέσματα, τα καλλίτερα από τα οποία είδαμε στο Κεφάλαιο 4 και τα οποία δημοσιεύτηκαν στα [10], [11] και [12]. Όπως επισημάνθηκε στο Κεφάλαιο 4, ο *bit-allocation* αλγόριθμος μας βοηθάει στην εύρεση ενός αποτελεσματικού τρόπου διάθεσης των *bits* στα υποδιανύσματα, προσπαθώντας παράλληλα να διατηρήσει σε υψηλά επίπεδα τα αποτελέσματα της αναγνώρισης.

Γιά κάθε πείραμα υπάρχουν 2 πίνακες. Ο πίνακας στα αριστερά δείχνει τη δομή των υποδιανυσμάτων, ενώ στα δεξιά περιγράφεται η ακολουθία του *bit-allocation*. Όλα τα πειράματα ξεκινούν από 12 *bits* - πείραμα 1. Οι πρώτες επαναλήψεις αυξάνουν τον αριθμό *bits* ανά 2, ενώ οι επόμενες ανά 1. Έτσι φτάνουμε γρήγορα σε κάποια καλά αποτελέσματα, ενώ από εκεί και πέρα η διαδικασία γίνεται πιο ακριβής. Το καλλίτερο αποτέλεσμα για κάθε επανάληψη (αυτό που φαίνεται έντονα) κατοχυρώνεται, παίρνοντάς το σαν βάση για την επόμενη επανάληψη. Πολλές φορές φτάνουμε γρήγορα στο μέγιστο αριθμό *bits* που έχουμε διαθέσει για την εκπαίδευση ενός υποδιανύσματος, οπότε για τις επόμενες επαναλήψεις, η διαδικασία συνεχίζεται για τα υπόλοιπα μόνο υποδιανύσματα.

Δεδομένης της γνώσης που αποκομίσαμε από τα αποτελέσματα των πειραμάτων (οι πρώτοι συντελεστές απαιτούν περισσότερη πληροφορία) θα προσπαθήσουμε να ερμηνεύσουμε τα πειράματα που ακολουθούν.

A-1.2 πείραμα 3

Υποδιανύσματα και πείραμα 1 :

Εδώ βλέπουμε ότι οι 3 πιο σημαντικοί συντελεστές (1-3 αντίστοιχα), είναι κατανομημένοι στα 3 υποδιανύσματα. Λόγω του μεγέθους τους, το δεύτερο υποδιάνυσμα απαιτεί ένα *bit* παραπάνω από το πρώτο, ενώ το τρίτο απαιτεί 5 *bits* καθώς περιλαμβάνει επιπρόσθετα, τον αμέσως επόμενο από βαθμό σημαντικότητας συντελεστή (4).

Επαναλήψεις :

1. (πείραμα 2.1) : συντελεστής 1
2. (πείραμα 3.3) : συντελεστές 2, 4
3. (πείραμα 4.1) : συντελεστής 1 (μέγιστος αριθμός *bits*)
4. (πείραμα 5.3) : συντελεστές 2, 4
5. (πείραμα 6.3) : συντελεστές 2, 4
6. (πείραμα 7.3) : συντελεστές 2, 4

Παρατηρούμε ότι για το δεύτερο υποδιάνυσμα, ποτέ δεν χρειάστηκαν πάνω από 4 *bits*, πληροφορία ίση περίπου με αυτή που χρειάζεται μόνος του ο συντελεστής 3. Άλλωστε οι υπόλοιποι 4 συντελεστές του διανύσματος είναι οι λιγότερο σημαντικοί του *MFCC* διανύσματος. Έτσι, μόνο το 1ο και 3ο υποδιάνυσμα κερδίζουν *bits*. Δεδομένου ότι δεν είχαμε προβλέψει πάνω από 7 *bits* για το 1ο υποδιάνυσμα, όλα τα υπόλοιπα τα κερδίζει το 3ο υποδιάνυσμα.

| <i>SubVector</i> | <i>elements</i> |
|------------------|-----------------|
| 1 | 1 5 10 |
| 2 | 3 8 9 12 13 |
| 3 | 2 4 6 7 11 |

| <i>exper</i> | <i>% error</i> | <i>bits – allocated</i> | <i>total</i> |
|--------------|----------------|-------------------------|--------------|
| 1 | | 3 4 5 | 12 |
| 2.1 | 14.96 | 5 4 5 | 14 |
| 2.2 | 16.94 | 3 6 5 | |
| 2.3 | 15.42 | 3 4 7 | |
| 3.1 | 12.14 | 7 4 5 | 16 |
| 3.2 | 13.36 | 5 6 5 | |
| 3.3 | 11.58 | 5 4 7 | |
| 4.1 | 9.12 | 7 4 7 | 18 |
| 4.2 | 10.77 | 5 6 7 | |
| 4.3 | 10.67 | 5 4 9 | |
| 5.2 | 8.59 | 7 5 7 | 19 |
| 5.3 | 8.48 | 7 4 8 | |
| 6.2 | 8.69 | 7 5 8 | 20 |
| 6.3 | 8.31 | 7 4 9 | |
| 7.2 | 8.23 | 7 5 9 | 21 |
| 7.3 | 7.82 | 7 4 10 | |

A-1.3 πείραμα 3+

Υποδιανύσματα και πείραμα 1 :

Εδώ βλέπουμε ότι οι 3 πιο σημαντικοί συντελεστές (1-3 αντίστοιχα), είναι όλοι στο πρώτο υποδιάνυσμα, οι αμέσως 5 πιο σημαντικοί στο 2ο και οι υπόλοιποι επίσης 5 στο 3ο. Όλα τα υποδιανύσματα είναι αρχικά με 4 *bits*. Ήδη με τα όσα έχουμε δει έως τώρα προβλέπουμε ότι ο τελικός αριθμός *bits* θα είναι μεγαλύτερος από το 1ο στο 3ο υποδιάνυσμα με κίνδυνο μάλιστα να μην φτάσουν πάλι τα *bits* για το 1ο υποδιάνυσμα.

Επαναλήψεις :

1. (πείραμα 2.1) : συντελεστές 1, 2, 3
2. (πείραμα 3.2) : λόγω μεγέθους
3. (πείραμα 4.2) : λόγω μεγέθους
4. (πείραμα 5.1) : συντελεστές 1, 2, 3
5. (πείραμα 6.1) : συντελεστές 1, 2, 3 (μέγιστος αριθμός *bits*)
6. (πείραμα 7.3) : υποδιάνυσμα 3 (ένα *bit* μόνο)
7. (πείραμα 8.3) : υποδιάνυσμα 3 (εντάζει τώρα)

Παρατηρούμε ότι οι προβλέψεις βγήκαν αληθινές!

| <i>SubVector</i> | <i>elements</i> |
|------------------|-----------------|
| 1 | 1 2 3 |
| 2 | 4 5 6 7 8 |
| 3 | 9 10 11 12 13 |

| <i>exper</i> | <i>% error</i> | <i>bits – allocated</i> | <i>total</i> |
|--------------|----------------|-------------------------|--------------|
| 1 | | 4 4 4 | 12 |
| 2.1 | 11.43 | 6 4 4 | 14 |
| 2.2 | 11.63 | 4 6 4 | |
| 2.3 | 14.19 | 4 4 6 | |
| 3.1 | 9.53 | 8 4 4 | 16 |
| 3.2 | 9.19 | 6 6 4 | |
| 3.3 | 10.39 | 6 4 6 | |
| 4.1 | 8.46 | 8 6 4 | 18 |
| 4.2 | 8.38 | 6 8 4 | |
| 4.3 | 8.56 | 6 6 6 | |
| 5.1 | 7.62 | 8 7 4 | 19 |
| 5.2 | 7.90 | 6 9 4 | |
| 5.3 | 8.48 | 6 7 6 | |
| 6.1 | 7.54 | 9 7 4 | 20 |
| 6.2 | 7.62 | 8 8 4 | |
| 6.3 | 7.57 | 6 7 5 | |
| 7.2 | 7.65 | 9 8 4 | 21 |
| 7.3 | 7.44 | 9 7 5 | |
| 8.2 | 7.42 | 9 8 5 | 22 |
| 8.3 | 7.32 | 9 7 6 | |

A-1.4 πείραμα 4

Υποδιανύσματα και πείραμα 1 :

Εδώ βλέπουμε ότι οι 4 πιο σημαντικοί συντελεστές (1-4 αντίστοιχα), είναι κατανεμημένοι στα 4 υποδιανύσματα, ενώ το 1ο και 2ο υποδιάνυσμα έχουν μέγεθος μόλις δυο, αφού περιέχουν τους αμέσως πιο σημαντικούς συντελεστές (5 και 6 αντίστοιχα). Το 1ο πείραμα κατανέμει 2, 4, 2 και 4 *bits* αντίστοιχα στα 4 υποδιανύσματα. Προβλέπουμε πάλι, ότι ο τελικός αριθμός *bits* για το 1ο και 4ο υποδιάνυσμα μάλλον θα να ο μεγαλύτερος, αφού αυτά περιέχουν τους 2 πιο σημαντικούς συντελεστές (1 και 2).

Επαναλήψεις :

1. (πείραμα 2.1) : συντελεστής 1
2. (πείραμα 3.3) : συντελεστές 4, 6
3. (πείραμα 4.4) : συντελεστής 2
4. (πείραμα 5.1) : συντελεστής 1 (μέγιστος αριθμός *bits*)
5. (πείραμα 6.4) : συντελεστής 2
6. (πείραμα 7.3) : συντελεστές 4, 6

Παρατηρούμε ότι για το 1ο υποδιάνυσμα έχουμε για άλλη μια φορά έλλειψη *bits*!

| <i>SubVector</i> | <i>elements</i> |
|------------------|-----------------|
| 1 | 1 5 |
| 2 | 3 8 9 12 13 |
| 3 | 4 6 |
| 4 | 2 7 10 11 |

| <i>exper</i> | <i>% error</i> | <i>bits – allocated</i> | <i>total</i> |
|--------------|----------------|-------------------------|--------------|
| 1 | | 2 4 2 4 | 12 |
| 2.1 | 15.70 | 4 4 2 4 | 14 |
| 2.2 | 20.19 | 2 6 2 4 | |
| 2.3 | 17.63 | 2 4 4 4 | |
| 2.4 | 19.10 | 2 4 2 6 | |
| 3.1 | 12.45 | 6 4 2 4 | 16 |
| 3.2 | 14.01 | 4 6 2 4 | |
| 3.3 | 11.26 | 4 6 4 4 | |
| 3.4 | 12.37 | 4 6 2 6 | |
| 4.1 | 9.80 | 6 4 4 4 | 18 |
| 4.2 | 10.75 | 4 6 4 4 | |
| 4.3 | 10.87 | 4 4 6 4 | |
| 4.4 | 9.55 | 4 4 4 6 | |
| 5.1 | 8.81 | 6 4 4 5 | 19 |
| 5.2 | 9.55 | 4 6 4 5 | |
| 5.3 | 9.55 | 4 4 6 5 | |
| 5.4 | 8.86 | 4 4 4 7 | |
| 6.2 | 8.66 | 6 5 4 5 | 20 |
| 6.3 | 8.69 | 6 4 5 5 | |
| 6.4 | 8.36 | 6 4 4 6 | |
| 7.2 | 8.15 | 6 5 4 6 | 21 |
| 7.3 | 8.03 | 6 4 5 6 | |
| 7.4 | 8.10 | 6 4 4 7 | |

A-1.5 πείραμα 5

Υποδιανύσματα και πείραμα 1 :

Εδώ απλά έχουμε ένα παραπάνω υποδιάνυσμα, με τους συντελεστές 8 και 10 από τα 20 και 40 αντίστοιχα υποδιάνυσμα του προηγούμενου πειράματος και περιμένουμε αντίστοιχη συμπεριφορά - αποτελέσματα.

Επαναλήψεις :

1. (πείραμα 2.1) : συντελεστής 1
2. (πείραμα 3.3) : συντελεστές 4, 6
3. (πείραμα 4.4) : συντελεστής 2
4. (πείραμα 5.4) : συντελεστής 2
5. (πείραμα 6.4) : συντελεστές 2 (μέγιστος αριθμός *bits*)
6. (πείραμα 7.1) : συντελεστής 1 (ένα μόνο *bit*)
7. (πείραμα 8.1) : συντελεστής 1 (εντάξει τώρα)

| <i>SubVector</i> | <i>elements</i> |
|------------------|-----------------|
| 1 | 1 5 |
| 2 | 3 9 12 13 |
| 3 | 4 6 |
| 4 | 2 7 11 |
| 5 | 8 10 |

| <i>exper</i> | <i>% error</i> | <i>bits – allocated</i> | <i>total</i> |
|--------------|----------------|-------------------------|--------------|
| 1 | | 2 3 2 3 2 | 12 |
| 2.1 | 18.77 | 4 3 2 3 2 | 14 |
| 2.2 | 23.83 | 2 5 2 3 2 | |
| 2.3 | 20.75 | 2 3 4 3 2 | |
| 2.4 | 19.99 | 2 3 2 5 2 | |
| 2.5 | 25.91 | 2 3 2 3 4 | |
| 3.1 | 15.21 | 6 3 2 3 2 | 16 |
| 3.2 | 15.06 | 4 5 2 3 2 | |
| 3.3 | 13.36 | 4 3 4 3 2 | |
| 3.4 | 13.36 | 4 3 2 5 2 | |
| 3.5 | 16.18 | 4 3 2 3 4 | |
| 4.1 | 10.28 | 6 3 4 3 2 | 18 |
| 4.2 | 11.32 | 4 5 4 3 2 | |
| 4.3 | 12.9 | 4 3 6 3 2 | |
| 4.4 | 10.24 | 4 3 4 5 2 | |
| 4.5 | 12.22 | 4 3 4 3 4 | |
| 5.1 | 9.02 | 6 3 4 4 2 | 19 |
| 5.2 | 9.32 | 4 5 4 4 2 | |
| 5.3 | 10.15 | 4 3 6 4 2 | |
| 5.4 | 8.92 | 4 3 4 6 2 | |
| 5.5 | 9.88 | 4 3 4 4 4 | |
| 6.1 | 8.69 | 6 3 4 5 2 | 20 |
| 6.2 | 8.93 | 4 5 4 5 2 | |
| 6.3 | 9.93 | 4 3 6 5 2 | |
| 6.4 | 8.38 | 4 3 4 7 2 | |
| 6.5 | 9.52 | 4 3 4 5 4 | |
| 7.1 | 7.72 | 5 3 4 7 2 | 21 |
| 7.2 | 8.07 | 4 4 4 7 2 | |
| 7.3 | 8.45 | 4 3 5 7 2 | |
| 7.5 | 8.20 | 4 3 4 7 3 | |
| 8.1 | 7.01 | 6 3 4 7 2 | 22 |
| 8.2 | 7.87 | 5 4 4 7 2 | |
| 8.3 | 7.57 | 5 3 5 7 2 | |
| 8.5 | 7.70 | 5 3 4 7 3 | |

A-1.6 πείραμα 5+

Υποδιανύσματα και πείραμα 1 :

Εδώ βλέπουμε μια κατά σειρά σημαντικότητας κατανομή των συντελεστών (όπως και στο πείραμα 3+) σε 5 υποδιανύσματα, με τα 3 τελευταία να έχουν μεγαλύτερο μέγεθος, αφού περιέχουν λιγότερο σημαντικούς συντελεστές. Φαίνεται, ότι τα πολύ καλά αποτελέσματα οφείλονται και στην καλλίτερη συσχέτιση (αν και μικρή), που υπάρχει ανάμεσα στους συντελεστές των υποδιανυσμάτων που επιλέχθηκαν.

Επαναλήψεις :

1. (πείραμα 2.1) : συντελεστές 1 και 2
2. (πείραμα 3.3) : συντελεστές 5, 6, 7 (μέγεθος 3)
3. (πείραμα 4.2) : συντελεστές 3 και 4 !
4. (πείραμα 5.4) : συντελεστές 8, 9, 10 (μέγεθος 3)
5. (πείραμα 6.2) : συντελεστές 3 και 4 (είχαν πάρει μόνο 1 *bit*)

Το γεγονός ότι αυτή τη φορά δεν έχουμε έλλειψη *bits*, καθώς και η καλή αρχική κατανομή κατανομή τους, μας δίνει σε μόλις 5 επαναλήψεις (2Kbps) αποτέλεσμα 6.63, πολύ κοντά στο βέλτιστο 6.55!

| SubVector | elements |
|-----------|----------|
| 1 | 1 2 |
| 2 | 3 4 |
| 3 | 5 6 7 |
| 4 | 8 9 10 |
| 5 | 11 12 13 |

| exper | % error | bits — allocated | total |
|------------|--------------|------------------|-------|
| 1 | 16.79 | 3 3 2 2 2 | 12 |
| 2.1 | 11.71 | 5 3 2 2 2 | 14 |
| 2.2 | 13.13 | 3 5 2 2 2 | |
| 2.3 | 11.96 | 3 3 4 2 2 | |
| 2.4 | 13.72 | 3 3 2 4 2 | |
| 2.5 | 13.84 | 3 3 2 2 4 | |
| 3.1 | 11.71 | 7 3 2 2 2 | 16 |
| 3.2 | 9.50 | 5 5 2 2 2 | |
| 3.3 | 9.30 | 5 3 4 2 2 | |
| 3.4 | 9.40 | 5 3 2 4 2 | |
| 3.5 | 10.08 | 5 3 2 2 4 | |
| 4.1 | 8.38 | 7 3 4 2 2 | 18 |
| 4.2 | 7.80 | 5 5 4 2 2 | |
| 4.3 | 8.43 | 5 3 6 2 2 | |
| 4.4 | 8.10 | 5 3 4 4 2 | |
| 4.5 | 8.69 | 5 3 4 2 4 | |
| 5.1 | 8.38 | 7 4 4 2 2 | 19 |
| 5.2 | 7.80 | 5 6 4 2 2 | |
| 5.3 | 7.81 | 5 4 6 2 2 | |
| 5.4 | 6.99 | 5 4 4 4 2 | |
| 5.5 | 7.95 | 5 4 4 2 4 | |
| 6.1 | 7.06 | 6 4 4 4 2 | 20 |
| 6.2 | 6.63 | 5 5 4 4 2 | |
| 6.3 | 6.96 | 5 4 5 4 2 | |
| 6.4 | 6.83 | 5 4 4 5 2 | |
| 6.5 | 7.16 | 5 4 4 4 3 | |

A-1.7 πείραμα 6

Υποδιανύσματα και πείραμα 1 :

Εδώ έχουμε 5 υποδιανύσματα με 2 συντελεστές και ένα με 3. Λόγω και του μεγάλου αριθμού υποδιανυσμάτων είναι δυσκολότερο να φτάσουμε με λίγες μόνο επαναλήψεις σε καλά αποτελέσματα, ειδικά αν δει κανείς ότι η αρχική κατανομή των *bits* δεν είναι και η καλλίτερη. Έτσι, στα 20 *bits* το σφάλμα αναγνώρισης είναι ακόμα στο 8.10!

Επαναλήψεις :

1. (πείραμα 2.4) : λόγω συντελεστή 2
2. (πείραμα 3.1) : λόγω συντελεστή 1
3. (πείραμα 4.3) : συντελεστές 4 και 6
4. (πείραμα 5.3) : συντελεστές 4 και 6 (για ελάχιστη διαφορά με 5.2 πείραμα)
5. (πείραμα 6.2) : λόγω συντελεστή 3 (επιτέλους ...)
6. (πείραμα 7.1) : λόγω συντελεστή 1 (μέγιστος αριθμός *bits*)
7. (πείραμα 8.4) : λόγω συντελεστή 2
8. (πείραμα 9.2) : λόγω συντελεστή 3

Λόγω μεγέθους του πίνακα τα αποτελέσματα φαίνονται στην επόμενη σελίδα.

A-1.8 Σύγκριση αποτελεσμάτων bit allocation

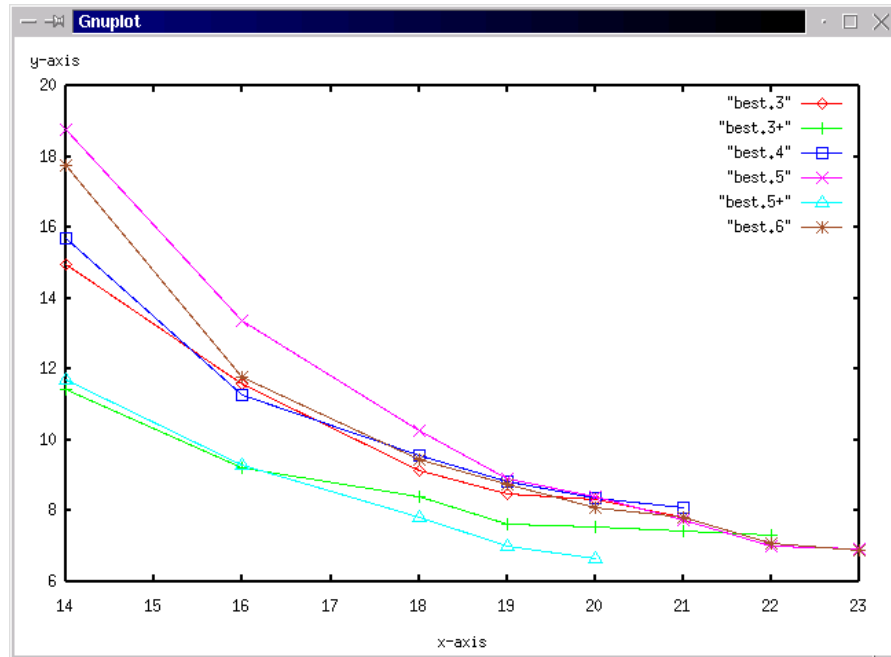
Είδαμε στο Κεφάλαιο 4, ότι η σημαντικότητα των συντελεστών παίζει πρωταρχικό ρόλο. Παρ'όλα αυτά, κανείς θα μπορούσε να χρησιμοποιήσει αυτό σαν πρωτεύον κριτήριο, με δευτερεύον κριτήριο τις συσχετίσεις μεταξύ των συντελεστών, αποκομίζοντας έτσι ακόμα μεγαλύτερα οφέλη. Φυσικά, μεγάλο ρόλο στη διεξαγωγή των πειραμάτων, διαδραματίζει και η αρχική κατανομή των *bits* στα υποδιανύσματα. Αν αυτή δεν είναι η κατάλληλη, όσο καλές και αν είναι οι υπόλοιπες επιλογές, είναι δύσκολο να καταλήξουμε σε καλά αποτελέσματα, σε λίγες μόνο επαναλήψεις. Επιπρόσθετα, το γεγονός ότι οι πρώτες επαναλήψεις ήταν με βήμα 2 *bits*, ίσως συνετέλεσε αρνητικά σε κάποιες περιπτώσεις. Τέλος, το γεγονός ότι κάποια *codebooks* αποδείχθηκαν μικρότερα από ότι είχε προβλεφτεί, συνετέλεσε στην όχι τόσο καλή απόδοση ή και στην αργή σύγκλιση σε κάποιο καλό αποτέλεσμα.

Η σύγκριση των αποτελεσμάτων φαίνεται στο Σχήμα A-1. Φαίνεται καθαρά η υπεροχή του πειράματος 5+, το οποίο φτάνει κοντά στο βέλτιστο 6.55, σε μόλις 4 επαναλήψεις. Τα πειράματα 6 και 5 καταφέρνουν να φτάσουν σε απόδοση 6.78 και 6.93 αντίστοιχα, σε 8 όμως επαναλήψεις - 23 *bits*. Ακολου-

θεί το πείραμα 3+, με απόδοση 7.32 στα 22 *bits*, ενώ τα πειράματα 3 και 4 δείχνουν σχετικά απογοητευτικά, με απόδοση μόλις 7.82 και 8.10 αντίστοιχα στα 21 *bits*.

| <i>SubVector</i> | <i>elements</i> |
|------------------|-----------------|
| 1 | 1 5 |
| 2 | 3 9 |
| 3 | 4 6 |
| 4 | 2 7 |
| 5 | 8 10 |
| 6 | 11 12 13 |

| <i>exper</i> | <i>% error</i> | <i>bits – allocated</i> | <i>total</i> |
|--------------|----------------|-------------------------|--------------|
| 1 | | 2 2 2 2 2 2 | 12 |
| 2.1 | 20.96 | 4 2 2 2 2 2 | 14 |
| 2.2 | 25.68 | 2 4 2 2 2 2 | |
| 2.3 | 22.83 | 2 2 4 2 2 2 | |
| 2.4 | 17.75 | 2 2 2 4 2 2 | |
| 2.5 | 29.46 | 2 2 2 2 4 2 | |
| 2.6 | 29.41 | 2 2 2 2 2 4 | |
| 3.1 | 11.76 | 4 2 2 4 2 2 | 16 |
| 3.2 | 14.10 | 2 4 2 4 2 2 | |
| 3.3 | 12.80 | 2 2 4 4 2 2 | |
| 3.4 | 15.70 | 2 2 2 6 2 2 | |
| 3.5 | 16.00 | 2 2 2 4 4 2 | |
| 3.6 | 16.43 | 2 2 2 4 2 4 | |
| 4.1 | 10.34 | 6 2 2 4 2 2 | 18 |
| 4.2 | 10.01 | 4 4 2 4 2 2 | |
| 4.3 | 9.45 | 4 2 4 4 2 2 | |
| 4.4 | 10.42 | 4 2 2 6 2 2 | |
| 4.5 | 11.66 | 4 2 2 4 4 2 | |
| 4.6 | 11.28 | 4 2 2 4 2 4 | |
| 5.1 | 9.07 | 6 2 3 4 2 2 | 19 |
| 5.2 | 8.79 | 4 4 3 4 2 2 | |
| 5.3 | 8.76 | 4 2 5 4 2 2 | |
| 5.4 | 9.09 | 4 2 3 6 2 2 | |
| 5.5 | 9.60 | 4 2 3 4 4 2 | |
| 5.6 | 9.60 | 4 2 3 4 2 4 | |
| 6.1 | 8.36 | 6 2 4 4 2 2 | 20 |
| 6.2 | 8.10 | 4 4 4 4 2 2 | |
| 6.3 | 9.17 | 4 2 6 4 2 2 | |
| 6.4 | 8.56 | 4 2 4 6 2 2 | |
| 6.5 | 9.12 | 4 2 4 4 4 2 | |
| 6.6 | 9.25 | 4 2 4 4 2 4 | |
| 7.1 | 7.82 | 6 3 4 4 2 2 | 21 |
| 7.2 | 8.10 | 4 5 4 4 2 2 | |
| 7.3 | 8.26 | 4 3 6 4 2 4 | |
| 7.4 | 8.10 | 4 3 4 6 2 2 | |
| 7.5 | 8.38 | 4 3 4 4 4 2 | |
| 7.6 | 8.26 | 4 3 4 4 2 4 | |
| 8.2 | 7.19 | 6 4 4 4 2 2 | 22 |
| 8.3 | 7.34 | 6 3 5 4 2 2 | |
| 8.4 | 7.09 | 6 3 4 5 2 2 | |
| 8.5 | 7.26 | 6 3 4 4 3 2 | |
| 8.6 | 8.18 | 6 3 4 4 2 3 | |
| 9.2 | 6.78 | 6 4 4 5 2 2 | 23 |
| 9.3 | 7.14 | 6 3 5 5 2 2 | |
| 9.4 | 7.26 | 6 3 4 6 2 2 | |
| 9.5 | 6.86 | 6 3 4 5 3 2 | |
| 9.6 | 6.96 | 6 3 4 5 2 3 | |



Σχήμα A-1: Σύγκριση αποτελεσμάτων Στο σχήμα φαίνονται και τα 6 πειράματα. Ο y -άξονας είναι το $WER(\%)$ και ο x -άξονας ο συνολικός αριθμός $bits$. Το πείραμα 5+ είναι μακράν η καλλίτερη επιλογή δίνοντας WER 6.63 στα μόλις 20 $bits$ (2Kbps)!

Καταλήγοντας, βλέπουμε ότι για την καλή απόδοση των πειραμάτων, ένας μεγάλος αριθμός παραγόντων πρέπει να καθοριστεί κατάλληλα, κάτι αρκετά δύσκολο. Παρ'όλα αυτά, καταφέραμε να πάρουμε πολύ καλά αποτελέσματα με μόλις 2 Kbps, κάτι εξαιρετικά ενθαρρυντικό. Είναι πράγματι μια πρόκληση, να δούμε αν και κατά πόσο ακόμα, μπορεί να μειωθεί αυτός ο ήδη πολύ χαμηλός ρυθμός μετάδοσης.

Βιβλιογραφία

- [1] Steve Young : *Large Vocabulary Continuous Speech Recognition: A Review*, April 1996.
- [2] Tony Robinson : *Speech Analysis*, lecture notes.
- [3] *The HTK Book*, Entropic, Cambridge Research Laboratory.
- [4] Lawrence Rabiner, Biing-Hwang Juang : *Fundamentals of Speech Recognition*, Prentice Hall.
- [5] John R. Deller, John G. Proakis, John H. L. Hansen : *Discrete Processing of Speech Signals* , Prentice Hall.
- [6] John Makhoul, Salim Roucos, Herbert Gish : *Vector Quantization in Speech Coding*, Proceedings of the IEEE, VOL 73, NO 11, November 1985.
- [7] Robert M. Gray : *Vector Quantization*, IEEE ASSR Magazine April 1984.
- [8] Allen Gersho, Robert M. Gray : *Vector Quantization And Signal Compression*, Kluwer Academic Publishers.
- [9] John R. Deller, John G. Proakis, John H. L. Hansen : *Discrete-Time Processing of Speech Signals*, Prentice Hall
- [10] V.Digalakis, L.Neumeyer, M.Perakakis : *Product-code Vector Quantization of Cepstral Parameters for Speech Recognition over the Web*, Proceedings ICSLP, December 1998.
- [11] V.Digalakis, L.Neumeyer, M.Perakakis : *Quantization of Cepstral Parameters for Speech Recognition over the Web* Proceedings ICCASP '98, Seattle, WA, May 1998.

- [12] V.Digalakis, L.Neumeyer, M.Perakakis : *Quantization of Cepstral Parameters for Speech Recognition over the Web* IEEE Journal on Selected Areas In Communications, Jan. 1999, volume 17, pp. 82-90.
- [13] D.Goddeau, W.Goldenthal, C.Weikart : *Deploying speech applications over the web*, Proc Eurospeech Rhodes Greece,Sept. 1997, pp. 685-688.
- [14] S.Bayer :*Embedding speech in web interfaces*, in Proc. ICSLP, Philadelphia, PA, Oct. 1996, pp. 1898-1901.
- [15] M.Sokolov : *Speaker verification on the World Wide Web*, in Proc. Eurospeech Rhodes Greece,Sept. 1997, pp. 847-850.
- [16] C.Hemphill, Y.Muthusamy : *Developing web-based speech applications*, in Proc. Eurospeech Rhodes Greece, Sept. 1997, pp. 847-850.
- [17] Raymond Lau, Giovanni Flammia, Christine Pao, and Victor Zue : *WEB GALAXY - Intergrating Spoken Language And Hypertext Navigation*, www.sls.lcs.mit.edu
- [18] Hakon Gudding, *Capacity Analysis of GPRS*, Master Thesis
- [19] David Pearce - Motorola Labs & Chairman ETSI STQ-Aurora DSR Working Group, *Enabling New Speech Driven Services for Mobile Devices : An overview of the ETSI standards activitiew for Distributed Speech Recognition Front-ends*
- [20] *Java 2 Platform Micro Edition(J2ME) for Creating Mobile Devices*, White Paper, Sun microsystems
- [21] *Mobile Applications Initiative : Developer's Guide*, www.mai.com, Ericsson
- [22] *Wirelles Packet Data and its Applications*, Symbian Conference Transcript, Symbian Developer Network(www.symbiandevnet.com)
- [23] Τσακαλίδης Σταύρος, Διπλωματική εργασία : *Κρυφά μαρκοβιανά μοντέλα με μείγματα διακριτών κατανομών για αναγνώριση ομιλίας μεγάλου λεξιλογίου*.