Πτυχιακή εργασία

«Εργαλεία και τεχνικές αυτόματης αναγνώρισης ομιλίας από τον υπολογιστή»

Του φοιτητή
Στογιαννίδη Κωνσταντίνου
Αρ. Μητρώου: 093448

Επιβλέπων καθηγητής
Διαμαντάρας Κωνσταντίνος

Θεσσαλονίκη 2018

Ευχαριστίες

Με την αποπεράτωση της πτυχιακής μου εργασίας κλείνει ένα κεφαλαίο της ζωής μου ως φοιτητής στο Τμήμα Μηχανικών Πληροφορικής. Γι' αυτό το λόγο θα ήθελα να ευχαριστήσω όλους εκείνου που με βοήθησαν σε αυτό μου το ταξίδι, παρέχοντας μου γνώσεις και εμπειρίες για την συνέχεια.

Ιδιαίτερα θα ήθελα να ευχαριστήσω όλους τους καθηγητές της σχολής μου για τις γνώσεις και την καθοδήγηση που μου έδωσαν όλα αυτά τα χρόνια και ιδιαίτερα, τον κύριο Κωνσταντίνο Διαμαντάρα για την ευκαιρία που μου έδωσε να αναλάβω αυτήν την πτυχιακή εργασία, καθώς και για την απαραίτητη καθοδήγηση του, ώστε να καταφέρω να ολοκληρώσω με επιτυχία την πτυχιακή μου εργασία.

Τέλος θέλω να ευχαριστήσω όλους τους φίλους μου και την οικογένεια μου που με στήριξαν και με βοήθησαν με όποιο τρόπο μπορούσε ο καθένας όλα αυτά τα χρόνια, σε όλες τις δυσκολίες της ζωής, και μου έδωσαν δύναμη να συνεχίσω.

Περίληψη

Το θέμα αυτής της πτυχιακής εργασίας είναι να δημιουργηθεί ένα ακριβές σύστημα αυτόματης αναγνώρισης ομιλίας. Για το σκοπό αυτό, χρησιμοποιήθηκε το Kaldi, ένα εργαλείο ανοιχτού κώδικα για την αναγνώριση ομιλίας γραμμένο σε C ++. Αρχικά, εξηγείται λεπτομερώς η κύρια διαδικασία της αυτόματης αναγνώρισης ομιλίας. Στη συνέχεια, παρουσιάζεται λεπτομερώς το εργαλείο Kaldi. Επιπρόσθετα, παρατίθεται και η εγκατάσταση του εργαλείου Kaldi. Έπειτα, γίνεται η ανάλυση των βημάτων που απαιτούνται για τη δημιουργία του συστήματος αυτόματης αναγνώρισης ομιλίας. Για την εκπαίδευση του συστήματός μας, χρησιμοποιούμε δυο τρόπους εκπέδευσης, τη μονοφωνική εκπαίδευση και την απλή εκπαίδευση τριφώνου.

Λέξεις κλειδιά:Αυτόματη αναγνώριση ομιλίας, Kaldi, ακουστικό μοντέλο, γλωσσικό μοντέλο, Hidden Markov Models, Νευρωνικα δίκτυα

Abstract

The topic of this thesis is to build an accurate automatic speech recognition system. For this purpose we use Kaldi, an open-source toolkit for speech recognition written in C++. First of all, the main process of automatic speech recognition is explained in details. Secondly, the toolkit Kaldi is explained in details. Furthermore, the installation of the Kaldi toolkit is listed. Next, we analyze the steps required to create the speech recognition system. To train our system, we use two different ways of training, monophonic education and simple trifle education.

Keywords: Automatic speech recognition (ASR), Kaldi, Acoustic model, Language model, Hidden Markov Models, Neural Networks.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Πίνακας Πινάκων

1 Εισαγωγή

Η Αυτόματη Αναγνώριση Ομιλίας (ASR) είναι ένας τομέας της τεχνητής νοημοσύνης που έχει ως ο κύριο στόχο να επιτρέπει τη προφορική επικοινωνία μεταξύ ανθρώπων και ηλεκτρονικών υπολογιστών. Με άλλα λόγια, ουσιαστικά επικεντρώνεται στο να μετατρέψει αυτόματα την ανθρώπινη ομιλία σε κείμενο. Το κύριο πρόβλημα της αυτόματης αναγνώρισης ομιλίας είναι η πολυπλοκότητα της ανθρώπινης γλώσσας. Οι άνθρωποι για να ακούσουν, δεν χρησιμοποιούν μόνο τα αυτιά τους. Χρησιμοποιούν τις γνώσεις που έχουν για τον ομιλητή, καθώς και το περιβάλλον του. Στην ASR έχουμε μόνο το σήμα ομιλίας. Όπως θα δούμε όμως παρακάτω, υπάρχουν διαφορετικές προσεγγίσεις για να προσπαθήσουμε να επιτύγχουμε μια καλή απόδοση της ομιλίας.

Τόσο η ακουστική μοντελοποίηση όσο και η γλωσσική μοντελοποίηση αποτελούν σημαντικά στοιχεία των αλγορίθμων αναγνώρισης ομιλίας, οι οποίοι βασίζονται σε σύγχρονες στατιστικές. Τα σύγχρονα συστήματα αναγνώρισης ομιλίας γενικής χρήσης, βασίζονται στα μοντέλα Hidden Markov (HMM). Αυτά είναι στατιστικά μοντέλα που εξάγουν μια ακολουθία συμβόλων ή ποσότητες.

Στο σύστημα αυτόματης αναγνώρισης ομιλίας που θα δημιουργήσουμε στα πλαίσια αυτής της εργασίας, θα χρησιμοποιήσουμετο το Kaldi, μια εργαλειοθήκη για την αναγνώριση ομιλίας γραμμένη σε C ++ και με άδεια χρήσης την Άδεια Apache v2.0.

1.1Επισκόπηση της εργασίας και στόχοι

Σκοπός αυτής της εργασίας είναι να μάθουμε για τα ASR και το εργαλελιο Kaldi, για να μπορέσουμε να δημιουργήσουμε ένα σύστημα αυτόματης αναγνώρισης ομιλίας.

Αρχικά, θα μελετήσουμε όλες τις διαδικασίες που σχετίζονται με την αυτόματη αναγνώριση της ομιλίας, με σκοπό να καταλάβουμε πώς ακριβώς λειτουργεί και, συνεπώς, να είμαστε σε θέση να δημιουργίσουμε ένα βασικό σύστημα.

Στη συνέχεια, θα κάνουμε μια αναφορά στο εργαλείο Kaldi. Θα γίνει μια αναλυτική επισκόπηση του εργαλείου και θα δούμε στη συνέχεια πως μπορούμε να το εγκαταστήσουμε σε ένα λειτουργικό περιβάλλον Unix.

Επιπρόσθετα, θα αναφέρουμε όλα τα βήματα που πρέπει να ακολουθήσει κάποιος για να δημιουργήσει ένα απλό σύστημα αυτόματης αναγνώρισης ομιλίας. Σε αυτό το έργο, για την εκπαίδευση του συστήματός μας θα χρησιμοποιήσουμε δύο τρόπους εκπαίδευσης.

1.2 Περίγραμμα εργασίας

Στο κεφάλαιο 2 παρουσιάζουμε την κύρια θεωρία για την αυτόματη Αναγνώριση Ομιλίας.

Στην αρχή του κεφαλαίου 3 παρουσιάζουμε το εργαλείο Kaldi και στη συνέχεια την εγκατάστασή του.

Έπειτα, στο κεφάλειο 4 παρουσιάζουμε με λεπτομέριεα το υποδειγματικό έργο που υλοποιήθηκε για τις απιτήσεις της συγκεκριμένης εργασίας.

Τέλος, το κεφάλαιο 5 περιλαμβάνει τα συμπεράσματα στα οποία καταλήξαμε κατά τη διάρκεια αυτής της εργασίας.

2 Αναγνώριση ομιλίας

2.1 Αυτόματη αναγνώριση ομιλίας

Η στατιστική προσέγγιση της αυτόματης αναγνώρισης ομιλίας στοχεύει στη μοντελοποίηση της στοχαστικής σχέσης μεταξύ ενός σήματος ομιλίας και της ομιλούμενης ακολουθίας λέξεων, με στόχο την ελαχιστοποίηση του αναμενόμενου ποσοστού σφάλματος ενός ταξινομητή. Το στατιστικό παράδειγμα διέπεται από το κανόνα απόφασης του Bayes. Δεδομένης μιας σειράς ακουστικών παρατηρήσεων $x_1^T=x_1,\ldots,x_T$ ως τα συστατικά στοιχεία μιας προφορικής έκφρασης, ο κανόνας απόφασης του Bayes αποφασίζει για τη σειρά λέξεων $w_1^N=w_1,\ldots,w_N$ η οποία μεγιστοποιεί τη μεταγενέστερη πιθανότητα κατάταξης $p(w_1^N|x_1^T)$:

$$[w_1^N]opt=argmaxP(w_1^N|x_1^T) \qquad (2.1)$$

Υπό την προϋπόθεση ότι χρησιμοποιείται η πραγματική κατανομή πιθανότητας, ο κανόνας απόφασης του Bayes είναι ο βέλτιστος μεταξύ όλων των κανόνων απόφασης. Με άλλα λόγια, εγγυάται, κατά μέσο όρο, το χαμηλότερο δυνατό ποσοστό σφάλματος ταξινόμισης. Ωστόσο, για τις περισσότερες εργασίες αναγνώρισης προτύπων, η πραγματική κατανομή πιθανότητας δεν είναι συνήθως γνωστή, αλλά πρέπει να αντικατασταθεί με τη κατάλληλη κατανομή μοντέλου. Στην αυτόματη αναγνώριση ομιλίας, το γενετικό μοντέλο, το οποίο αποσυνθέτει τη μεταγενέστερη πιθανότητα κατάταξης σε ένα προϊόν με δυο ανεξάρτητες στοχαστικές πηγές γνώσης, έγινε ευρέως αποδεκτό.

$$P(w_1^N|x_1^T)=P(x_1^T)*P(x_1^T|w_1^N)/P(x_1^T) \qquad (2.2)$$

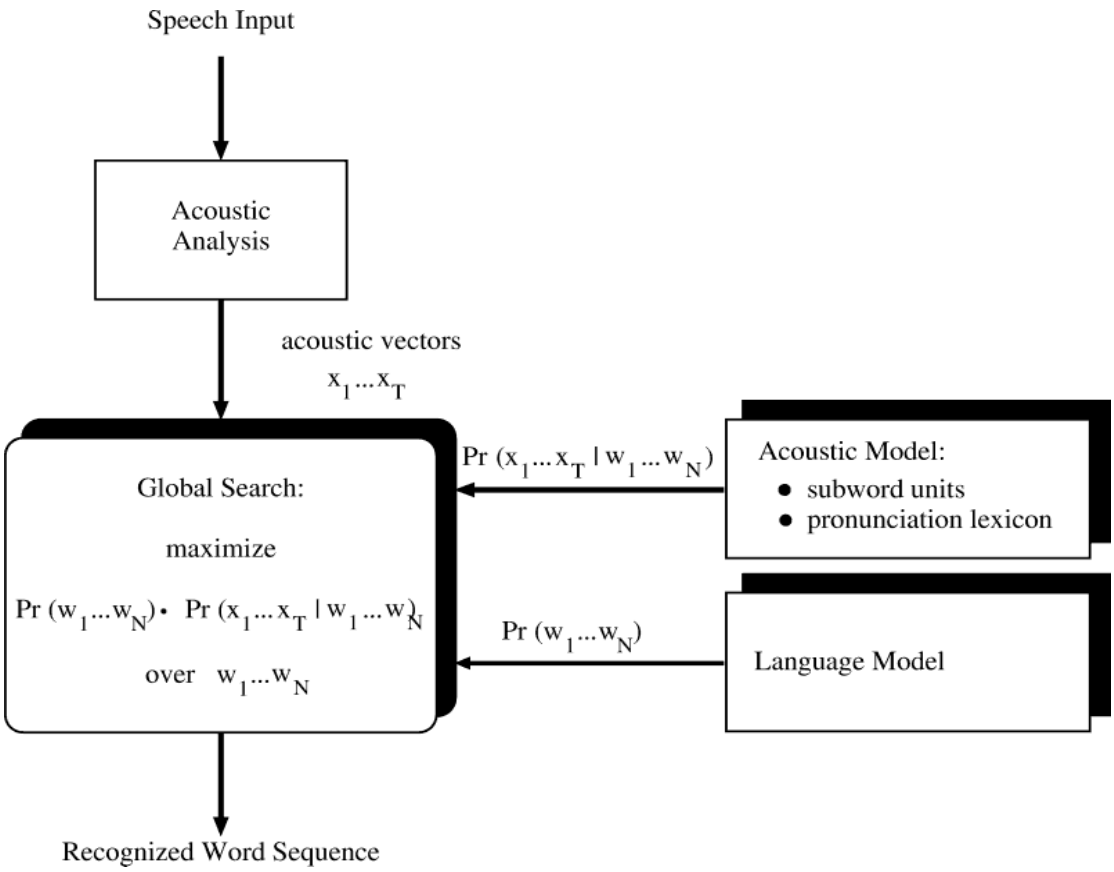
Ο παρονομαστής $p(x_1^T)$ στην εξίσωση 2.2 θεωρείτε ότι είναι ανεξάρτητος από την ακολουθία λέξεων w_1^N και ως εκ τούτου, ο κανόνας απόφασης είναι ισοδύναμος με:

$$P(w_1^N|x_1^T)=P(x_1^T)*P(x_1^T|w_1^N) \qquad (2.3)$$

Η ακολουθία λέξεων [w_1^N]opt που μεγιστοποιεί τη μεταγενέστερη πιθανότητα κατάταξης, καθορίζεται με αναζήτηση για εκείνη την ακολουθία λέξεων που μεγιστοποιεί το προϊόν των ακόλουθων δυο στοχαστικών πηγών γνώσεων:

- Το ακουστικό μοντέλο ($x_1^T | w_1^N$) που καταγράφει την πιθανότητα παρατήρησης μιας ακολουθίας ακουστικών παρατηρήσεων x_1^T δεδομένης μιας ακολουθίας λέξεων w_1^N.
- Το γλωσσικό μοντέλο P(w_1^N) το οποίο παρέχει μια προηγούμενη πιθανότητα για την ακολουθία λέξεων w_1^N.

Ένας στατιστικός αναγνωριστής ομιλίας αξιολογεί και συνδυάζει και τα δύο μοντέλα μέσω της παραγωγής και της βαθμολόγησης ενός μεγάλου αριθμού εναλλακτικών ακολουθιών λέξεων (λεγόμενες και υποθέσεις) κατά τη διάρκεια μιας σύνθετης διαδικασίας αναζήτησης. Το σχήμα 2.1 απεικονίζει τη βασική αρχιτεκτονική ενός στατιστικού συστήματος αυτόματης αναγνώρισης ομιλίας [1].



Εικόνα 1:Αρχιτεκτονική συστήματος αυτόματης αναγνώρισης ομιλίας

2.1.1 Ανάλυση σήματος

Το πρώτο βήμα σε οποιοδήποτε σύστημα αυτόματης αναγνώρισης ομιλίας είναι η εξαγωγή χαρακτηριστικών, δηλαδή η ταυτοποίηση των συστατικών του ηχητικού σήματος, τα οποία είναι κατάλληλα για την αναγνώριση του γλωσσικού περιεχομένου, και την απομάκρυνση όλων των άλλων υλικών που μεταφέρουν πληροφορίες. Δεν υπάρχουν δυο εκφράσεις της ίδιας λέξης ή φράσης για να δημιουργηθεί το ίδιο ψηφιακό σήμα. Με άλλα λόγια, ο σκοπός της ανάλυσης σήματος είναι να αντλήσει ένα χαρακτηριστικό διάνυσμα, έτσι ώστε τα δυανύσματα για το ίδιο φώνημα, να είναι όσο το δυνατόν πιο κοντά το ένα στο άλλο, ενώ τα διανύσματα για διαφορετικά φωνήματα να είναι πολύ διαφορετικά μεταξύ τους.

Οι κύριοι παράγοντες που θα μπορούσαν να προκαλέσουν τη διαφορά μεταξύ δύο τυχαίων δειγμάτων ομιλίας είναι:

- Φωνητική ταυτότητα: Οι διαφορές μεταξύ της προφοράς των ομιλητών εξαρτώνται από το φύλο, τη διάλεκτο, τη φωνή, κ.λπ.
- Μικρόφωνο: Και λοιπές ιδιότητες του καναλιού μετάδοσης.
- Περιβάλλον: Θόρυβος περιβάλλοντος, ακουστική δωματίου, κ.λπ.

Οι συνήθεις τεχνικές επεξεργασίας σήματος που χρησιμοποιούνται στην αυτόματη αναγνώριση ομιλίας βασίζονται στο συντελεστή Mel Frequency Cepstral Coefficients (MFCC) και στο Perceptual Linear Prediction. Σε αυτή την εργασία θα χρησιμοποιήσουμε το MFCC.

Το βασικό σημείο για να το καταλάβουμε, είναι ότι οι ήχοι που παράγονται από έναν άνθρωπο, φιλτράρονται από το σχήμα της φωνητικής οδούς. Αυτό το σχήμα καθορίζει ποιος ήχος βγαίνει και εκδηλώνεται στο φάκελο του φάσματος ισχύος, σε μικρό χρονικό διάστημα. Αν μπορούμε να προσδιορίσουμε το σχήμα με ακρίβεια, αυτό θα μας δώση μια ακριβής αναπαράσταση του φωνήματος που παράγεται. Η δουλειά του MFCC και του PLP είναι να αντιπροσωπεύουν με ακρίβεια αυτό το φάκελο.

Οι μετασχηματισμοί του MFCC και του PLP εφαρμόζονται σε ένα δειγματοληπτικό και κβαντισμένο ηχητικό σήμα. Ο συνολικός υπολογισμός του MFCC που ακολουθεί το kaldi είναι:

- Επεξεργάζεται τον αριθμό των πλαισίων στο αρχείο (τυπικά, πλαίσια 25ms μετατοπισμένα κάθε φορά κατά 10ms)
- Για κάθε πλαίσιο:
    1. Εξάγει τα δεδομένα, πραγματοποιεί προαιρετική αποσύνθεση, δίνει έμφαση εκ των προτέρων στην αφαίρεση της αντιστάθμισης dc, και τη πολλαπλασιάζει με μια λειτουργία παραθύρου.
    2. Σε αυτό το σημείο, εκτελεί την ενέργεια (αν χρησιμοποιείται log-energy και όχι CO).
    3. Κάνει Fast Fourier Transform (FFT) και υπολογίζει το φάσμα ισχύος.
    4. Υπολογίζει την ενέργεια σε κάθε μεμβράνη.
    5. Υπολογίζει το αρχείο καταγραφής των ενεργειών και ακολουθεί το μετασχηματισμό συνημιτόνου, κρατώντας όσους συντελεστές καθορίζονται.
    6. Προαιρετικά, κάνει cepstral liftering. Ουσιαστικά πρόκειται για μια κλιμάκωση των συντελεστών, για να εξασφαλιστεί ένα εύλογο εύρος.

Η εξαγωγή χαρακτηριστικών είναι ένα ουσιαστικό πρώτο βήμα στις εφαρμογές αναγνώρισης ομιλίας. Εκτός από τη στατική εξαγωγή χαρακτηριστικών από κάθε πλαίσιο δεδομένων ομιλίας, είναι ωφέλιμο να χρησιμοποιηθούν κάποιοι μετασχηματισμοί, με σκοπό τη βελτίωση της αναγνώρισης.

Μετασχηματισμοί, προβολές και άλλες λειτουργίες χαρακτηριστικών που συνήθως δεν είναι συγκεκριμένες για ομιλητές περιλαμβάνουν:

- Σύνδεση πλαισίων και υπολογισμός χαρακτηριστικών Delta.

- Ανάλυση γραμμικού διαχωρισμού (Linear Discriminant Analysis (LDA) transform).
- Ετεροσεκτική ανάλυση γραμμικού διαχωρισμού (Heteroscedastic Linear Discriminant Analysis (HLDA)).
- Εκτίμιση μέγιστης πιθανότητας γραμμικού μετασχηματισμού (Maximum Likelihood Linear Transform (MLLT) estimation) [2].

2.1.2 Ακουστικό μοντέλο

Το ακουστικό μοντέλο $P(x_1^T | w_1^N)$ παρέχει μια στοχαστική περιγραφή για την υλοποίηση μιας ακολουθίας διανυσμάτων ακουστικής παρατήρησης x_1^T δεδομένης μιας ακολουθίας λέξεων w_1^N. Λόγω της ακεραιότητας των δεδομένων, το μοντέλο για μεμονωμένες λέξεις, καθώς και το μοντέλο για ολόκληρες φράσεις, επιτυγχάνονται συνδυάζοντας τα ακουστικά μοντέλα βασικών μονάδων υπο-λέξεων σύμφωνα με ένα λεξικό προφοράς. Μονάδες υπο-λέξεων μικρότερες από λέξεις, ενεργοποιούν έναν αναγνωριστή ομιλίας για να επιτρέπεται η αναγνώριση λέξεων που δεν εμφανίζονται στα δεδομένα εκπαίδευσης. Έτσι, το σύστημα αναγνώρισης μπορεί να εξασφαλίσει ότι υπάρχουν αρκετές περιπτώσεις για κάθε μονάδα υπο-λέξεων που έχουν παρατηρηθεί κατά τη διάρκεια της εκπαίδευσης, έτσι ώστε να καταστεί δυνατή μια αξιόπιστη εκτίμηση των υποκείμενων παραμέτρων του μοντέλου.

Ο τύπος των μονάδων υπο-λέξεων που χρησιμοποιούνται σε έναν αναγνωριστή ομιλίας εξαρτάται από την ποσότητα των διαθέσιμων δεδομένων εκπαίδευσης και την επιθυμητή πολυπλοκότητα του μοντέλου. Ενώ τα συστήματα αναγνώρισης που έχουν σχεδιαστεί για μικρά μεγέθη λεξιλογίου (<100 λέξεις) εφαρμόζουν τυπικά μοντέλα ολόκληρων λέξεων, τα συστήματα που έχουν αναπτυχθεί για την αναγνώριση μεγάλων λεξιλογίων (>5000 λέξεις) χρησιμοποιούν συχνά μικρότερες μονάδες υπο-λέξεων, οι οποίες μπορεί να αποτελούνται από συλλαβές, φωνήματα, κ.λπ. Φωνήματα που εξαρτώνται από το περιβάλλον αναφέρονται επίσης ως n-phones. Κοινώς χρησιμοποιούμενες μονάδες υπο-λέξεων που χρησιμοποιούνται σε συστήματα αναγνώρισης ομιλίας με μεγάλα λεξιλόγια είναι n-phones στο πλαίσιο ενός ή δύο γειτονικών φωνημάτων, τα λεγόμενα triphones ή

quinphones. Τα μοντέλα φωνημάτων που εξαρτώνται από το περιβάλλον επιτρέπουν τη λήψη διαφορετικών αρθρώσεων που υπόκειται ένα φώνημα όταν πραγματοποιείται σε διαφορετικά φωνητικά πλαίσια (συν-άρθρωση) [1].

Τυπικά, τα συστατικά φωνήματα παράγονται για διάφορες ακουστικές πραγματοποιήσεις της ίδιας λέξης με διαφορετική διάρκεια και διαφορετική φασματική διαμόρφωση, ακόμα και αν οι εκφράσεις παράγονται από τον ίδιο ομιλητή. Συνεπώς, κάθε φώνημα θα συγκεντρώνει έναν άγνωστο αριθμό ακουστικών παρατηρήσεων. Η χρονική παραμόρφωση των διαφορετικών προφορών καθώς και η φασματική παραλλαγή στο ακουστικό σήμα μπορεί να περιγραφεί μέσω ενός μοντέλου Hidden Markov Model (HMM). Ένα HMM είναι ένας στοχαστικός αυτοματοποιητής πεπερασμένων καταστάσεων, που μοντελοποιεί τη μεταβολή του ακουστικού σήματος μέσω μιας στοχαστικής διαδικασίας δυο σταδίων. Ο αυτοματοποιητής ορίζεται μέσω ενός συνόλου καταστάσεων με μεταβάσεις που συνδέουν τις καταστάσεις. Η πιθανότητα $P(x_1^T|w_1^N)$ επεκτείνεται από μη παρατηρίσιμες (κρυφές) μεταβλητές που αντιπροσωπεύουν τις καταστάσεις [2]:

$$P(w_1^N|x_1^T)=\sum P(x_1^T,s_1^T|w_1^N) \qquad (2.4)$$

2.1.3 Γλωσσικό μοντέλο

Το γλωσσικό μοντέλο $P(w_1^N)$ παρέχει μια προγενέστερη πιθανότητα για τη σειρά λέξεων $w_1^N=w_1,\ldots,w_N$. Έτσι, εκ φύσεως αποσκοπεί στη καταγραφή της σύνταξης, της σημασιολογίας και της πραγματικότητας μιας γλώσσας. Από τη χρονική στιγμή που τα γλωσσικά μοντέλα είναι ανεξάρτητα από ακουστικές παρατηρήσεις, οι παράμετροί τους μπορούν να εκτιμηθούν από μεγάλες συλλογές κειμένων, όπως π.χ. εφημερίδες, άρθρα περιοδικών ή περιεχόμενο ιστού. Λόγω ενός, θεωρητικά, άπειρου αριθμού πιθανών ακολουθιών λέξεων, τα γλωσσικά μοντέλα απαιτούν ένα κατάλληλο μοντέλο παραδοχής, για να καταστεί εφικτό το πρόβλημα της εκτίμησης. Για την αναγνώριση ομιλίας μεγάλου λεξιλογίου, έχουν γίνει ευρέως αποδεκτά τα γλωσσικά μοντέλα m-gram. Ένα γλωσσικό μοντέλο m-gram βασίζεται στη παραδοχή ότι μια ακολουθία λέξεων ακολουθεί μια (m-1) διαδικασία Markov. Με άλλα λόγια, η

πιθανότητα μιας λέξης w_m υποτίθεται ότι εξαρτάται μόνο από τις m-1 προηγούμενες λέξεις [1].

2.2 Μοντέλα, μέθοδοι και αλγόριθμοι

2.2.1 Μοντέλα Hidden Markov (HMMs)

Τα σύγχρονα συστήματα αναγνώρισης της ομιλίας γενικής χρήσης βασίζονται στα μοντέλα Hidden Markov. Τα οποία είναι στατιστικά μοντέλα που εξάγουν μία ακολουθία από σύμβολα ή ποσότητες. Τα Μοντέλα HM (Hidden Markov) χρησιμοποιούνται στην αναγνώριση ομιλίας, διότι ένα σήμα ομιλίας/λόγου μπορεί να θεωρηθεί ως ένα τμηματικό στάσιμο σήμα ή ένα στατικό σήμα σύντομου χρόνου. Σε σύντομο χρονικό διάστημα κλίμακας (π.χ. 1 δέκατο του δευτερολέπτου), η ομιλία μπορεί να προσεγγιστεί ως στάσιμη διαδικασία [3].

Ένας άλλος λόγος που τα Μοντέλα HM είναι δημοφιλείς είναι επειδή μπορούν να εκπαιδευθούν αυτόματα και είναι απλά και υπολογιστικά εφικτό να χρησιμοποιηθούν. Στην αναγνώριση ομιλίας, το Μοντέλο HM εξάγει μία ακολουθία Ν-Διαστάσεων πραγματικών τιμών διανυσμάτων (με το Ν να είναι ένα μικρός ακέραιος αριθμός όπως 10), εξάγοντας ένα από αυτά κάθε 10 χιλιοστά του δευτερολέπτου. Τα διανύσματα θα αποτελούνται από φασματικούς συντελεστές, οι οποίοι λαμβάνονται μέσω ενός μετασχηματισμού Fourier από ένα σύντομο χρονικό διάστημα του παραθύρου της ομιλίας και αφαιρώντας το φάσμα χρησιμοποιώντας ένα μετασχηματισμό ημιτόνου, στην συνέχεια, λαμβάνεται το πρώτο (πιο σημαντικό) συντελεστή. Το Μοντέλο HM συνηθίζει να έχει σε κάθε κατάσταση μία στατιστική κατανομή που είναι ένα μίγμα διαγώνιας συν διακύμανσης Gaussians, η οποία θα δώσει μία πιθανότητα για κάθε διάνυσμα που παρατηρήθηκε. Για κάθε λέξη ή για πιο γενικά συστήματα αναγνώρισης ομιλίας, θα έχουν μία διαφορετική κατανομή εξόδου. Για μία σειρά από λέξεις το Μοντέλο HM είναι μία συνένωση διαφόρων εκπαιδευμένα επιμέρους μοντέλα για τις ξεχωριστές λέξεις [3].

Αυτά που περιγράφονται παραπάνω είναι τα βασικά στοιχεία, των πιο κοινών συστημάτων βασισμένα σε Μοντέλα HM για την αναγνώριση ομιλίας. Τα

σύγχρονα συστήματα αναγνώρισης ομιλίας χρησιμοποιούν διάφορους συνδυασμούς τυποποιημένων τεχνικών προκειμένου να βελτιωθούν τα αποτελέσματα πάνω από την βασική προσέγγιση που περιγράφεται παραπάνω [3].

Αποκωδικοποίηση της ομιλίας (ο όρος για το τι συμβαίνει όταν στο σύστημα παρουσιάζεται μία νέα έκφραση και πρέπει να υπολογιστεί η πιο πιθανή προέλευση έκφρασης) θα χρησιμοποιήσει κατά πάσα πιθανότητα τον αλγόριθμό Viterbi για να βρει το καλύτερο μονοπάτι και εδώ υπάρχει μία επιλογή μεταξύ συνδυασμού Μοντέλων HM δυναμικά δημιουργημένα, ο οποίος περιλαμβάνει τόσο την ακουστική όσο και την γλωσσική πληροφορία του μοντέλου, και συνδυασμένα στατιστικά εκ των προτέρων (την πεπερασμένη κατάσταση του μετατροπέα, ή FST, προσέγγιση) [3].

Μία πιθανή βελτίωση στην αποκωδικοποίηση είναι να κρατηθεί ένα σύνολο καλών υποψηφίων, αντί απλά κρατώντας τον καλύτερο υποψήφιο, και να χρησιμοποιηθεί μία καλύτερη λειτουργία βαθμολόγησης (εκ νέου βαθμολόγηση) για να βαθμολογηθεί αυτό το σύνολο, έτσι ώστε να μπορεί να επιλεχθεί το καλύτερο σύμφωνα με διπλή βαθμολόγηση. Το σύνολο των υποψηφίων μπορεί να κρατηθεί είτε ως λίστα (N-Λίστα η καλύτερη προσέγγιση) ή ως ένα υποσύνολο των μοντέλων (ένα πλέγμα). Η αναβαθμολόγηση γίνεται συνήθως με την προσπάθεια να ελαχιστοποιηθεί ο κίνδυνος Bayes (ή μία προσέγγιση αυτού). Αντί να πάρει την αρχική πρόταση με μέγιστη πιθανότητα, προσπαθεί να αναλάβει την πρόταση που ελαχιστοποιεί το προσδόκιμο της απώλειας της συγκεκριμένης λειτουργίας σε σχέση με όλες τις πιθανές καταγραφές. Η συνάρτηση απώλειας είναι συνήθως η απόσταση Levenshtein, αν και μπορεί να είναι διαφορετικές αποστάσεις για συγκεκριμένα καθήκοντα, το σύνολο των πιθανών καταγραφών κλαδεύεται για να διατηρηθεί ανιχνευσιμότητα. Οι αποδοτικοί αλγόριθμοι έχουν επινοηθεί για την αναβαθμολόγηση των πλεγμάτων που εκπροσωπήθηκαν ως σταθμισμένες πεπερασμένες μετατροπείς καταστάσεων με επεξεργασμένες τις αποστάσεις, παρουσιάζοντας τους εαυτούς τους ως μία πεπερασμένη κατάσταση μετατροπών επαληθεύσεις ορισμένων υποθέσεων [3].

2.2.2 Αναγνώριση ομιλίας δυναμικής χρονικής στρέβλωσης (Dynamic Time Warping)

Η Δυναμική Χρονική Στρέβλωση ήταν ο αλγόριθμος που είχε χρησιμοποιηθεί περισσότερο στην αναγνώριση ομιλίας, αλλά πλέον έχει εκτοπιστεί σε μεγάλο βαθμό από τον πιο επιτυχημένο αλγόριθμο Hidden Markov Models [4].

Δυναμική Χρονική Στρέβλωση είναι ένας αλγόριθμος για τη μέτρηση της ομοιότητας μεταξύ δύο αλληλουχιών που μπορεί να διαφέρουν στον χρόνο ή στην ταχύτητα. Για παράδειγμα, οι ομοιότητες στα πρότυπα περπατήματος θα ανιχνευθούν, ακόμα και αν σε ένα βίντεο το πρόσωπο περπατούσε αργά και αν σε άλλο αυτός ή αυτή περπατούσε πιο γρήγορα, ή ακόμα και αν υπήρχαν επιταχύνσεις και επιβραδύνσεις κατά τη διάρκεια μιας παρατήρησης. Ο DTW έχει εφαρμοστεί σε βίντεο, ήχο και γραφικά. Όλα τα δεδομένα που μπορούν να μετατραπούν σε μία γραμμική αναπαράσταση μπορούν να αναλυθούν με DTW [4].

Μία πολύ γνωστή εφαρμογή είναι η αυτόματη αναγνώριση ομιλίας, για να αντιμετωπιστούν οι διαφορετικές ταχύτητες ομιλίας. Σε γενικές γραμμές, είναι μία μέθοδος που επιτρέπει σε έναν υπολογιστή να βρει μία βέλτιστη αντιστοιχία μεταξύ δύο συγκεκριμένων αλληλουχιών με ορισμένους περιορισμούς. Δηλαδή, οι αλληλουχίες είναι «στρεβλή» μη γραμμικά να ταιριάζουν μεταξύ τους. Αυτή η μέθοδος ευθυγράμμισης αλληλουχίας χρησιμοποιείται συχνά στα δεδομένα του Μοντέλου Hidden Markov.

2.2.3 Νευρωνικά δίκτυα

Τα νευρωνικά δίκτυα εμφανίστηκαν ως ελκυστική προσέγγιση ακουστικής μοντελοποίησης στα ASR, στα τέλη της δεκαετίας του 1980. Από τότε, τα νευρωνικά δίκτυα έχουν χρησιμοποιηθεί σε πολλές πτυχές της αναγνώρισης ομιλίας, όπως στη ταξινόμηση φωνημάτων, στη μεμονωμένη αναγνώριση λέξεων, στην αναγνώριση ομιλίας οπτικοακουστικών μέσων, στην αναγνώριση ομιλητών οπτικοακουστικών μέσων καθώς και στη προσαρμογή ομιλητών [5].

Σε αντίθεση με τα Μοντέλα Hidden Markov, τα Νευρωνικά Δίκτυα δεν κάνουν υποθέσεις σχετικά με την λειτουργία στατιστικών ιδιοτήτων και έχουν πολλές ιδιότητες που τα καθιστούν ως ελκυστικό μοντέλο αναγνώρισης για την αναγνώριση ομιλίας. Όταν χρησιμοποιείται για να εκτιμήσουν τις πιθανότητες ενός χαρακτηριστικού τμήματος της ομιλίας, τα Νευρωνικά Δίκτυα επιτρέπουν διακριτική εκπαίδευση με ένα φυσικό και αποτελεσματικό τρόπο. Ωστόσο, παρά την αποτελεσματικότητά τους στην κατάταξη των μονάδων μικρής διάρκειας, τα Νευρωνικά Δίκτυα είναι σπάνια επιτυχής για τα καθήκοντα συνεχούς αναγνώρισης, κυρίως λόγω της έλλειψης της ικανότητας να διαμορφώνουν χρονικές εξαρτήσεις [5].

3 Kaldi

3.1 Ιστορία

Το kaldi άρχισε την ύπαρξή του το 2009 στο εργαστήριο του πανεπιστημίου Johns Hopkins με τίτλο “Low Development Cost, High Quality Speech Recognition for New Languages and Domains”. Το επίκεντρο αυτού του έργου ήταν η μοντελοποίηση βασισμένη στο μοντέλο “Subspace Gaussian Mixture Model (SGMM)” και ορισμένες έρευνες για την εκμάθηση λεξικών. Το λογισμικό του kaldi άρχισε να αναπτύσσεται σε αυτό το εργαστήριο, αλλά η συνταγή που αναπτύχθηκε εκείνη την εποχή εξακολουθούσε να εξαρτάται από το HTK (Hidden Markov Model Toolkit). Οι συμμετέχοντες σε αυτό το εργαστήριο αναφέρονται παρακάτω (με αλφαβητική σειρά) [6]:

- Mohit Agarwal,
- Pinar Akyazi,
- Lukas Burget,
- Arnab Ghoshal,
- Ondrej Glembek,
- Nagendra Goel,
- Martin Karafiat,
- Feng Kai,
- Daniel Povey,
- Ariya Rastrow,
- Richard C. Rose,
- Petr Schwarz,
- Samuel Thomas.

Μερικοί από τους συμμετέχοντες σε αυτό το εργαστήριο, συμφώνησαν να συναντηθούν και πάλι το καλοκαίρι του 2010 στο Μπρνο της Τσεχίας (που φιλοξενήθηκε από το Τεχνολογικό Πανεπιστήμιο του Μπρνο). Στόχος αυτού του εργαστηρίου, ήταν να δημιουργήσει μια συνταγή βασισμένη στο έργο που έγινε το 2009, το οποίο ήταν καθαρό και αποδεσμεύσιμο, και να δημιουργήσει ένα εργαλείο ομιλίας γενικού σκοπού ως υποπροϊόν. Το πρόβλημα που

επιχειρήθηκε να λυθεί, ήταν ότι η προηγούμενη συνταγή βασιζόταν σε διαφορετικά σενάρια που περιελάμβαναν τόσο το HTK όσο και τον κώδικα "Kaldi". Αυτό είχε ως αποτέλεσμα, να μην είναι εύκολο να ενσωματωθεί. Θεωρήθηκε επίσης, ότι ένα καλά σχεδιασμένο, σύγχρονο εργαλείο ομιλίας γενικής χρήσης με ανοιχτή άδεια, θα αποτελούσε πλεονέκτημα για την κοινότητα αναγνώρισης ομιλίας. Τον Αύγουστο του 2010 συναντήθηκε η ακόλουθη ομάδα ανθρώπων στο Μπρνο για να εργαστεί σε αυτό το έργο (και πάλι αλφαβητικά) [6]:

- Pinar Akyazi,
- Lukas Burget,
- Gilles Boullianne,
- Ondrej Glembek,
- Arnab Ghoshal,
- Nagendra Goel,
- Mirko Hannemann,
- Petr Motlicek,
- Daniel Povey,
- Yanmin Qian,
- Petr Schwarz,
- Jan Silowsky, Georg Stemmer, and Karel Vesely.

Πολύ κώδικας γράφτηκε το καλοκαίρι του 2010, αλλά δεν υπήρχε ακόμα ένα πλήρες σύστημα εργασίας. Μερικοί από τους συμμετέχοντες στο σεμινάριο του 2010 συνέχισαν να εργάζονται για την ολοκλήρωση της δέσμης εργαλείων και για να αποκτήσουν ένα σύνολο λειτουργικών σεναρίων κατάρτισης. Ο κώδικας κυκλοφόρησε στις 14 Μαΐου 2011 [6].

Από την αρχική κυκλοφορία, το Kaldi διατηρείται και αναπτύσσεται σε μεγάλο βαθμό από τον Daniel Povey, ο οποίος εργαζόταν στη Microsoft Research μέχρι τις αρχές του 2012 και έκτοτε στο πανεπιστήμιο Johns Hopkins. Υπήρξαν φυσικά και σημαντικές συνεισφορές από άλλους: κυρίως τον Karel Vesely, ο οποίος ανέπτυξε το πλαίσιο κατάρτισης για το νευρωνικό δίκτυο και τον Arnab Ghoshal, ο οποίος συντονίζει το έργο της ακουστικής

μοντελοποίησης. Ο συνολικός αριθμός των ατόμων που έχουν συνεισφέρει κώδικα ή σεναρίων ή επιδιορθώσεων είναι περίπου 70 μέχρι στιγμής [6].

Σύμφωνα με το μύθο, το όνομα Kaldi προέρχεται από τον αιθίοπα βοσκό που ανακάλυψε το φυτό του καφέ [6].

3.2 Επισκόπηση του εργαλείου

Το Kaldi είναι ένα σύνολο εργαλειών ανοικτού κώδικα, για την αναγνώριση ομιλίας, γραμμένο σε C++ και έχει λάβει άδεια χρήσης βάση της άδειας Apache v2.0. Ο στόχος του Kaldi είναι να έχει σύγχρονο και ευέλικτο κώδικα, ο οποίος θα μπορεί να κατανοηθεί, να τροποποιηθεί και να επεκταθεί με σχετική ευκολία. Το kaldi είναι διαθέσιμο μέσω του SourceForge και του GitHub. Τα εργαλεία σχεδιάστηκα για να λειτουργούν ιδανικά σε συστήματα που μοιάζουν με Unix, ωστόσο μπορούν να χρησιμοποιηθούν και σε συστήματα που διαθέτουν λειτουργικό windows και ios.

Οι ερευνητές στην αυτόματη αναγνώριση ομιλίας (ASR), έχουν διάφορες δυνατότητες επιλογής εργαλείων ανοιχτού κώδικα για την κατασκευή ενός συστήματος αναγνώρισης. Αξιοσημείωτες μεταξύ αυτών είναι οι εξής:

- HTK, γραμμένο σε C
- Julius, γραμμένο σε C
- Sphinx-4, γραμμένο σε java
- RWTH ASR toolkit, γραμμένο σε C++.

Ωστόσο, κάποιες ειδικές απαιτήσεις, όπως είναι το πλαίσιο πεπερασμένης κατάστασης μετατροπέα (FST), η εκτεταμένη υποστήριξη γραμμικής άλγεβρας και η μη περιοριστική άδεια, οδήγησαν στην ανάπτυξη του kaldi. Κάποια από τα σημαντικά χαρακτηριστικά που περιλαμβάνει το kaldi είναι:

- Ενσωμάτωση σε επίπεδο κώδικα με μεταγωγείς πεπερασμένων καταστάσεων, χρησιμοποιώντας το OpenFst toolkit ως βιβλιοθήκη.
- Εκτεταμένη υποστήριξη γραμμικής άλγεβρας. Περιλαμβάνει μια πρότυπη βιβλιοθήκη, η οποία αναδιπλώνει τις τυπικές ρουτίνες BLAS

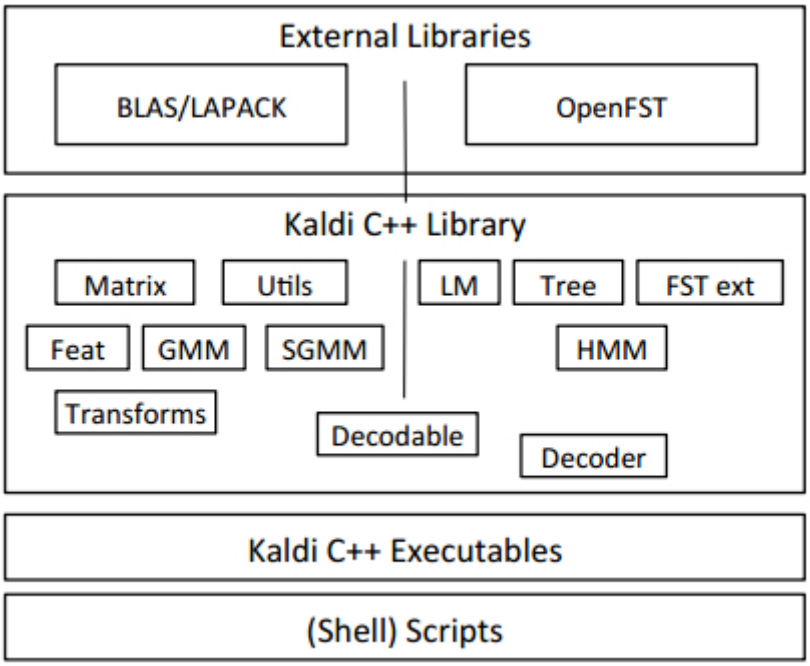
και LAPACK, για τις οποίες θα μιλήσουμε πιο αναλυτικά στο επόμενο κεφάλαιο.

- Επεκτάσιμος σχεδιασμός. Γίνεται προσπάθεια να παρέχονται οι αλγόριθμοι με τη γενικότερη δυνατή μορφή. Για παράδειγμα, οι αποκωδικοποιητές δουλεύουν με μια διεπαφή που παρέχει μια βαθμολογία για ένα συγκεκριμένο πλαίσιο και το σύμβολο εισαγωγής FST. Έτσι, ο αποκωδικοποιητής θα μπορούσε να λειτουργήσει από οποιαδήποτε κατάλληλη πηγή βαθμολογίας
- Ανοιχτή άδεια. Όπως προαναφέραμε, ο κώδικας έχει λάβει άδεια χρήσης κάτω από την άδεια Apache v2.0, η οποία είναι μια από τις λιγότερο περιοριστικές άδειες που είναι διαθέσιμες.
- Πλήρεις συνταγές. Στόχος είναι να διαθέτονται πλήρεις συνταγές για τη δημιουργία συστημάτων αναγνώρισης ομιλίας, τα οποία λειτουργούν από ευρέως διαθέσιμες βάσεις δεδομένων, όπως αυτές που παρέχονται από τη Κοινοπραξία Γλωσσικών Δεδομένων (Linguistic Data Consortium).
- Δυσμενές δοκιμές. Ο στόχος είναι να υπάρχουν οι αντίστοιχες ρουτίνες δοκιμής για όλο το κώδικα [7].

Ο στόχος της απελευθέρωσης πλήρων συνταγών είναι μια σημαντική πτυχή του Kaldi. Δεδομένου ότι ο κώδικας είναι διαθέσιμος δημοσίως με άδεια χρήσης που επιτρέπει τις τροποποιήσεις και την επανέκδοση, αυτοί που θα ασχοληθούν με το Kaldi, ενθαρρύνονται να απελευθερώσουν τον κώδικα τους, μαζί με τους καταλόγους σεναρίων, σε μορφή παρόμοια με αυτή του σεναρίου του Kaldi [6].

Η βασική χρήση του Kaldi είναι η έρευνα πάνω στην ακουστική μοντελοποίηση. Επομένως, βγαίνει το συμπέρασμα ότι οι πλησιέστεροι ανταγωνιστές είναι το HTK και το RWTH ASR toolkit (RASR). Το κύριο πλεονέκτημα έναντι του HTK είναι ο σύγχρονος, ευέλικτος και καθαρά δομημένος κώδικας, καθώς και η καλύτερη υποστήριξη WFST και μαθηματική υποστήριξη. Επίσης, οι όροι άδειας είναι πιο ανοιχτό από το HTK ή το RASR [7].

Δίνουμε μια σχηματική επισκόπηση του εργαλείου Kaldi στο παρακάτω σχήμα:



Εικόνα 2:Επισκόπηση του εργαλείου Kaldi

Το Kaldi εξαρτάται από δύο εξωτερικές βιβλιοθήκες που είναι επίσης ελεύθερα διαθέσιμες. Η μια είναι το OpenFst, για το πλαίσιο πεπερασμένης κατάστασης, και η άλλη είναι αριθμητική βιβλιοθήκη άλγεβρας. Για την ακρίβει, χρησιμοποιούνται οι πρότυπες “Basic Linear Algebra Subroutines” (BLAS) και “Linear Algebra PACKage” (LAPACK) [7].

Οι ενότητες της βιβλιοθήκης C++ του Kaldi, μπορούν να ομαδοποιηθούν σε δύο ξεχωριστά τμήματα, το καθένα από τα οποία εξαρτάται μόνο από μία από τις εξωτερικές βιβλιοθήκες (BLAS/LAPACK και OpenFst), όπως φαίνεται και στο σχήμα παραπάνω. Μια μεμονωμένη ενότητα, το DecodableInterface, συνδέει αυτά τα δύο τμήματα. Οι ενότητες που βρίσκονται χαμηλά στο σχήμα, εξαρτώνται από μια ή και περισσότερες ενότητες που βρίσκονται από πάνω τους [7].

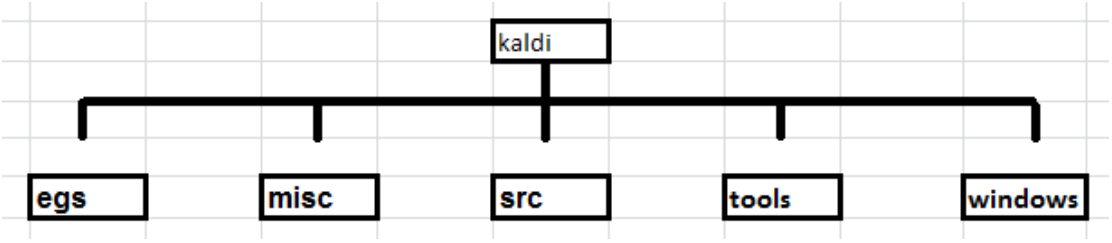
Πρόσβαση στις λειτουργίες της βιβλιοθήκης παρέχεται μέσω κάποιων εργαλειών της γραμμής εντολών, γραμμένα σε C ++, τα οποία στη συνέχεια

καλούνται από μια γλώσσα σεναρίου (script language) για τη δημιουργία και την εκτέλεση ενός συστήματος αναγνώρισης ομιλίας. Κάθε εργαλείο έχει πολύ συγκεκριμένη λειτουργικότητα με ένα μικρό σύνολο χαρακτηριστικών της γραμμής εντολών. Για παράδειγμα, υπάρχουν ξεχωριστά εκτελέσιμα για τη συγκέντρωση στατιστικών, για το άθροισμα των συσσωρευτών, και την ενημέρωση ενός ακουστικού μοντέλου βασισμένο σε GMM χρησιμοποιώντας την εκτίμιση μέγιστης πιθανότητας. Επιπλέον, όλα τα εργαλεία μπορούν να διαβάσουν από και να γράψει σε αγωγούς. Αυτό το γεγονός κάνει εύκολη την σύνδεση των διάφορων εργαλειών [7].

Για να αποφευχθεί η “σήψη του κώδικα”, έγινε προσπάθεια να δομηθεί το Kaldi με τέτοιο τρόπο, ώστε η εφαρμογή ενός νέου χαρακτηριστικού να συνοδεύεται και από τη προσθήκη νέου κώδικα και εργαλειών της γραμμής εντολών και όχι τη τροποποίηση των υπάρχοντων [7].

3.2.1 Ιεραρχία φακέλων

Αφού εγκαταστήσουμε το Kaldi με επιτυχία (τα αναλυτικά βήματα για την εγκατάσταση θα τα δούμε σε επόμενο κεφάλαιο), θα πρέπει να δημιουργηθεί ένας φάκελος με την ονομασία kaldi (root directory). Οι επιμέρους φάκελοι που βρίσκονται μέσα φάκελο Kaldi, φαίνονται στο παρακάτω σχήμα:



Εικόνα 3:Ιεραρχία φακέλων στο Kaldi

Σε αυτό το σημείο, θεωρούμε χρήσιμο να αναφέρουμε τι ακριβώς περιέχει ο κάθε φάκελος.

- egs: Ο συγκεκριμένος φάκελος, είναι και ο ποιο σημαντικός φάκελος για τη κατανόηση του Kaldi, περιέχει παραδείγματα σεναρίων, που μας επιτρέπουν να δημιουργήσουμε γρήγορα συστήματα αυτόματης αναγνώρισης ομιλίας (asr systems) για πάνω από 30 δημοφιλείς

εταιρίες ομιλίας. Επιπρόσθετα, επισυνάπτεται και τεκμηρίωση για κάθε έργο.

- misc: Σε αυτόν το φάκελο βρίσκονται πρόσθετα εργαλεία και εφόδια, που ωστόσο δεν είναι απαραίτητα για την ορθή λειτουργία του Kaldi.
- src: Αυτός ο φάκελος περιλαμβάνει τον πηγαίο κώδικα του Kaldi.
- tools: Στον φάκελο tools θα βρούμε χρήσιμα εξαρτήματα, καθώς και εξωτερικά εργαλεία. Όπως γίνεται αντιληπτό, θα μας διευκόλυνε πολύ εάν αποθηκεύαμε στο φάκελο αυτό όλα τα εξωτερικά εργαλεία που θα χρειαστεί να εγκαταστήσουμε, όπως είναι το srilm που αναφέραμε σε προηγούμενη ενότητα.
- windows: Τέλος, εδώ βρίσκονται τα εργαλεία που θα χρειαστούμε, εάν επιλέξουμε να εκτελέσουμε το Kaldi χρησιμοποιώντας Windows.

3.3 Εξαγωγή χαρακτηριστικών

Ο κώδικας που έχει γραφτεί και προορίζεται για την εξαγωγή χαρακτηριστικών και την ανάγνωση κυματομορφών, στοχεύει στη δημιουργία των τυπικών χαρακτηριστικών MFCC και PLP, ορίζοντας εύλογες προεπιλεγμένες τιμές, αφήνοντας όμως διαθέσιμες τις επιλογές που πιθανότατα να θελήσει κάποιος να τροποποιήσει (για παράδειγμα, τον αριθμό των mel bins, ελάχιστες και μέγιστες αποκοπές συχνότητας κ.λπ.). Το Kaldi υποστηρίζει τις πιο συχνά χρησιμοποιούμενες προσεγγίσεις εξαγωγής χαρακτηριστικών, όπως είναι για παράδειγμα το VTLN, μέση τιμή cepstral και η ομαλοποίηση διακύμανσης, LDA, STC / MLLT, HLDA και ούτω καθεξής [7].

3.4 Εκδόσεις

Κατά τη διάρκεια της ζωής του, το Kaldi είχε τρείς διαφορετικές μεθόδους έκδοσης. Αρχικά φιλοξενήθηκε στο Sourceforge και ήταν ένα έργο βασισμένο σε υποεκδόσεις. Έπειτα μεταφέρθηκε στο GitHub, και για κάποιο χρονικό διάστημα ο μόνος διαθέσιμος αριθμός έκδοσης ήταν ο Git hash του commit. Τον Ιανουάριο του 2017 εισήχθηκε ένα σχήμα έκδοσης αριθμών. Η πρώτη έκδοση του Kaldi ήταν η 5.0.0, σε αναγνώριση του γεγονότος ότι το σχέδιο είχε ήδη υπάρξει για αρκετό καιρό. Το βασικό σχήμα είναι (μεγάλο).(μικρό).(patch), αλλά ο αριθμός έκδοσης "patch" μπορεί επίσης να

περιλαμβάνει χαρακτηριστικά (συνήθως συμβατά). Ο "αριθμός patch" αυξάνεται αυτόματα κάθε φορά που μια δέσμευση για την Kaldi συγχωνεύεται στο github [6].

3.5 Προαπαιτούμενα

Το ιδανικό περιβάλλον για την εγκατάσταση και την εκτέλεση του Kaldi είναι ένα σύμπλεγμα μηχανών Linux (οποιαδήποτε μεγάλη διανομή) που εκτελεί Sun GridEngine (SGE), με πρόσβαση σε κοινόχρηστους καταλόγους μέσω του NFS ή κάποιου παρόμοιου συστήματος αρχείων δικτύου. Στην ιδανική περίπτωση, μερικοί υπολογιστές στο δίκτυο θα έχουν NVidia GPUs, οι οποίο μπορούν να χρησιμοποιηθούν για την κατάρτιση νευρωνικών δικτύων.

Το ελάχιστο υπολογιστικό περιβάλλον για την εκτέλεση του Kaldi είναι οποιοδήποτε περιβάλλον που μοιάζει με Unix. Επίσης, είναι δυνατό να εκτελεστεί σε ένα μόνο μηχάνημα, αν και φυσικά θα είναι πιο αργό και ίσως χρειαστεί να μειωθεί ο αριθμός των εργασιών που χρησιμοποιούνται σε ορισμένα από τα παραδείγματα σεναρίων για να αποφευχθεί η εξάντληση της μνήμης του μηχανήματος.

Το Kaldi ανταποκρίνεται καλύτερα σε Debian και Red Hat Linux, αλλά τρέχει σε οποιαδήποτε διανομή Linux, ή σε Cygwin ή Mac OsX [6].

3.6 Εγκατάσταση του εργαλείου

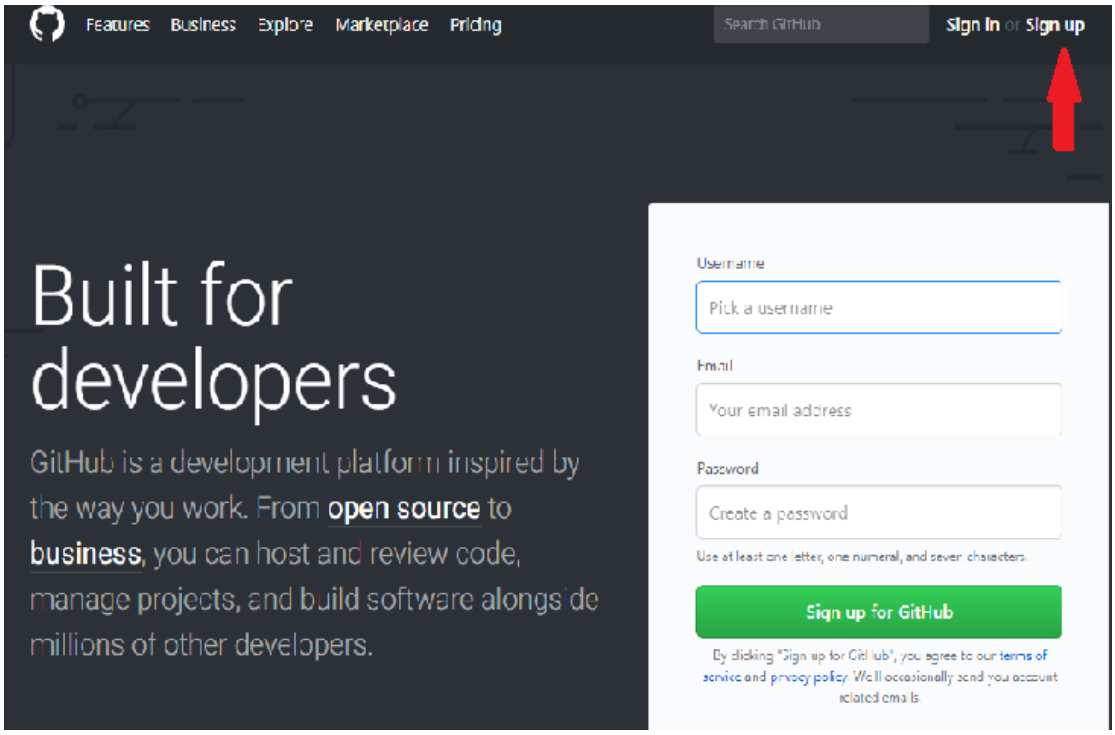
3.6.1 Εγκατάσταση github

Επειδή ο πηγαίος κώδικας του Kaldi βρίσκεται στο GitHub, θα πρέπει πρώτα να εγκαταστήσουμε το GitHub. Όπως αναφέραμε και παραπάνω, προτείνετε να το υπολογιστικό περιβάλλον να μοιάζει σε Unix. Ανοίγοντας, λοιπόν, ένα τερματικό σε περιβάλλον Unix, πληκτρολογούμε την παρακάτω εντολή:

```
sudo apt-get install git
```

Έπειτα, θα πρέπει να ανοίξουμε έναν φυλλομετρητή και να πάμε στην ιστοσελίδα του GitHub, που είναι η εξής:

https://github.com/



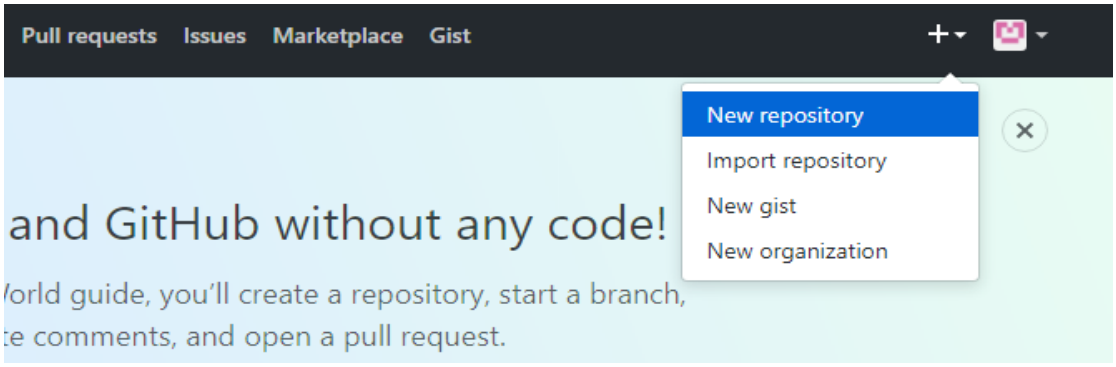
Εικόνα 4:Αρχική σελίδα GitHub

Στη συνέχεια, κάνουμε εγγραφή στο GitHub, δημιουργώντας έτσι έναν λογαριασμό. Τώρα πρέπει να κάνουμε κάποιες ρυθμίσεις. Για αυτό το σκοπό, επιστρέφουμε στο τερματικό και γράφουμε τις ακόλουθες εντολές:

git config –global user.name “username”

git config –global user.email “username@email.com”

όπου “username” και “username@email.com” το όνομα χρήστη μας και το email μας αντίστοιχα. Αφού εισέλθουμε με τα στοιχεία μας στο GitHub, δημιουργούμε ένα repository.



Εικόνα 5:Δημιουργία repository στο GitHub

Πάλι μέσα από το τερματικό, χρειαζόμαστε τις εντολές που φαίνονται παρακάτω:

mkdir test, όπου δημιουργούμε έναν φάκελο test

cd test, όπου πλοηγούμαστε μέσα στο φάκελο test

git init, όπου αρχικοποιούμε το φάκελο που μόλις δημιουργήσαμε

git remote add origin https://github.com/username/test.git, όπου του δύνουμε τη δυνατότητα να επικοινωνήσει με τον «έξω κόσμο».

Η εγκατάσταση του GitHub, θα πρέπει να έχει ολοκληρωθεί με επιτυχία.

3.6.2 Εγκατάσταση kaldi

Δουλεύοντας σε ένα Unix-like σύστημα, πληκτρολογούμε την εντολή

Git clone https://github.com/kaldi-asr/kaldi.git kaldi --origin upstream.

Με αυτό τον τρόπο, κλονοποιούμε το κώδικα που βρίσκεται στο GitHub, κατευθείαν στο τερματικό μας.

Στη συνέχεια θα πρέπει να εγκαταστήσουμε κάποιες βιβλιοθήκες και εργαλεία, που είναι απαραίτητα για να λειτουργήσει σωστά το Kaldi. Σε αυτό το σημείο, το Kaldi μας διευκολύνει κατά έναν μεγάλο βαθμό, διότι παρέχει μέσα σε έναν φάκελο τα απαραίτητα scripts, για να εγκαταστήσουμε στο σύστημά μας ότι χρειαζόμαστε. Μέσα από το τερματικο, πλοηγούμαστε στο κατάλληλο φάκελο ως εξής:

cd kaldi/tools/extras.

Μέσα στο φάκελο extras, υπάρχει το script check_dependencies.sh. Εάν εκτελεστεί το συγκεκριμένο script, ελέγχει τι χρειάζεται το σύστημά μας, και μας λέει λεπτομερώς ποια scripts να εκτελέσουμε για να εγκατασταθούν όλα τα προαπαιτούμενα προγράμματα. Στο δικό μας παράδειγμα, χρειάστηκε να τρέξω τις παρακάτω εντολές με τη σειρά που φαίνεται:

bash check_dependencies.sh

sudo apt-get install zlib1g-dev, βιβλιοθήκη C

sudo apt-get install auto make

sudo apt-get install auto conf

sudo apt-get install libtool, script που υποστηρίζει βιβλιοθήκες

sudo apt-get install subversion, μία έκδοση ανοικτού κώδικα του συστήματος ελέγχου έκδοσης

sudo apt-get install libatlas3-base, βιβλιοθήκη C

bash install_atlas.sh, βιβλιοθήκη γραμμικής άλγεβρας

sudo ln –s –f bash /bin/sh, δημιουργούμε έναν σύνδεσμο μεταξύ bash και sh

sudo apt-get install build-essential, C++ compiler

Στη συνέχεια θα πλοηγηθούμε ένα επίπεδο πάνω, και πιο συγκεκριμένα στο φάκελο tools, για να εκτελέσουμε την εντολή make:

cd ..

make

Τώρα πρέπει να πάμε στο φάκελο src και να τρέξουμε τις εξής εντολές:

./configure –shared

Make depend, όπου πραγματοποιείται η μεταγλώτιση

Make

Τέλος, χρειαζόμαστε και την εργαλειοθήκη srilm. Το srilm είναι ένα σύνολο εργαλειών, με τη βοήθεια των οποίων δημιουργούμε γλωσσικά μοντέλα. Το srilm με τη σειρά του, προαπαιτεί και αυτό κάποιες εφαρμογές, όπως είναι το Tcl developer xchange και τη βιβλιοθήκη ibc6-dev-amd64. Το Tcl developer xchange περιέχει το tool command language, που είναι μια πολύ ισχυρή, αλλά εύκολη στη μάθηση, δυναμική γλώσσα προγραμματισμού και το Tk, που είναι μια εργαλειοθήκη γραφικού περιβάλλοντος διεπαφής χρήστη. Για την

εγκατάσταση των παραπάνω εργαλειών, μέσα από ένα τερματικό εκτελούμε τις ακόλουθες εντολές:

sudo apt-get install tcl-dev tk-dev

sudo aptitude install libcb-dev-amd64

Το srilm το κατεβάζουμε από την ιστοσελίδα

http://www.speech.sri.com/projects/srilm/download.html

και ακολουθώντας τις οδηγίες που περιέχονται μέσα στο srilm, πραγματοποιούμε την εγκατάσταση.

4 Υποδειγματικό έργο

Σε αυτή την ενότητα, θα δούμε με τη σειρά τα βήματα που απαιτούνται για να δημιουργήσουμε ένα σύστημα ASR βασισμένο στα δικά μας δεδομένα ήχου. Για τις ανάγκες του έργου αυτού, ας υποθέσουμε ότι οι ομιλητές που θα ηχογραφηθούν, θα πρέπει να λένε τρείς λέξεις, με όποια σειρά και επαναληψιμότητα αυτοί θέλουν. Αυτές οι λέξεις θα είναι οι παρακάτω:

- qualify
- quality
- quantity

Επίσης, ας υποθέσουμε ότι οι ομιλητές θα είναι 3, και ότι ο καθένας θα ηχογραφηθεί 9 φορές. Για τις ανάγκες της συγκεκριμένης εργασίας θα ονομάσουμε τους τρείς ομιλητές bill, bob και nick. Οπότε προκύπτουν 27 αρχεία τύπου wav. Το κάθε αρχείο θα περιέχει, επομένως, τρείς λέξεις που θα καταγράφονται στην αγγλική γλώσσα. Κάθε ένα από αυτά τα αρχεία ήχου θα ονομάζεται με αναγνωρίσιμο τρόπο (π.χ. qualify_quality_quantity.wav, το οποίο θα περιέχει τη προφορική φράση qualify, quality, quantity) και θα τοποθετείται στον αναγνωρίσιμο φάκελο που αντιπροσωπεύει έναν συγκεκριμένο ομιλητή κατά τη διάρκεια μίας συγκεκριμένης περιόδου εγγραφής. Επομένως, για να συνοψίσουμε, το παραδειγματικό μας σύνολο δεδομένων μοιάζει με το παρακάτω:

- 3 διαφορετικοί ομιλητές
- Ο κάθε ομιλητής θα λέει 9 προτάσεις
- 27 εκφράσεις
- 81 λέξεις
- Η κάθε έκφραση αποτελείται από 3 λέξεις

Όπως προαναφέραμε και στην ενότητα ιεραρχία φακέλων, όλα τα έργα μας τα τοποθετούμε στο φάκελο egs. Δημιουργούμε λοιπόν μέσα στο φάκελο egs, ένα φάκελο με την ονομασία test (kaldi/egs/test), όπου θα τοποθετήσουμε όλα τα πράγματα που σχετίζονται με το έργο μας. Δημιουργούμε άλλον ένα φάκελο μέσα στο φάκελο test, με την ονομασία test_audio

(kaldi/egs/colors/test_audio). Μέσα στον test_audio, δημιουργούμε άλλους δυο φακέλους, τους train και test. Μέσα σε αυτούς τους φακέλους θα τοποθετήσουμε τα ηχογραφημένα δεδομένα, μέσα σε αναγνωρισμένους φακέλους που αντιπροσωπεύουν τους τρείς ομιλητές.

Τώρα πρέπει να δημιουργήσουμε κάποια αρχεία κειμένου που θα επιτρέψουν στο kaldi να επικοινωνήσει με τα ηχητικά μας δεδομένα. Κάθε αρχείο που θα δημιουργήσουμε για αυτό το σκοπό, καθώς και κάθε αρχείο γλωσσικής μοντελοποίησης που θα δημιοιυργήσουμε στη συνέχεια, μπορεί να θεωρηθεί ως αρχείο κειμένου με κάποιο αριθμό συμβολοσειρών (κάθε συμβολοσειρά σε μια νέα γραμμή). Αυτές οι συμβολοσειρές είναι αναγκαίο να ταξινομηθούν. Το kaldi μας προσφέρει κάποια εργαλεία για να το κάνουμε αυτό αυτόματα. Στο φάκελο kaldi/egs/test/utils (θα αναφέρουμε αργότερα πως θα δημιουργηθεί αυτός ο φάκελος και το περιεχόμενό του), υπάρχει το αρχείο validate_data_dir.sh, για τον έλεγχο της ταξινόμισης, καθώς και το αρχείο fix_data_dir.sh, για να ταξινομηθούν τα αρχεία που βρίσκονται μέσα σε έναν συγκεκριμένο φάκελο.

Πρακτικά, μέσα στο κατάλογο test, δημιουργούμε έναν νέο κατάλογο, με το όνομα data. Στη συνέχεια δημιουργούμε άλλους δυο υποκαταλόγους μέσα στο κατάλογο data, τους οποίους ονομάζουμε test και train. Επιπρόσθετα, δημιουργούμε σε κάθε έναν από τους υποφακέλους, που μόλις δημιουργήσαμε, τα ακόλουθα αρχεία.

- spk2gender
    - Αυτό το αρχείο μας ενημερώνει για το φύλο των ομιλητών. Ας υποθέσουμε ότι το speakerID είναι ένα μοναδικό όνομα κάθε ομιλιτή. Επειδή, όμως, ο κάθε ομιλητής έχει μόνο ένα φάκελο ακουστικών δεδομένων από μια συνεδρία εγγραφής, αποτελεί επίσης, και recordID. Στο παράδειγμά μας έχουμε τρείς ομιλητές αρσενικού γένους.
      Μοτίβο: <speakerID> <gender>

Bill m

Nick m

Πίνακας 1:Αρχείο spk2gender για εκπαίδευση

- wav.scp
  - Αυτό το αρχείο συνδέει κάθε φράση που λέγεται από ένα άτομο κατά τη διάρκεια συγκεκριμένης περιόδου εγγραφής, με ένα αρχείο ήχου που σχετίζεται με τη φράση αυτή. Αν ακολουθήσουμε την ονοματοδοσία, το utteranceID δεν είναι τίποτα περισσότερο από το speakerID (όνομα φακέλου ομιλητή), κολλημένο με το όνομα του αρχείου *.wav χωρίς τη κατάληξη .wav.
    Μοτίβο: <utteranceID> <full_path_to_audio_file>

bill_quality_qualify_qualify /home/kostas/kaldi/egs/test/test_audio/train/bill/quality_qualify_qualify.wav bill_quality_qualify_quality /home/kostas/kaldi/egs/test/test_audio/train/bill/quality_qualify_quality.wav bill_quality_qualify_quantity /home/kostas/kaldi/egs/test/test_audio/train/bill/quality_qualify_quantity.wav bill_quality_quality_qualify /home/kostas/kaldi/egs/test/test_audio/train/bill/quality_quality_qualify.wav bill_quality_quality_quality /home/kostas/kaldi/egs/test/test_audio/train/bill/quality_quality_quality.wav bill_quality_quality_quantity /home/kostas/kaldi/egs/test/test_audio/train/bill/quality_quality_quantity.wav bill_quality_quantity_qualify /home/kostas/kaldi/egs/test/test_audio/train/bill/quality_quantity_qualify.wav

bill_quality_quantity_quality
/home/kostas/kaldi/egs/test/test_audio/train/bill/quality_quantity_quality.wav
bill_quality_quantity_quantity
/home/kostas/kaldi/egs/test/test_audio/train/bill/quality_quantity_quantity.wav
nick_qualify_qualify_qualify
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_qualify_qualify.wav
nick_qualify_qualify_quality
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_qualify_quality.wav
nick_qualify_qualify_quantity
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_qualify_quantity.wav
nick_qualify_quality_qualify
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_quality_qualify.wav
nick_qualify_quality_quality
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_quality_quality.wav
nick_qualify_quality_quantity
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_quality_quantity.wav
nick_qualify_quantity_qualify
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_quantity_qualify.wav
nick_qualify_quantity_quality
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_quantity_quality.wav
nick_qualify_quantity_quantity
/home/kostas/kaldi/egs/test/test_audio/train/nick/qualify_quantit

y_quantity.wav

Πίνακας 2:Αρχείο wav.scp για εκπαίδευση

- text
  - Αυτό το αρχείο συνδέει κάθε φράση που λέγεται από ένα άτομο κατά τη διάρκεια συγκεκριμένης περιόδου εγγραφής, με την αναλυτική της γραφική αναπαράσταση.
    Μοτίβο: <utteranceID> <text_transcription>

bill_quality_qualify_qualify quality qualify qualify
bill_quality_qualify_quality quality qualify quality
bill_quality_qualify_quantity quality qualify quantity
bill_quality_quality_qualify quality quality qualify
bill_quality_quality_quality quality quality quality
bill_quality_quality_quantity quality quality quantity
bill_quality_quantity_qualify quality quantity qualify
bill_quality_quantity_quality quality quantity quality
bill_quality_quantity_quantity quality quantity quantity
nick_qualify_qualify_qualify qualify qualify qualify
nick_qualify_qualify_quality qualify qualify quality
nick_qualify_qualify_quantity qualify qualify quantity
nick_qualify_quality_qualify qualify quality qualify
nick_qualify_quality_quality qualify quality quality
nick_qualify_quality_quantity qualify quality quantity
nick_qualify_quantity_qualify qualify quantity qualify
nick_qualify_quantity_quality qualify quantity quality
nick_qualify_quantity_quantity qualify quantity quantity

Πίνακας 3:Αρχείο text για εκπαίδευση

- utt2spk
  - Αυτό το αρχείο αναφέρει στο σύστημα ASR ποια φράση ανήκει σε ποιόν ομιλητή.
    Μοτίβο: <utteranceID> <speakerID>

bill_quality_qualify_qualify bill

bill_quality_qualify_quality bill
bill_quality_qualify_quantity bill
bill_quality_quality_qualify bill
bill_quality_quality_quality bill
bill_quality_quality_quantity bill
bill_quality_quantity_qualify bill
bill_quality_quantity_quality bill
bill_quality_quantity_quantity bill
nick_qualify_qualify_qualify nick
nick_qualify_qualify_quality nick
nick_qualify_qualify_quantity nick
nick_qualify_quality_qualify nick
nick_qualify_quality_quality nick
nick_qualify_quality_quantity nick
nick_qualify_quantity_qualify nick
nick_qualify_quantity_quality nick
nick_qualify_quantity_quantity nick

Πίνακας 4:Αρχείο utt2spk για εκπαίδευση

- corpus.txt
    - Αυτό το αρχεία θα τοποθετηθεί σε εναν ελαφρώς διαφορετικό κατάλογο. Μέσα στο κατάλογο kaldi/egs/colors/data, δημιουργούμε το κατάλογο το κατάλογο με την ονομασία local, μέσα στον οποίο τοποθετούμε το αρχείο corpus.txt. Το συγκεκριμένο αρχείο θα περιέχει τη γραφική αναπαράσταση από κάθε φράση που μπορεί να εμφανιστείστο σύστημα ASR. Στο συγκεκριμένο παράδειγμα θα είναι 27 (18 στο φάκελο train και 9 στο φάκελο test) γραμμές από 27 αρχεία ήχου.

      Μοτίβο: <text_transcription>

qualify qualify qualify
qualify qualify quality
qualify qualify quantity
qualify quality qualify

qualify quality quality
qualify quality quantity
qualify quantity qualify
qualify quantity quality
qualify quantity quantity
quality qualify qualify
quality qualify quality
quality qualify quantity
quality quality qualify
quality quality quality
quality quality quantity
quality quantity qualify
quality quantity quality
quality quantity quantity
quantity qualify qualify
quantity qualify quality
quantity qualify quantity
quantity quality qualify
quantity quality quality
quantity quality quantity
quantity quantity qualify
quantity quantity quality
quantity quantity quantity

Πίνακας 5:Αρχείο corpus.txt για εκπαίδευση

Ως τελικό αποτέλεσμα της προηγούμενής μας ενέργειας, θα πρέπει να έχουμε τα παραπάνω αρχεία που ονομάζονταιι με τον ίδιο τρόπο στους υποφακέλους test και train, αλλά σχετίζονται με δυο διαφορετικά σύνολα δεδομένων. Τα παραπάνω παραδείγματα αναφέρονται στα αρχεία που θα χρησιμοποιηθούν για την εκπαίδευση του συστήματος. Άρα θα βρίσκονται μέσα στο φάκελο train. Τα αρχεία που θα χρησιμοποιηθούν για τον έλεγχο του συστήματος τα παραθέτουμε παρακάτω.

- spk2gender

Bob m

Πίνακας 6:Αρχείο spk2gender για έλεγχο

- wav.scp

bob_quantity_qualify_qualify /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_qualify_qualify.wav bob_quantity_qualify_quality /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_qualify_quality.wav bob_quantity_qualify_quantity /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_qualify_quantity.wav bob_quantity_quality_qualify /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_quality_qualify.wav bob_quantity_quality_quality /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_quality_quality.wav bob_quantity_quality_quantity /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_quality_quantity.wav bob_quantity_quantity_qualify /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_quantity_qualify.wav bob_quantity_quantity_quality /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_quantity_quality.wav bob_quantity_quantity_quantity /home/kostas/kaldi/egs/test/test_audio/test/bob/quantity_quantity_quantity.wav

Πίνακας 7:Αρχείο wav.scp για έλεγχο

- text

bob_quantity_qualify_qualify quantity qualify qualify
bob_quantity_qualify_quality quantity qualify quality
bob_quantity_qualify_quantity quantity qualify quantity
bob_quantity_quality_qualify quantity quality qualify
bob_quantity_quality_quality quantity quality quality
bob_quantity_quality_quantity quantity quality quantity
bob_quantity_quantity_qualify quantity quantity qualify
bob_quantity_quantity_quality quantity quantity quality
bob_quantity_quantity_quantity quantity quantity quantity

Πίνακας 8:Αρχείο text για έλεγχο

- utt2spk

bob_quantity_qualify_qualify bob
bob_quantity_qualify_quality bob
bob_quantity_qualify_quantity bob
bob_quantity_quality_qualify bob
bob_quantity_quality_quality bob
bob_quantity_quality_quantity bob
bob_quantity_quantity_qualify bob
bob_quantity_quantity_quality bob
bob_quantity_quantity_quantity bob

Πίνακας 9:Αρχείο utt2spk για έλεγχο

Στη συνέχεια, θα χρειαστούμε κάποια αρχεία γλωσσικής μοντελοποίησης. Επομένως, θα πλοηγηθούμε στο κατάλογο kaldi/egs/test/data/local, και θα δημιουργήσουμε τον υποκατάλογο dict. Στη συνέχεια, μέσα στο κατάλογο dict, θα δημιουργήσουμε μια σειρά από αρχεία.

- lexicon.txt
    - Αυτό το αρχείο συνδέει τη κάθε λέξη από το λεξικό μας, με τη γραφική αναπαράσταση των φωνημάτων της.
      Μοτίβο: <word> <phone> <phone>…

!SIL sil
<UNK> spn

qualify ku ah li fa hi
quality ku ah li ti
quantity ku an ti ti

Πίνακας 10:Αρχείο lexicon.txt

- nonsilence_phones.txt
  - Αυτό το αρχείο περιέχει όλα τα φωνήματα που χρησιμοποιήθηκαν στο συγκεκριμένο παράδειγμα.
    Μοτίβο: <phone>

ah
an
fa
hi
ku
li
ti

Πίνακας 11:Αρχείο nonsilence_phones.txt

- silence_phones.txt
  - Αυτό το αρχείο καταγράφει όλα τα φωνήματα σιωπής.
    Μοτίβο: <phone>

sil
spn

Πίνακας 12:Αρχείο silence_phones.txt

- optional_silence.txt
  - Αυτό το αρχείο περιέχει τα προαιρετικά φωνήματα σιωπής.
    Μοτίβο: <utteranceID> <speakerID>

sil

Πίνακας 13:Αρχείο optional_silence.txt

Αφού έχουμε κάνει όλα τα παραπάνω, πρέπει να προσθέσουμε τα απαραίτητα εργαλεία που μας παρέχει το kaldi, τα οποία χρησιμοποιούνται ευρέως σε υποδειγματικά σενάρια. Από το υποδειγματικό παράδειγμα wsj θα πάρουμε δυο φακέλους και θα τους τοποθετήσουμε στο δικό μας έργο. Από

kaldi/egs/wsj/s5 παίρνουμε τους φακέλους utils και steps, και τους τοποθετούμε στο kaldi/egs/test.

Στη συνέχεια, από το παράδειγμα voxforge, που υπάρχει και αυτό από την εγκατάσταση του kaldi, χρειαζόμαστε το script score.sh. Από kaldi/egs/voxforge/s5/local αντιγράφουμε το αρχείο score.sh στη τοποθεσία kaldi/egs/test/local. Το συγκεκριμένο αρχείο θα μας βοηθήσει να αποκτήσουμε τα αποτελέσματα αποκωδικοποίησης.

Αφού ολοκληρώσαμε τα παραπάνω βήματα, θα δημιουργήσουμε κάποια αρχεία ρυθμίσεων. Στο kaldi/egs/test δημιουργούμε το κατάλογο conf. Μέσα στο κατάλογο conf δημιουργούμε τα παρακάτω αρχεία.

- decode.config

```
first_beam=10.0
beam=13.0
lattice_beam=6.0
```

Πίνακας 14:Αρχείο decode.config

- mfcc.conf

```
--use-energy=false
```

Πίνακας 15:Αρχείο mfcc.conf

Να επισημάνουμε σε αυτό το σημείο ότι δεν είναι απαραίτητο να δημιουργήσουμε αρχεία ρυθμίσεων, αλλά μπορεί να είναι μια καλή συνήθεια για το μέλλον.

Επίσης να σημειώσουμε ότι χρησιμοποιήθηκαν δύο μέθοδοι εκπαίδευσης.

- MONO – μονοφωνική εκπαίδευση
- TRI1 – απλή εκπαίδευση τριφώνων

Πριν προχωρήσουμε στα επόμενα βήματα που απαιτούνται για την ολοκλήρωση του συστήματός μας, θεωρούμε απαραίτητο να αναφερθούμε πιο λεπτομερώς στις δυο αυτές μεθόδους εκπαίδευσης.

Τα μονοφωνικά μοντέλα είναι το πρώτο μέρος της διαδικασίας εκπαίδευσης. Επίσης, συνηθίζεται να εκπαιδεύουμε μόνο ένα υποσύνολο των δεδομένων, κυρίως για αποτελεσματικότητα. Τα περισσότερα μονοφωνικά μοντέλα μπορούν να δημιουργηθούν με λίγα δεδομένα, και στη συνέχεια αυτά τα μοντέλα χρησιμοποιούνται κυρίως για την εκπαίδευση νεότερων μοντέλων.

Γενικά, η προφορά μιας λέξης μπορεί να δοθεί ως μια σειρά συμβόλων που αντιστοιχεί στις μεμονωμένες μονάδες ήχου, οι οποίες συνθέτουν μια λέξη. Αυτές οι μονάδες ήχου ονομάζονται “φωνήματα”. Ένα μονοφώνιο αναφέρεται σε ένα μόνο φώνημα.

Η εκπαίδευση του μοντέλου τριφώνου περιλαμβάνει πρόσθετα χαρακτηριστικά για τον αριθμό των φύλλων ή των καταστάσεων HMM στο δέντρο αποφάσεων και στον αριθμό των Gaussians. Για παράδειγμα, ας υποθέσουμε ότι υπάρχουν 50 φωνήματα στο λεξικό μας. Θα μπορούσαμε να έχουμε μία κατάσταση HMM ανά φώνημα, αλλά γνωρίζουμε ότι τα φωνήματα θα διαφέρουν σημαντικά ανάλογα με το αν βρίσκονται στην αρχή, στο μέσο ή στο τέλος μιας λέξης. Ως εκ τούτου, θα θέλαμε τουλάχιστον τρεις διαφορετικές καταστάσεις HMM για κάθε φώνημα. Έτσι προκύπτουν τουλάχιστον 150 καταστάσεις HMM για τη μοντελοποίηση ακριβώς αυτού του σεναρίου.

Ο ακριβής αριθμός των φύλλων και των Gaussians συχνά αποφασίζεται ευρετικά. Οι αριθμοί θα εξαρτώνται σε μεγάλο βαθμό από την ποσότητα των δεδομένων, τον αριθμό των φωνητικών ερωτήσεων και το στόχο του μοντέλου. Υπάρχει επίσης ο περιορισμός, ότι ο αριθμός των Gaussians θα πρέπει πάντα να υπερβαίνει τον αριθμό των φύλλων.

Γενικά, ένα τρίφωνο είναι απλά μια ομάδα 3 φωνημάτων με τη μορφή “L + X + R”, όπου το φώνημα “L” (δηλαδή το αριστερό φώνημα) προηγείται του φωνήματος “X” και του φωνήματος “R”.

Αυτές οι δυο μέθοδοι αρκούν, για να αναδειχθούν αξιοσημείωτες διαφορές στα αποτελέσματα αποκωδικοποίησης.

Η τελευταία μας δουλειά για να ολοκληρώσουμε το σύστημα ASR, είναι να πρετοιμάσουμε τα σενάρια εκτέλεσης. Τα παρακάτω σενάρια βασίζονται στο

υποδειγματικό παράδειγμα voxforge. Επίσης, να αναφέρουμε ότι η τοποθεσία των παρακάτω αρχείων είναι η εξής:

- Kaldi/egs/test

- cmd.sh

```
# Setting local system jobs (local CPU - no external clusters)
export train_cmd=run.pl
export decode_cmd=run.pl
```

Πίνακας 16:Αρχείο cmd.sh

- path.sh

```
# Defining Kaldi root directory
export KALDI_ROOT=`pwd`/../..

# Setting paths to useful tools
export
PATH=$PWD/utils/:$KALDI_ROOT/src/bin:$KALDI_ROOT/tools/open
fst/bin:$KALDI_ROOT/src/fstbin/:$KALDI_ROOT/src/gmmbin/:$KALDI
_ROOT/src/featbin/:$KALDI_ROOT/src/lmbin/:$KALDI_ROOT/src/sg
mm2bin/:$KALDI_ROOT/src/fgmmbin/:$KALDI_ROOT/src/latbin/:$PW
D:$PATH

# Defining audio data directory (modify it for your installation
directory!)
export DATA_ROOT="/home/kostas/kaldi/egs/colors/colors_audio"

# Enable SRILM
source $KALDI_ROOT/tools/env.sh

# Variable needed for proper data sorting
export LC_ALL=C
```

Πίνακας 17:Αρχείο path.sh

- run.sh

```
#!/bin/bash

. ./path.sh || exit 1
. ./cmd.sh || exit 1

nj=1       # number of parallel jobs - 1 is perfect for such a small data set
lm_order=1 # language model order (n-gram quantity) - 1 is enough for digits grammar

# Safety mechanism (possible running this script with modified arguments)
. utils/parse_options.sh || exit 1
[[ $# -ge 1 ]] && { echo "Wrong arguments!"; exit 1; }

# Removing previously created data (from last run.sh execution)
rm -rf exp mfcc data/train/spk2utt data/train/cmvn.scp data/train/feats.scp data/train/split1 data/test/spk2utt data/test/cmvn.scp data/test/feats.scp data/test/split1 data/local/lang data/lang data/local/tmp data/local/dict/lexiconp.txt

echo
echo "===== PREPARING ACOUSTIC DATA ====="
echo

# Needs to be prepared by hand (or using self written scripts):
#
# spk2gender  [<speaker-id> <gender>]
# wav.scp     [<uterranceID> <full_path_to_audio_file>]
# text        [<uterranceID> <text_transcription>]
# utt2spk     [<uterranceID> <speakerID>]
# corpus.txt  [<text_transcription>]
```

```
# Making spk2utt files
utils/utt2spk_to_spk2utt.pl data/train/utt2spk > data/train/spk2utt
utils/utt2spk_to_spk2utt.pl data/test/utt2spk > data/test/spk2utt

echo
echo "===== FEATURES EXTRACTION ====="
echo

# Making feats.scp files
mfccdir=mfcc
# Uncomment and modify arguments in scripts below if you have any problems with data sorting
# utils/validate_data_dir.sh data/train     # script for checking prepared data - here: for data/train directory
  # utils/fix_data_dir.sh data/train          # tool for data proper sorting if needed - here: for data/train directory
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/train exp/make_mfcc/train $mfccdir
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/test exp/make_mfcc/test $mfccdir

# Making cmvn.scp files
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train $mfccdir
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test $mfccdir

echo
echo "===== PREPARING LANGUAGE DATA ====="
echo

# Needs to be prepared by hand (or using self written scripts):
```

```
#
# lexicon.txt           [<word> <phone 1> <phone 2> ...]
# nonsilence_phones.txt    [<phone>]
# silence_phones.txt    [<phone>]
# optional_silence.txt  [<phone>]

# Preparing language data
utils/prepare_lang.sh data/local/dict "<UNK>" data/local/lang data/lang

echo
echo "===== LANGUAGE MODEL CREATION ====="
echo "===== MAKING lm.arpa ====="
echo

loc=`which ngram-count`;
if [ -z $loc ]; then
   if uname -a | grep 64 >/dev/null; then
         sdir=$KALDI_ROOT/tools/srilm/bin/i686-m64
   else
         sdir=$KALDI_ROOT/tools/srilm/bin/i686
   fi
   if [ -f $sdir/ngram-count ]; then
         echo "Using SRILM language modelling tool from $sdir"
         export PATH=$PATH:$sdir
   else
         echo "SRILM toolkit is probably not installed.
            Instructions: tools/install_srilm.sh"
         exit 1
   fi
fi

local=data/local
```

```
mkdir $local/tmp
ngram-count -order $lm_order -write-vocab $local/tmp/vocab-full.txt -wbdiscount -text $local/corpus.txt -lm $local/tmp/lm.arpa

echo
echo "===== MAKING G.fst ====="
echo

lang=data/lang
arpa2fst --disambig-symbol=#0 --read-symbol-table=$lang/words.txt $local/tmp/lm.arpa $lang/G.fst

echo
echo "===== MONO TRAINING ====="
echo

steps/train_mono.sh --nj $nj --cmd "$train_cmd" data/train data/lang exp/mono  || exit 1

echo
echo "===== MONO DECODING ====="
echo

utils/mkgraph.sh --mono data/lang exp/mono exp/mono/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd "$decode_cmd" exp/mono/graph data/test exp/mono/decode

echo
echo "===== MONO ALIGNMENT ====="
echo

steps/align_si.sh --nj $nj --cmd "$train_cmd" data/train data/lang
```

```
exp/mono exp/mono_ali || exit 1

echo
echo "===== TRI1 (first triphone pass) TRAINING ====="
echo

steps/train_deltas.sh --cmd "$train_cmd" 2000 11000 data/train
data/lang exp/mono_ali exp/tri1 || exit 1

echo
echo "===== TRI1 (first triphone pass) DECODING ====="
echo

utils/mkgraph.sh data/lang exp/tri1 exp/tri1/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd
"$decode_cmd" exp/tri1/graph data/test exp/tri1/decode

echo
echo "===== run.sh script is finished ====="
echo
```

Πίνακας 18:Αρχείο run.sh

Αφού έχουμε ολοκληρώσει τα παραπάνω βήματα με επιτυχία, μπορούμε να τρέξουμε το σύστημά μας. Για να γίνει αυτό, θα πρέπει να ανοίξουμε τη γραμμή εντολών και να πλοηγηθούμε στο φάκελο kaldi/egs/test. Έπειτα τρέχουμε το αρχείο cmd.sh, το οποίο είναι απαραίτητο για τη ρύθμιση εργασιών τοπικού συστήματος. Στη συνέχεια, θα πρέπει να τρέξουμε το αρχείο path.sh, με το οποίο ορίζουμε στο σύστημά μας τα paths για το root directory, για κάποια χρήσιμα εργαλεία, για τα δεδομένα ήχου, καθώς ενεργοποιούμε καιτο srilm, το οποίο είναι ένα εργαλείο για τη δημιουργία γλωσσικών μοντέλων. Τέλος, έχουμε να εκτελέσουμε το βασικό αρχείο εκτέλεσης, run.sh.

Για να κατανοήσουμε καλύτερα τον τρόπο λειτουργίας ενός συστήματος αυτόματης αναγνώρισης ομιλίας, μπορούμε να διαχωρίσουμε το συγκεκριμένο αρχείο σε έξι στάδια. Ως πρώτο στάδιο, μπορούμε να θεωρήσουμε τη προετοιμασία των ακουστικών δεδομένων. Σε αυτό το στάδιο, συμπεριλαμβάνονται όλες οι ενέργειες που έχουν πραγματοποιηθεί για τη προετοιμασία των ακουστικών δεδομένων. Για παράδειγμα, τα αρχεία spk2gender, wav.scp, text, utt2spk, corpus.txt συμπεριλαμβάνονται στο συγκεκριμένο στάδιο.

Στο δεύτερο στάδιο, το σύστημά μας εξάγει τα απαραίτητα χαρακτηριστικά. Ποιο συγκεκριμένα, υπολογίζει το συντελεστή mfcc.

Στο τρίτο στάδιο γίνεται η προετοιμασία των γλωσσικών δεδομένων. Αυτό συμβαίνει, αρχικά με τη δημιουργία των αρχείων lexicon.txt, nonsilence_phones.txt, silence_phones.txt και optional_silence.txt, και στη συνέχεια το kaldi μας παρέχει ένα εκτελέσιμο αρχείο prepare_lang.sh, το οποίο είναι υπεύθυνο για τη προετοιμασία των γλωσσικών δεδομένων.

Στο τέταρτο στάδιο, και με τη βοήθεια του εργαλείου srilm, δημιουργείτε το γλωσσικό μοντέλο.

Ως πέμπτο στάδιο θεωρούμε τη μονοφωνική εκπαίδευση, καθώς και την αποκρυπτογράφηση των αποτελεσμάτων, που ακολουθεί της εκπαίδευσης. Για τη μονοφωνική εκπαίδευση, το kaldi μας παρέχει το αρχείο train_mono.sh, και για την αποκρυπτογράφηση το αρχείο decode.sh. Να πούμε σε αυτό το σημείο, ότι το αρχείο train_mono.sh δέχεται ως παραμέτρους τα paths των φακέλων, που περιέχουν τα δεδομένα που προορίζονται για εκπαίδευση, καθώς και τα γλωσσικά δεδομένα. Επίσης, ορίζουμε και έναν φάκελο (exp/mono) όπου θα τοποθετηθούν τα αποτελέσματα στη φάση της αποκρυπτογράφησης.

Στο τελευταίο στάδιο, πραγματοποιείτε η τριφωνική εκπαίδευση, καθώς και η αποκρυπτογράφησή της. Όπως και στη μονοφωνική εκπαίδευση, έτσι και εδώ το kaldi μας παρέχει το κατάλληλο αρχείο, για να πραγματοποιηθεί η τριφωνική εκπαίδευση. Αυτό το αρχείο είναι το train_delta.sh και βρίσκεται στο

φάκελο steps. Το συγκεκριμένο αρχείο, πέρα από τις παραμέτρους που δεχόταν και το αρχείο train_mono.sh, δέχεται και τον αριθμό των καταστάσεων HMM, που στη προκειμένη περίπτωση είναι 2000. Στη συνέχει, για την αποκρυπτογράφηση των αποτελεσμάτων χρησιμοποιείτε πάλι το αρχείο decode.sh.

Εκτός από το γεγονός ότι θα παρατηρήσουμε κάποια αποκωδικοποιητικά αποτελέσματα στο παράθυρο του τερματικού, μπορούμε να πάμε στο φάκελο kaldi/egs/test/exp, όπου περιέχονται λεπτομερώς τα αποκωδικοποιητικά αποτελέσματα. Παρατηρούμε εκεί τους φακέλους mono και tri1. Η δομή των καταλόγων αυτών είναι ίδια. Εάν πλοηγηθούμε στο κατάλογο mono/decode μπορούμε να βρούμε τα αρχεία αποτελεσμάτων (που ονομάζονται με τρόπο wer_ {αριθμός}). Τα αρχεία καταγραφής για τη διαδικασία αποκωδικοποίησης μπορούν να βρεθούν στο φάκελο log, ο οποίος βρίσκεται στον ίδιο κατάλογο.

Στο δικό μας παράδειγμα, όπου χρησιμοποιήθηκε ο ένας ομιλητής (bob) για να αξιολογήσουμε το σύστημά μας, παρουσιάστηκε ποσοστό λάθους ανά λέξη και ανά πρόταση ίσο με το μηδέν.

5 Συμπεράσματα

Φτάνοντας στο τέλος αυτής της πτυχιακής εργασίας θα ήταν σωστό να γίνει μία αναφορά επιγραμματικά στις καταστάσεις όπως εξελίχθηκαν κατά την εκπόνησή της.

Τα θετικά σημεία της πτυχιακής αυτής είναι οι γνώσεις που αποκομίσθηκαν στην γλώσσα προγραμματισμού python, στην εργαλειοθήκη kaldi, στη γραμμή εντολών σε περιβάλλον unix, αλλά φυσικά και στις επιμέρους εφαρμογές και βιβλιοθήκες που χρησιμοποιήθηκαν για να εκπονηθεί η συγκεκριμένη εργασία.

Οι αρχικές ιδέες που είχαν τεθεί ως στόχοι τηρήθηκαν και το αποτέλεσμα είναι η επιτυχής σχεδίαση και υλοποίηση ενός συστήματος αυτόματης αναγνώρισης ομιλίας.

Μετά από πολλές δοκιμασίας διαπιστώθηκε ότι η αναγνώριση ομιλίας είναι πολύ ικανοποιητική σε ιδανικές συνθήκες, για ένα μικρό σύνολο δεδομένων.

Σαν ιδανικές συνθήκες θεωρείται ένας κλειστός χώρος (π.χ. γραφείο) χωρίς εξωτερικό θόρυβο, αλλά ούτε και εσωτερικό θόρυβο κοντά στο μικρόφωνο. Ένα παράδειγμα εσωτερικού θορύβου του γραφείου μπορεί να θεωρηθεί ο ανεμιστήρας ψύξης ενός φορητού υπολογιστή.

6 Βιβλιογραφία

[1] Macherey, W. (2010), *Discriminative Training and Acoustic Modeling for Automatic Speech Recognition*.

[2] Rosillo Gil, V. (2016), *Automatic speech recognition with kaldi toolkit*.

[3] Δασκαλάκης, Μ. (2015), *Ανάπτυξη εφαρμογής με χρήση μηχανών αναγνώρισης ομιλίας*.

[4] Felix, T. (2008), *Dynamic Programming Algorithms in Speech Recognition*.

[5] Wikipedia (2017), *Artificial Neural Networks*.

[6] Povey, D. (2011) kaldi-asr.org.

[7] Povey, D. (2011), *The Kaldi Speech Recognition Toolkit*.

[8] Popovic, B. et al. (2015), *Deep Neural Network Based Continuous Speech Recognition for Serbian Using the Kaldi Toolkit.*

[9] Ροδομαγουλάκης, Ι. Ποταμιάνος, Γ. Μαραγγός, Π. (2013), *Advances in large vocabulary continuous speech recognition in Greek: Modeling and nonlinear features.*