ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Επεξεργασία Φωνής & Φυσικής Γλώσσας

Ροή Σ: Σήματα, Έλεγχος, Ρομποτική

8ο Εξάμηνο

2η Εργαστηριακή Άσκηση

Αναγνώριση Φωνής με το Kaldi Toolkit

Χρήστος Δημόπουλος

03117037

1) Σκοπός Άσκησης

Σκοπός της άσκησης αυτής είναι η υλοποίηση ενός συστήματος επεξεργασίας και αναγνώρισης φωνής με το εργαλείο *Kaldi*. Πιο συγκεκριμένα, το σύστημα που θα αναπτύξουμε αφορά σε αναγνώριση φωνημάτων (***Phone Recognition***) από ηχογραφήσεις της USC-TIMIT. Χρησιμοποιούνται δεδόμενα audio από 4 διαφορετικούς ομιλητές, με τα αντίστοιχα transcriptions, ώστε να εκπαιδεύσουμε και να εκτιμήσουμε το σύστημά σας.

Η διαδικασία σχεδιασμού του συστήματος μπορεί να χωριστεί σε 4 μέρη. Το πρώτο μέρος αποσκοπεί στην εξαγωγή κατάλληλων ακουστικών χαρακτηριστικών από τα φωνητικά δεδομένα (***Mel-Frequency Cepstral Coefficients***). Τα εν λόγω χαρακτηριστικά είναι στην ουσία ένας αριθμός συντελεστών cepstrum που εξάγονται μετά από ανάλυση των σημάτων σημάτων φωνής με μια ειδικά σχεδιασμένη συστοιχία φίλτρων (Mel filterbank). Η συστοιχία αυτή είναι εμπνευσμένη από το μη γραμμικό τρόπο που το ανθρώπινο αυτί αντιλαβάνεται τον ήχο και ειδικά σχεδιασμένη από ψυχοακουστικές μελέτες.

Το δεύτερο μέρος αφορά τη δημιουργία **γλωσσικών μοντέλων** από τα transcriptions του σετ δεδομένων, τα οποία θα δίνουν την a priori πιθανότητα στο τελικό σύστημα.

Το τρίτο μέρος αφορά την **εκπαίδευση** των ακουστικών μοντέλων χρησιμοποιώντας τα ακουστικά χαρακτηριστικά τα οποία εξήχθησαν.

Τέλος, συνδυάζοντας τις παραπάνω μονάδες μπορεί να κατασκευαστεί το τελικό σύστημα αναγνώρισης φωνής, το οποίο δεδομένου ενός σήματος φωνής, εξάγει τα ακουστικά χαρακτηριστικά και τα χρησιμοποιεί ώστε να αποκωδικοποιήσει το σήμα σε μία ακολουθία φωνημάτων ή λέξεων.

2) Θεωρητικό Υπόβαθρο

Στη συνέχεια, αναλύουμε εν συντομία το θεωρητικό υπόβαθρο που απαιτείται για τη διεξαγωγή της εργαστηριακής άσκησης.

Mel-Frequency Cepstrum Coefficients (MFCCs)

Το Mel-Frequency Cepstrum είναι η αναπαράσταση του φασματογραφήματος ενός σήματος, η οποία προκύπτει με εφαρμογή Μετασχηματισμού Συνημιτόνου (DCT) στον λογάριθμο του Φάσματος Ισχύος του σήματος, πάνω στη μη γραμμική κλίμακα Mel, και συνιστά τεχνική **Αυτόματης Αναγνώρισης Φωνής** (ASR). Η εν λόγω αναπαράσταση επιτυγχάνει να προσομοιώσει καλύτερα το μη γραμμικό τρόπο που το ανθρώπινο αυτί αντιλαμβάνεται τον ήχο και περιγράφεται συνοπτικά από τους συντελεστές MFCCs.

Προκειμένου να ανακτήσουμε του συντελεστές MFCC ενός ηχητικού σήματος ακολουθούμε την εξής διαδικασία:

1. Αρχικά, εφαρμόζουμε στο σήμε ένα **pre-emphasis φίλτρο**, ώστε να ενισχύσουμε τις υψίσυχνες συνιστώσες του. Με αυτόν τον τρόπο, εξισορροπούμε το φάσμα του σήματος, καθώς οι υψηλές συχνότητες τείνουν να έχουν μικρότερο πλάτος από τις χαμηλές, αποτρέπουμε αριθμητικά προβλήματα στον υπολογισμό του Μετασχηματισμού Fourier και βελτιώνουμε τον σηματοθορυβικό λόγο (SNR). Συνήθως, χρησιμοποιούμε φίλτρο $1^{ης}$ τάξης με συντελεστή α περίπου 0.96.

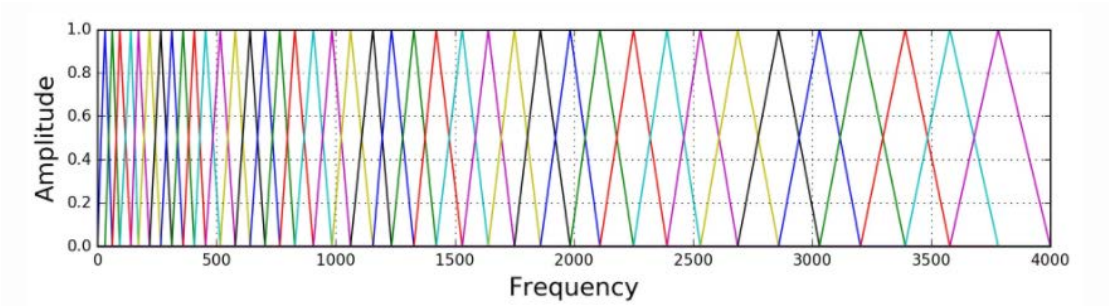
2. Προβαίνουμε στην ανάλυση βραχέος χρόνου του σήματος, απομονώντας χρονικά **παράθυρα** του. Για την ανάλυση αυτή, χρησιμοποιούμε παράθυρα Hamming.

3. Στη συνέχεια, για κάθε χρονικό παράθυρο, παίρνουμε τον Μετασχηματισμό Fourier του παραθυρομένου σήματος (**STFT**) και στη συνέχεια υπολογίζουμε το φάσμα ισχύος (**Periodogram**) ως εξής:

$$P = \frac{|DFT(xi)|^2}{N}$$

Όπου xi είναι το ι-οστό πλαίσιο του σήματος x και N το συνολικό πλήθος παραθύρων.

4. Ακολούθως, εφαρμόζουμε στα φάσματα ισχύος που λάβαμε μια συστοιχία τριγωνικών επικαλυπτόμενων φίλτρων στην κλίμακα Mel (**Triangular Mel Filterbank**).



Mel Filterbank

5. Τέλος, στον **λογάριθμο** των φασμάτων ισχύος εφαρμόζουμε Διακριτό Μετασχηματισμό Συνημιτόνων (**DCT**), αποσυσχετίζοντας τη συστοιχία φίλτρων, ώστε να λάβουμε τους cepstral συντελεστές, που συνιστούν μια συμπιεσμένη αναπαράσταση του σήματος. Τυπικά κρατιέται ένας αριθμός συντελεστών (12-13), ενώ 'πετιούνται' οι υπόλοιποι, καθώς δεν συνεισφέρουν σημαντικά στο πρόβλημα ASR.

Ωστόσο, οι τιμές MFCC δεν είναι ισιαίτερα εύρωστες παρουσία προσθετικού θορύβου, γι'αυτό συνήθως χρειάζεται κάποια μορφή κανονικοποίησης. Μια αποτελεσματική μέθοδος κανονικοποίησης είναι η λεγόμενη ***Cepstral Mean & Variance Normalization*** (CMVN), που προκύπτει με αφαίρεση της μέσης τιμής του σήματος και διαίρεση με την τυπική του απόκλιση:

$$\widehat{x_t}(i) = \frac{x(i) - \mu(i)}{\sigma(i)}$$

όπου $x_t(i)$το $i-$οστό πλαίσιο του σήματος τη χρονική στιγμή t.
μ(i) η μέση τιμή και σ(i) τυπική απόκλιση.

Γλωσσικό Μοντέλο (Language Model)

Ως γλωσσικό μοντέλο θεωρούμε μια πιθανοτική κατανομή πάνω σε μια ακολουθία λέξεων. Δεδομένης μιας ακολουθίας λέξεων w μήκους n, δηλαδή, ανατίθεται η πιθανότητα $P(w_1 w_2 \ldots w_n)$ σε ολόκληρη την ακολουθία. Το γλωσσικό μοντέλο προσφέρει ανάλυση με βάση τα **συμφραζόμενα (context sensitive)**, και έτσι επιτυγχάνεται διάκριση ανάμεσα σε λέξεις και προτάσεις που είναι **ομόηχες**. Η πιθανότητα εμφάνισης μιας ακολουθίας n λέξεων w_i , i = 1,2,…, n μπορεί να αναλυθεί με βάση τον κανόνα αλυσίδας πιθανοτήτων ως:

$$P(w_1 w_2 \ldots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \ldots P(w_n|w_1 w_2 \ldots w_{n-1})$$

Το πιο απλό γλωσσικό μοντέλο είναι το **n-gram model**, σύμφωνα με το οποίο κάθε λέξη εξαρτάται μόνο από τις N προηγούμενες της (Ιδιότητα Έλλειψης Μνήμης Markov):

$$P(w_i|w_1 w_2 \ldots w_{i-1}) = P(w_i|w_{i-n+1} \ldots w_{i-1})$$

Ειδικές περιπτώσεις του παραπάνω μοντέλου, είναι τα μοντέλα **unigram** και **bigram**, όπου υπάρχει εξάρτηση από καμία και ακριβώς μια προηγούμενη λέξη αντιστοίχως. Οι πιθανότητες εμφάνισης ακολουθιών λέξεων συνηθίζεται να παρουσιάζονται ως κόστη με χρήση του **λογαρίθμου** τους (-logP) και υπολογίζονται με βάση τον κανόνα **Maximum Likelihood Estimation**. Για παράδειγμα, για το bigram μοντέλο έχουμε:

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1} w_i)}{\#(w_{i-1}\, *)}$$

Κύριο μέτρο εκτίμησης της επίδοσης ενός γλωσσικού μοντέλου πάνω σε ένα test set W = $w_1 w_2 \ldots w_n$, είναι η πολυπλοκότητα (**Perplexity**), η οποία ορίζεται ως:

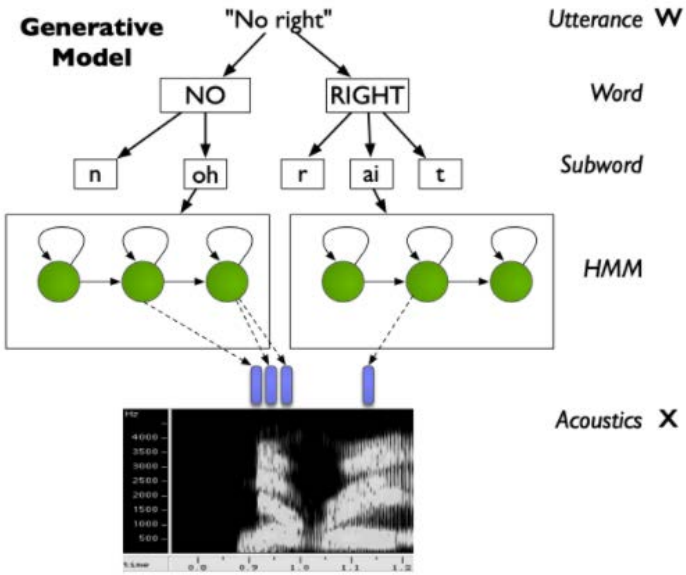
$$PP(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Για την αντιμετώπιση περιπτώσεων, όπου ορισμένες λέξεις είναι εξαιρετικά σπάνιες και δεν εμφανίζονται συχνά υπό συγκεκριμένες ακολουθίες – άρα οδηγούν σε μηδενικές πιθανότητες – χρησιμοποιούμε διάφορες τεχνικές, όπως **backoff**, **add-k smoothing** και **interpolation**, οι οποίες αναλύονται εκτενώς στο βιβλίο των Jurafski & Martin [1].

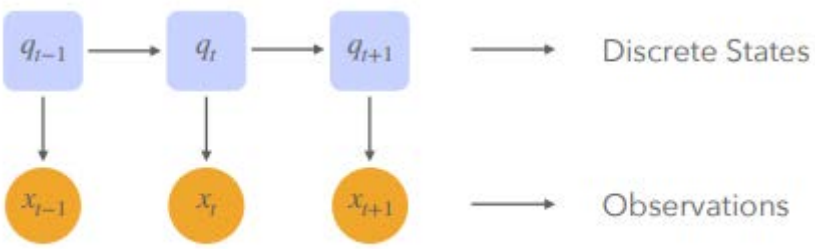
Ακουστικό Μοντέλο (Acoustic Model)

Σε ένα πρόβλημα ASR το ακουστικό μοντέλο αναπαριστά τη σχέση ενός ηχητικού σήματος με τα φωνήματα που το απαρτίζουν. Δημιουργείται παίρνοντας μια μεγάλη βάση δεδομένων από ηχητικά σήματα ομιλητών (speech corpus) και δημιουργώντας με τη χρήση αλγορίθμων εκπαίδευσης στατιστικές αναπαραστάσης κάθε φωνήματος της γλώσσας των ομιλητών. Οι εν λόγω αναπαραστάσεις ονομάζονται **Hidden Markov Models (HMM)**, ενώ κάθε φώνημα περιγράφεται από μια εξ αυτών.

Μια πρόταση W, λοιπόν, αρχικά διασπάται σε λέξεις και στη συνέχεια σε φωνήματα, ενώ κάθε φώνημα αναπαρίσταται από μια HMM. Κάθε HMM αποτελείται από μια πληθώρα κρυφών καταστάσεων και παράγει ορισμένα γνωρίσματα με βάση ένα Γκαουσιανό Μίγμα (**Gaussian Mixture Model – GMM**) για κάθε κατάσταση. Τα γνωρίσματα αυτά συνιστούν τα ακουστικά χαρακτηριστικά X της πρότασης. [2]



Ένα *HMM* παράγει τις παρατηρήσεις x_i σύμφωνα με μια πιθανοτική κατανομή (Gaussian) της κατάστασης στη οποία ανήκει, ενώ υποθέτουμε **ανεξαρτησία** μεταξύ παρατήρησης x_i και άλλων καταστάσεων. Παραμετροποιείται σύμφωνα με τις πιθανότητες μετάβασης μεταξύ καταστάσεων $a_{jk} = P(q_t = j|q_{t-1} = k)$, τις αρχικές πιθανότητες $\pi_j = P(q_0 = j)$ και τις πιθανότητες των παρατηρήσεων $b_j(x) = P(x|q = j)$.



Ένα *GMM* που περιγράφει κάθε κατάσταση του Κρυφού Μαρκοβιανού Μοντέλου συνιστά ένα μίγμα Γκαουσσιανών, δηλαδή ένα άθροισμα Μ συνιστωσών από Gaussian Πυκνότητες Πιθανότητας, πολλαπλασιαζόμενες με αντίστοιχα βάρη. Δηλαδή:

$$P(x|\lambda) = \sum_{k=1}^{M} w_k N(x|\mu_k, \Sigma_k)$$

Όπου:

$$\sum_{k=1}^{M} w_k = 1$$

$N(x|\mu_k, \Sigma_k)$: *Γκαουσσιανή πυκνότητα πιθανότητας μέσης τιμή* μ_k *και διασποράς* Σ_k

3) Βήματα Προπαρασκευής

Τα βήματα προπαρασκευής της εργαστηριακής άσκησης έχουν αναλυθεί λεπτομερώς σε προηγούμενη αναφορά. Ενδεικτικά, υπενθυμίζουμε τη δομή των directories που δημιουργούνται:

```
.
├─── data
│   ├─── dev
│   │   ├─── text
│   │   ├─── utt2spk
│   │   ├─── uttids -> /home/{user}/slp_lab2_data/filesets/validation_utterances.txt
│   │   └─── wav.scp
│   ├─── lexicon.txt -> /home/{user}/slp_lab2_data/lexicon.txt
│   ├─── test
│   │   ├─── text
│   │   ├─── utt2spk
│   │   ├─── uttids -> /home/{user}/slp_lab2_data/filesets/test_utterances.txt
│   │   └─── wav.scp
│   ├─── train
│   │   ├─── text
│   │   ├─── utt2spk
│   │   ├─── uttids -> /home/{user}/slp_lab2_data/filesets/train_utterances.txt
│   │   └─── wav.scp
│   └─── transcription.txt -> /home/{user}/slp_lab2_data/transcription.txt
├─── prelab2.py
└─── wav -> /home/{user}/slp_lab2_data/wav
```

Όπου {user}: το όνομα της τοπικής μηχανής που εκτελείται το script '*prelab2.py*'.

4) Βήματα Κυρίως Μέρους

4.1 Προετοιμασία διαδικασίας Αναγνώρισης Φωνής για τη USC-TIMIT

Για την πραγματοποιήση των βημάτων προετοιμασίας της διαδικασίας αναγνώρισης φωνής της USC-TIMIT με χρήση του λογισμικού Kaldi αρχικά εκτελούμε το αρχείο **4.1.makedirs.sh**. Με χρήση αυτού, υλοποιούμε τα εξής:

- Αρχικά, από τη διαδικασία wsj του kaldi πήραμε τα αρχεία path.sh και cmd.sh. Στο πρώτο θέσαμε τη μεταβλητή KALDI_ROOT στο directory που βρίσκεται ο κύριος φάκελος της εγκατάστασης του Kaldi και το κάνουμε source σε κάθε bash script μας, ώστε να έχουμε διαθέσιμες όλες τις εντολές του Kaldi. Στο δεύτερο αλλάζουμε τις τιμές των μεταβλητών train_cmd, decode_cmd, cuda_cmd σε run.pl.
- Δημιουργούμε soft links μέσα στον φάκελο usc με ονόματα 'steps' και 'utils', το οποία δείχνουν στους αντίστοιχους φακέλους της wsj.
- Δημιουργούμε τον φάκελο local και μέσα σε αυτόν ένα soft link που δείχνει στο αρχείο score_kaldi.sh εντός του steps. Το αρχείο αυτό ονομάζεται *score.sh*, ώστε να λειτουργήσει στη συνέχεια το evaluation των GMM-HMMs.
- Δημιουργούμε τον φάκελο conf και μέσα σε αυτόν μεταφέρουμε το αρχείο mfcc.conf το οποίο περιλαμβάνει τα configurations των MFCCs.
- Τέλος, δημιουργούμε τα directories data/lang, data/local/dict, data/local/lm_tmp, data/local/nist_lm και ολοκληρώνουμε το στάδιο προετοιμασίας.

4.2 Προετοιμασία Γλωσσικού Μοντέλου

Τα βήματα προετοιμασίας του γλωσσικού μοντέλου εκτελούνται στο bash script '**4.2.makelm.sh**'. Ειδικότερα:

1) Στον φάκελο data/local/dict αποθηκεύουμε τα βασικά αρχεία που χρησιμεύουν για τη δημιουργία του γλωσσικού μοντέλου:
   - Τα αρχεία silence_phone.txt και optional_silence.txt που περιέχουν το φώνημα της σιωπής sil.
   - Το αρχείο nonsilence_phones.txt το οποίο περιέχει όλα τα υπόλοιπα φωνήματα ανά γραμμή και ταξινομημένα. Παρήχθει με την εντολή:
     - *cut -d ' ' -f 2- lexicon.txt | \ sed 's/ /\n/g' | \ sort -u > nonsilence_phones.txt.*
   - Το αρχείο lexicon.txt το οποίο αποτελεί λεξικό του γλωσσικού μοντέλου και συνιστά 1-1 αντιστοίχιση κάθε φωνήματος με τον εαυτό του. Περιλάμβάνει το φώνημα σιωπής sil και το ειδικό σύμβολο <oov> (out of vocabulary).
   - Τα αρχεία lm_train.txt, lm_test.txt, lm_dev.txt που αποτελούν αντιγραφές των αντίστοιχων text αρχείων της προπαρασκευής με του ειδικούς χαρακτήρες <s> και </s> στην αρχή και στο τέλος κάθε πρότασης αντίστοιχα.
   - Το κενό αρχείο extra_questions.txt.

2) Στη συνέχεια, με χρήση της εντολής **build-lm.sh** του πακέτου IRSTLM, μέσα στον φάκελο data/local/lm_tmp δημιουργούμε την ενδιάμεση μορφή από unigram και bigram γλωσσικά μοντέλα κατάληξης .ilm.gz. Τα γλωσσικά μοντέλα αποθηκεύονται ως lm_phone_ug.ilm.gz και lm_phone_bg.ilm.gz αντίστοιχα.

3) Έπειτα, στον φάκελο data/local/nist_lm αποθηκεύουμε τα compiled γλωσσικά μοντέλα σε μορφή ARPA, χρησιμοποιώντας την εντολή **compile-lm**:
   - *compile-lm ./data/local/lm_tmp/lm_phone_ug.ilm.gz -t=yes /dev/stdout | grep -v unk | gzip -c > ./data/local/nist_lm/lm_phone_ug.arpa.gz*
   - *compile-lm ./data/local/lm_tmp/lm_phone_bg.ilm.gz -t=yes /dev/stdout | grep -v unk | gzip -c > ./data/local/nist_lm/lm_phone_bg.arpa.gz*

   Τα γλωσσικά μοντέλα αποθηκεύονται ως lm_phone_ug.arpa.gz και lm_phone_bg.arpa.gz αντίστοιχα.
4) Ακολούθως, μέσα στον φάκελο data/lang δημιουργούμε το FST του λεξικού της γλώσσας (L.fst) χρησιμοποιώντας την εντολή του Kaldi **prepare_lang.sh**:
   - *./prepare_lang.sh data/local/dict '<oov>' data/local/lm_tmp data/lang*
5) Τα αρχεία utt2spk, text και wav.scp είναι ήδη ταξινομημένα ως προς utterance ID και ομιλητή f1, f5, m1, m3, οπότε δεν χρειάζεται κάποι επιπλέον ενέργεια.
6) Εκτελούμε το script utils/utt2spk_to_spk2utt.pl προκειμένου να δημιουργήσουμε τα αρχεία spk2utt, που αντιστοιχούν καθένα εκ των ομιλητών με τα utterances του για τα set train, dev, test.
7) Τέλος, δημιουργούμε το FST της γραμματικής (G.fst), με χρήση της εντολής ./timit_format_data.sh. Ως αποτέλεσμα, δημιουργούνται οι φάκελοι data/lang_phone_ug και data/lang_phones_bg για το unigram και bigram μοντέλο αντιστοίχως.

Ερώτημα 1: Για τα γλωσσικά μοντέλα που δημιουργήσατε υπολογίστε το perplexity στο validation και στο test set. Τι δείχνουν αυτές οι τιμές;

Κάνοντας χρήσης της εντολής compile-lm του Kaldi, μέσα στο script **compute_perp.sh**, υπολογίζουμε το perplexity για τα unigram και bigram γλωσσικά μοντέλα, πάνω στα set dev και test. Τα αποτελέσματα αποθηκεύονται στο directory usc/perplexity και συνοψίζονται στους ακόλουθους πίνακες:

Perplexity (PP)	Dev Set	Test Set
Unigram Model	54.66	54.47
Bigram Model	26.86	26.56

Out Of Voc (%)	Dev Set	Test Set
Unigram Model	2.8	2.89
Bigram Model	2.8	2.89

Παρατηρούμε, λοιπόν, ότι η **επίδοση του bigram γλωσσικού μοντέλου είναι αρκετά καλύτερη από αυτή του unigram** – σχεδόν μισό perplexity – ενώ γενικά και τα δύο μοντέλα **αποδίδουν καλύτερα στο test set αντί του validation set**. Επίσης, όπως είναι λογικό, το ποσοστό άγνωστων λέξεων (oov%) εξαρτάται μόνο από το set που γίνεται η ανάλυση (dev ή test) και όχι από τα γλωσσικά μοντέλα. Αυτό που διαφοροποιεί την επίδοση των μοντέλων έχει να κάνει κυρίως με το **penalty** που επιβάλλει το καθένα εξ αυτών για κάθε λέξη εκτός λεξιλογίου που συναντά.

4.3 Εξαγωγή Ακουστικών Χαρακτηριστικών

Στη συνέχεια, εκτελώντας το script **4.3.mfcc.sh** εξάγουμε τα ακουστικά χαρακτηριστικά για καθένα εκ των τριών σετ train, dev, test με χρήση των εντολών *make_mfcc.sh* και *compute_cmvn_stats.sh.*

Ως αποτέλεσμα, σε κάθε ένα από τα τρία σετ δημιουργούνται τα αρχεία feats, frame_shift, utt2dur, utt2num_frames με πληροφορίες σχετικά με τα χαρακτηριστικά των MFCCs και το αρχείο cmvn, σχετικό με το Cepstrum Mean & Variance Normalization. Επιπλέον, δημιουργούνται οι φάκελοι .backup, conf, log, data και split4.

Ερώτημα 2: Με τη δεύτερη εντολή πραγματοποιείται το λεγόμενο Cepstral Mean and Variance Normalization. Τι σκοπό εξυπηρετεί;

Το CMVN συνιστά τεχνική κανονικοποιήσης για τη δημιουργία ενός εύρωστου συστήματος αναγνώρισης φωνής, με στόχο την αντιμετώπιση των αλλοιώσεων που προκαλεί ο θόρυβος. Πιο συγκεκριμένα, ο προσθετικός στάσιμος θόρυβος (AWGN) τροποποιεί τις κατανομές των MFCCs, μετατοπίζοντας τις μέσες τιμές τους και συρρικνώνοντας τη διασπορά τους. Προκειμένου να αντιμετωπιστεί αυτό, και να έχουν οι cepstral συντελεστές **κοινές στατιστικές ιδιότητες**, η τεχνική CMVN μετασχηματίζει γραμμικώς τις cepstral κατανομές, ώστε να έχουν **μηδενική μέση τιμή και μοναδιαία διασπορά**.

Προτού προβούμε στην εκπαίδευση και επαλήθευση του μοντέλου, λοιπόν, κανονικοποιούμε κάθε διάνυσμα ακουστικών χαρακτηριστικών σύμφωνα με τον τύπο:

$$\widehat{x_t}(i) = \frac{x(i) - \mu_t(i)}{\sigma_t(i)}$$

όπου $x_t(i)$ *η* $i-$*οστή συνιστώσα του αρχικού σήματος τη χρονική στιγμή* t.
μ(i) η μέση τιμή και σ(i) η τυπική απόκλιση της ι-οστής συνιστώσας, υπολογιζόμενες σε ένα ολισθώμενο παράθυρο μήκους N:

$$\mu_t(i) = \frac{1}{N} \sum_{n=t-N/2}^{t+\frac{N}{2}-1} x_n(t)$$

$$\sigma_t(i) = \frac{1}{N} \sum_{n=t-N/2}^{t+\frac{N}{2}-1} (x_n(t) - \mu_t(i))^2$$

Ερώτημα 3: Πόσα ακουστικά frames εξήχθησαν για κάθε μία από τις 5 πρώτες προτάσεις του training set; Τι διάσταση έχουν τα χαρακτηριστικά;

Διαβάζοντας τις 5 πρώτες γραμμές του αρχείου **utt2num_frames** για το training set, εξάγουμε το πλήθος των ακουστικών frames των 5 πρώτων προτάσεων. Ειδικότερα, κάθε γραμμή περιλαμβάνει το utteranceID της πρότασης και τον αριθμό των ακουστικών της frames:

usctimit_ema_f1_001 237

usctimit_ema_f1_002 377

usctimit_ema_f1_003 317

usctimit_ema_f1_005 399

usctimit_ema_f1_006 338

Έπειτα, προκειμένου να προσδιορίσουμε τη **διάσταση** των χαρακτηριστικών, αρχικά βλέπουμε τη διάρκεια ολίσθησης ενός frame κάθε φορά στο αρχείο frame_shift (0.01 sec δηλαδή **10 msec**). Επομένως, η διάσταση ενός μητρώου χαρακτηριστικών είναι:

$$dimension = \left(\frac{Duration}{Frame\ Shift}\right) x\ 13$$

Όπου:

Frame Shift: η διάρκεια ολίσθησης του frame δηλαδή 0.01sec.

Duration: Η διάρκεια του utterance σε δευτερόλεπτα, η οποία μπορεί να φανεί στο αρχείο utt2dur.

Ο αριθμός 13 συνιστά το πλήθος των συντελεστών που κρατιέται για μια αρκετά ικανοποιητική αναπαράσταση του Mel-Frequency Cepstrum.

4.4 Εκπαίδευση Ακουστικών Μοντέλων και Αποκωδικοποίηση Προτάσεων

Στη συνέχεια, εκτελώντας το script **4.4.hmm.sh** εκπαιδεύουμε monophone & triphone GMM-HMM ακουστικά μοντέλα, συσχετίζοντας τα ηχητικά σήματα των utterances με τα φωνήματα που τα απαρτίζουν. Πιο συγκεκριμένα, ακολουθούμε την εξής διαδικασία:

1) Εκπαιδεύουμε ένα monophone GMM-HMM acoustic model πάνω στο training set με χρήση της εντολής train_mono.sh του Kaldi. Το μοντέλο αποθηκεύεται στο directory exp/mono:
   - . ./steps/train_mono.sh data/train data/lang exp/mono
2) Έπειτα, δημιουργούμε τον γράφο HCLG του Kaldi, σύμφωνα με τη γραμματική G.fst, δοκιμάζοντας unigram και bigram μοντέλα. Χρησιμοποιώντας την εντολή **mkgraph.sh,** φτιάχνουμε τους γράφους για τα unigram και bigram models και αποθηκεύουμε το αποτέλεσμα στα directories exp/mono_graph_ug και exp/mono_graph_bg αντίστοιχα.
3) Στη συνέχεια, αποκωδικοποιούμε τις προτάσεις των validation και test sets με τον αλγόριθμο Viterbi, χρησιμοποιώντας την εντολή decode.sh:
   ./steps/decode.sh exp/mono_graph_ug data/dev exp/mono/decode_dev_ug

./steps/decode.sh exp/mono_graph_bg data/dev exp/mono/decode_dev_bg
./steps/decode.sh exp/mono_graph_ug data/test exp/mono/decode_test_ug
./steps/decode.sh exp/mono_graph_bg data/test exp/mono/decode_test_bg

4) Ως μέτρο εκτίμησης της επίδοσης της αποκωδικοποιήσης των μοντέλων μας χρησιμοποιούμε τη μετρική Phone Error Rate (PER):

$$PER = 100\frac{insertions + deletions + substitutions}{\#phones}$$

Τα αποτελέσματα του decoding για τα unigram και bigram models πάνω στα σετ dev και test αποθηκεύονται στα directories exp/mono/decode_{dev/test}_{ug/bg} και συνοψίζονται στον κάτωθι πίνακα:

PER (%)	Dev Set	Test Set
Unigram Model	%WER 51.64 [3188 / 6173, 76 ins, 1789 del, 1323 sub]	%WER 50.03 [2998 / 5992, 59 ins, 1675 del, 1264 sub]
Bigram Model	%WER 46.25 [2855 / 6173, 123 ins, 1273 del, 1459 sub]	**%WER 43.21** [2589 / 5992, 104 ins, 1130 del, 1355 sub]

Μια από τις υπερπαραμέτρους που επηρεάζουν το decoding είναι το λεγόμενο beam. Ειδικότερα, για την αποκωδικοποίηση ενός ήχου διασχίζουμε έναν weighted γράφο, ώστε να βρούμε το πιο πιθανό μονοπάτι από φωνήματα. Ο γράφος αυτός είναι πολύ μεγάλος και χρειάζεται εκθετικό χρόνο για να διασχιστεί πλήρως. Για τον λόγο αυτό, ''κλαδεύουμε'' ορισμένα μονοπάτια του και κρατάμε τα N πιο πιθανά. Αυτό συνιστά το **beam** και επιτυγχάνεται με χρήση **lattices**. Όσο μικρότερο beam, τόσο λιγότερα μονοπάτια εξερευνούμε, οπότε μειώνεται ο συνολικός χρόνος decoding, αλλά ως tradeoff αυξάνεται η πιθανότητα εσφαλμένης αποκωδικοποίησης. Το μέγεθος του beam search καθορίζεται από τις εξής παραμέτρους (bigram model -test set):

--max-active=7000 --beam=13.0 --lattice-beam=6.0 --acoustic-scale=0.083333

Η παράμετρος **max-active** καθορίζει τον μέγιστο αριθμό καταστάσεων του HMM που μπορούν να είναι ενεργοποιημένες ταυτόχρονα και ελέγχει το beam pruning με χρήση weight cutoffs.

Μια επιπλέον παράμετρος είναι το λεγόμενο **mean** που αφορά τις στατιστικές ιδιότητες των φωνημάτων και σχετίζεται με το CMVN που εφαρμόσαμε στο προηγούμενο βήμα (μέση τιμή).

Στο καλύτερο μοντέλο μας (bigram model – test set) οι τιμές των παραμέτρων μας φαίνονται στο αρχείο exp/mono/decode_test_bg/log/analyse_lattice_depth_stats.log και είναι συνολικά:

Overall, lattice depth (10,50,90-percentile)=(3,18,112) and mean=64.3

5) Ως τελευταίο βήμα, κάνουμε alignment των φωνημάτων χρησιμοποιώντας το monophone μοντέλο, με την εντολή **align_si.sh**, αποθηκεύοντας το αποτέλεσμα στο directory exp/mono_ali. Έπειτα, χρησιμοποιώντας αυτά τα alignment, εκπαιδεύουμε ένα triphone μοντέλο με χρήση της εντολής **train_deltas.sh**. Το triphone model αποθηκεύται στον φάκελο exp/tri1 και ακολουθώντας την ίδια διαδικασία με το monophone μοντέλο, υπολογίζουμε τον γράφο HCLG του triphone model και εκτελούμε εκ νέου αποκωδικοποίηση για τα σετ dev και test.
Τα αποτελέσματα του decoding για το **triphone** model, συνοψίζονται στον ακόλουθο πίνακα:

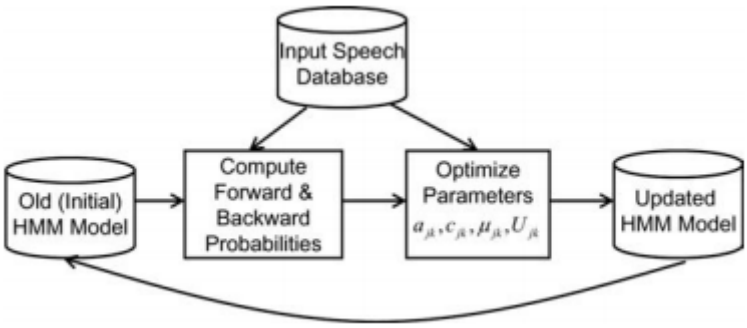
PER (%)	Dev Set	Test Set
Unigram Model	%WER 37.00 [2284 / 6173, 252 ins, 711 del, 1321 sub]	%WER 35.53 [2129 / 5992, 256 ins, 595 del, 1278 sub]
Bigram Model	%WER 33.82 [2088 / 6173, 222 ins, 584 del, 1282 sub]	**%WER 32.08** [1922 / 5992, 237 ins, 465 del, 1220 sub]

Παρατηρούμε, δηλαδή, ότι το triphone μοντέλο **αποδίδει αρκετά καλύτερα** σε σχέση με το monophone, με τις καλύτερες επιδόσης του μοντέλου να εντοπίζονται στο test set με χρήση bigram language model. Αξίζει να σημειωθεί, επίσης, ότι σε αντίθεση με το monophone μοντέλο, το Phone Error Rate στο triphone μοντέλο δεν διαφοροποιείται σημαντικά ανάμεσα στην επιλογή bigram και unigram γλωσσικών μοντέλων.

Ερώτημα 4: Εξηγήστε τη δομή ενός ακουστικού μοντέλου GMM-HMM. Τι σκοπό εξυπηρετούν τα μαρκοβιανά μοντέλα στη συγκεκριμένη περίπτωση και τι τα μίγματα γκαουσιανών; Με ποιό τρόπο γίνεται η εκπαίδευση ενός τέτοιου μοντέλου; Περιγράψτε τη διαδικασία εκπαίδευσης ενός μονοφωνικού μοντέλου.

Η δομή ενός GMM-HMM ακουστικού μοντέλου αναλύθηκε εκτενώς και στο θεωρητικό υπόβαθρο της εργαστηριακής αναφοράς. Όπως αναφέρθηκε κάθε κατάσταση του HMM χαρακτηρίζεται από μία μίξη Gaussian πυκνοτήτων πιθανοτήτων, η οποία αναπαριστά τη στατιστική συμπεριφορά των διανυσμάτων ακουστικών χαρακτηριστικών X_t, στις καταστάσεις του μοντέλου. [3]

Προκειμένου να γίνει η εκπαίδευση του HMM μιας λέξης (ή ενός φωνήματος) χρησιμοποιείται ένα επισημειωμένο σύνολο προτάσεων εκπαίδευσης (που μεταγράφονται σε μονάδες λέξεων και υπο-λέξεων) για να οδηγηθεί μια αποτελεσματική διαδικασία εκπαίδευσης που είναι γνωστή ως **Αλγόριθμος Baum-Welch ή Forward-Backward Algorithm**. Ο αλγόριθμος αυτός στοιχίζει κάθε HMM λέξης ή φωνήματος με την ομιλία εισόδου και στη συνέχεια προβαίνει στην εκτίμηση των κατάλληλων μέσων, διασπορών και βαρών μίξης για τις Γκαουσσιανές κατανομές σε κάθε κατάσταση του μοντέλου, με βάση την τρέχουσα στοίχιση. Η μέθοδος Baum-Welch είναι **hill climbing** αλγόριθμος και λειτουργεί **επαναληπτικά** έως ότου επιτευχθεί μια σταθερή στοίχιση μοντέλων και διανυσμάτων χαρακτηριστικών του σήματος φωνής. [3]

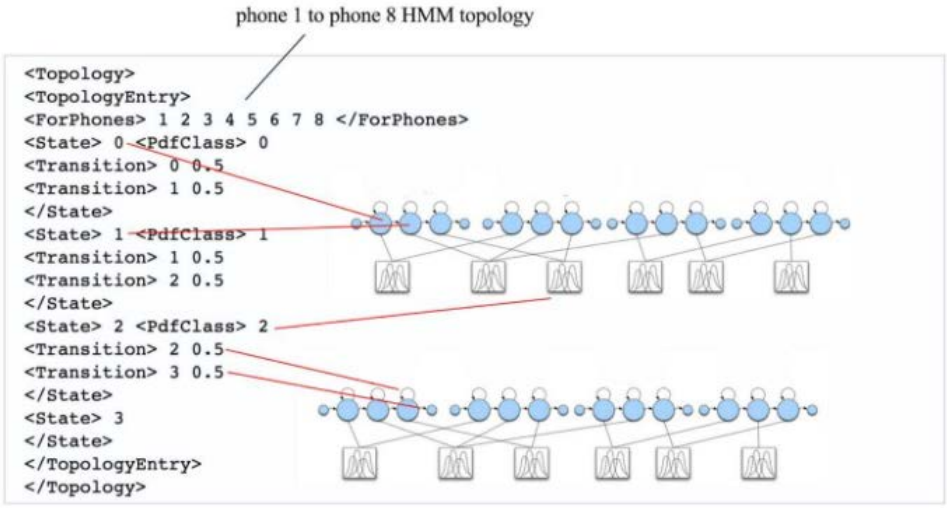


The Baum–Welch training procedure based on a given training set of utterances.

Προκειμένου να κατανοήσουμε καλύτερα την εκπαίδευση του **monophone HMM**, δηλαδή μοντέλου που δεν λαμβάνει υπόψη τα συμφραζόμενα (context-independent model), αναλύουμε τη διαδικασία που ακολουθεί το Kaldi κατά την εκτέλεση της εντολής train_mono.sh.

Κατά την αρχικοποίηση της διαδικασίας, δημιουργείται ένα απλό δέντρο απόφασης φωνημάτων χωρίς καμία διακλάδωση (**flat phonetic decision tree without any split**).

Ύστερα, δημιουργείται η **τοπολογία** του μοντέλου για τα φωνήματα στο αρχείο data/lang/topo. Σε κάθε γραμμή του αρχείου, η πρώτη κατάσταση είναι η αρχική και η τελευταία η τελική χωρίς μεταβάσεις εξερχόμενες από αυτήν. Η τοπολογία αποτελείται από 3 internal states και από 3 emitting states, που καθορίζουν τα διανύσματα χαρακτηριστικών. Για παράδειγμα, για τα φωνήματα 1-8 έχουμε:



Στη συνέχεια, το Kaldi μεταγλωττίζει τον γράφο εκπαίδευσης για να επιταχύνει τη διαδικασία του training αργότερα. Ως αποτέλεσμα, δημιουργείται ένα FST για κάθε training utterance, τα οποία συνολικά απαρτίζουν τη δομή του HMM (**FST Graph**). Ως σύμβολα εισόδου έχουν τα αναγνωριστικά των μεταβάσεων (transition IDs) που περιλαμβάνουν τα αναγνωριστικά των πυκνοτήτων πιθανότητας του GMM, ενώ ως σύμβολα εξόδου έχουμε λέξεις. Ως κόστη θεωρούνται οι πιθανότητες προφοράς στο λεξικό.

Ακολούθως, το Kaldi πραγματοποιεί το πρώτο **alignment**, στο οποίο υποθέτει πως κάθε HMM κατάσταση καλύπτει τον ίδιο αριθμό από audio frames. Για να το επιτύχει αυτό, χρησιμοποιεί τον **αλγόριθμο Viterbi** (δείτε παρακάτω) αντί του forward-backward και μετά **επανεκτιμά** τα GMM acoustic models.

Μόλις ολοκληρωθεί η διαδικασία για πρώτη φορά, πραγματοποιούνται **επαναλήψεις** για την εκπαίδευση του μοντέλου. Μέσα στο loop, γίνονται τα εξής:

- Align phone states according to the GMM models
- Accumulate stats for GMM training
- Perform Maximum Likelihood to re-estimate the GMM-based acoustic model.

Μετά από μια πληθώρα iterations, ολοκληρώνεται η διαδικασία εκπαίδευσης του monophone HMM model.

Ερώτημα 5: Γράψτε πώς υπολογίζεται η a posteriori πιθανότητα σύμφωνα με τον τύπο του Bayes για το πρόβλημα της αναγνώρισης φωνής. Συγκεκριμένα, πώς βρίσκεται η πιο πιθανή λέξη (ή φώνημα στην περίπτωσή μας) δεδομένης μίας ακολουθίας ακουστικών χαρακτηριστικών;

Γενικά, το πρόβλημα αναγνώρισης φωνής (ASR) αντιμετωπίζεται ως ένα στατιστικό πρόβλημα απόφασης. Συγκεκριμένα, διατυπώνεται ως μια διεργασία απόφασης με βαση τη μέγιστη a posteriori πιθανότητα, όπου αναζητούμε την ακολουθία λέξεων $\widehat{W}$ που μεγιστοποιεί την a priori πιθανότητα, P(W|X), της ακολουθίας, με δεδομένη την ακολουθία των διανυσμάτων ακουστικών χαρακτηριστικών X. [3] Δηλαδή:

$$\widehat{W} = \arg\max_{W} P(W|X)$$

Με χρήση του κανόνα του Bayes, μπορούμε να ξαναγράψουμε την παραπάνω Εξίσωση στη μορφή:

$$\widehat{W} = \arg\max_{W} \frac{P(X|W)P(W)}{P(X)}$$

Η παραπάνω εξίσωση δείχνει ότι ο υπολογισμός της a posteriori πιθανότητας αναλύεται σε δύο όρους, έναν που ορίζει την a priori πιθανότητα, ότι η ακολουθία των λέξεων, W, παρήγαγε την ακολουθία χαρακτηριστικών X, δηλαδή ο όρος $P(X|W)$ που αντιστοιχεί στο **ακουστικό μοντέλο**. Ο δεύτερος όρος P(W), που αντιστοιχεί στο **γλωσσικό μοντέλο**, εκφράζει την πιθανότητα να εμφανιστεί στα συμφραζόμενα η ακολουθία λέξεων W. Επίσης, αγνοούμε τον όρο στον παρονομαστή, P(X), διότι είναι ανεξάρτητος από την ακολουθία λέξεων W ως προς την οποία γίνεται η βελτιστοποίηση του προβλήματος. [3]

Επομένως, η τελική μορφή του προβλήματος ASR είναι:

$$\widehat{W} = \arg\max_{W} P_A(X|W)P_L(W)$$

Πιο συγκεκριμένα, η εύρεση τις πιο πιθανής λέξης (ή φωνήματος στην περίπτωση μας), δεδομένης μιας ακολουθίας ακουστικών χαρακτηριστικών X, επιτυγχάνεται με τον **αλγόριθμο Viterbi**. Ο στόχος του εν λόγω αλγορίθμου είναι να βρει τη διαδρομή μέγιστης πιθανότητας που συνδέει κάθε διάνυσμα χαρακτηριστικών X με μια μοναδική κατάσταση μοντέλου GMM - HMM, σύμφωνα με τους περιορισμούς σύνδεσης καταστάσεων του HMM. Ορίζουμε την ποσότητα δt(i), ως την πιθανότητα να βρεθεί η βέλτιστη διαδρομή στην κατάσταση I, στο χρονικό πλαίσιο t, αφού έχουν παρέλθει t διανύσματα της πρότασης και δοθέντος ενός μοντέλου HMM λ [3]. Δηλαδή:

$$\delta_t(i) = \max_{q_1,q_2,\dots,q_{t-1}} P[q_1,q_2,\dots,q_{t-1},q_t = i,\ X_1,X_2,\dots,X_t|\lambda]$$

Ο αλγόριθμος Viterbi υπολογίζει τη διαδρομή υψηλότερης πιθανότητας μέσω της ακόλουθης διαδρομής:

- Αρχικοποίηση:

$$\delta_1(i) = \pi_i b_i(X_1), 1 \le i \le N$$

$$\psi_1(i) = 0, 1 \le i \le N$$

Όπου π_i πιθανότητα αρχικής κατάστασης, b_i η πιθανότητα να εκπεμφθεί το διάνυσμα χαρακτηριστικών Xt από την κατάσταση i, σύμφωνα με την Gaussian μίξη, και ψ1(i) ένα μητρώο οπισθοδρόμησης καταστάσεων που φυλάσσει την πληροφορία για το ποιά κατάσταση προηγήθηκε της τρέχουσας.

- Αναδρομή:

$$\delta_t(j) = \max_{1\le i\le N}[\delta_{t-1}(i)\alpha_{ij}]b_j(X_t), 2 \le t \le T, 1 \le j \le N$$

$$\psi_t(j) = \max_{1\le i\le N}[\delta_{t-1}(i)\alpha_{ij}], 2 \le t \le T, 1 \le j \le N$$

- Τερματισμός:

$$P* = \max_{1\le i\le N}[\delta_T(i)]$$

$$q_T* = P* = arg\max_{1\le i\le N}[\delta_T(i)]$$

- Οπισθοδρόμηση Καταστάσεων (Διαδρομής)

$$q_t* = \psi_{t+1}(q_{t+1}*), t = T-1, T-2, \dots, 1$$

Ερώτημα 6: Εξηγήστε τη δομή του γράφου HCLG του Kaldi περιγραφικά.

Για την αποκωδικοποίηση του ακουστικού μοντέλου μας, δημιουργήσαμε με χρήση του Kaldi τον γράφο HCLG, ο οποίος προκύπτει ως διαδοχική σύνθεση FSTs HCLG = $H^{o}C^{o}L^{o}G$, όπου:

i. Ο G είναι ένας αποδοχέας (δηλαδή τα σύμβολα εισόδου και εξόδου ταυτίζονται κάθε φορά), ο οποίος κωδικοποιεί τη **γραμματική** ή το γλωσσικό μοντέλο.
ii. Ο L είναι το **λεξικό**, δηλαδή τα σύμβολα εισόδου είναι φωνήματα και τα σύμβολα εξόδου λέξεις.
iii. Ο C αναπαριστά την εξάρτηση από τα **συμφραζόμενα**: τα σύμβολα εξόδου είναι φωνήματα και τα σύμβολα εισόδου αναπαριστούν context-dependent φωνήματα (π.χ. παράθυρα N φωνημάτων)
iv. Ο H ορίζει το Κρυφό Μαρκοβιανό Μοντέλο HMM, καθώς έχει ως σύμβολα εξόδου context-dependent φωνήματα και ως σύμβολα εισόδου τα αναγνωριστικά των μεταβάσεων μεταξύ καταστάσεων (transition IDs), τα οποία κωδικοποιούν τις κατανομές πιθανότητες των κρυφών καταστάσεων.

Είναι επιθυμητό η έξοδος του FST να είναι **determinized** και **minimized**. Προκειμένου ο γράφος HCLG να είναι determinized & minimized χρειάζεται να εισάγουμε τα **disambiguation symbols** <s> και </s> στην αρχή και στο τέλος κάθε πρότασης αντίστοιχα. Επιπλέον, είναι επιθυμητό ο τελικός γράφος να είναι **stochastic**, κάτι το οποίο μπορεί να επιτευχθεί με την τεχνική του weight pushing. [4]

Αν θέλουμε να συνοψίσουμε τη δημιουργία του τελικού γράφου, τότε ορίζοντας ως asl = 'add self-loops' και rds = 'remove disambiguation-symbols', καθώς και ως H' τον μετατροπέα H χωρίς self-loops, η διαδικασία μπορεί να περιγραφεί ως:

$$HCLG = asl(min(rds(det(H' \circ min(det(C \circ min(det(L \circ G))))))))$$

Βελτιώσεις Συστήματος ASR

Συμπερασματικά, οι τυπικοί αλγόριθμοι που αναπτύξαμε επαρκούν για την ανάπτυξη ενός ASR συστήματος με ικανοποιητικές επιδόσεις. Στην πράξη, βέβαια, τα σύγχρονα συστήματα ASR ακολουθούν μια **attention-based encoder-decoder αρχιτεκτονική**, η οποία υλοποιείται συνήθως με RNNs ή Transformers. Έχοντας εξάγει τα log mel συχνοτικά χαρακτηριστικά, όπως δείξαμε, θέτουμε ως είσοδο μια ακολουθία από t διανύσματα ακουστικών χαρακτηριστικών F = f1,f2,…,ft, ένα διάνυσμα για κάθε 10msec πλαίσια. Η έξοδος του συστήματος μπορεί να είναι γράμματα ή τμήματα λέξεων. [1]

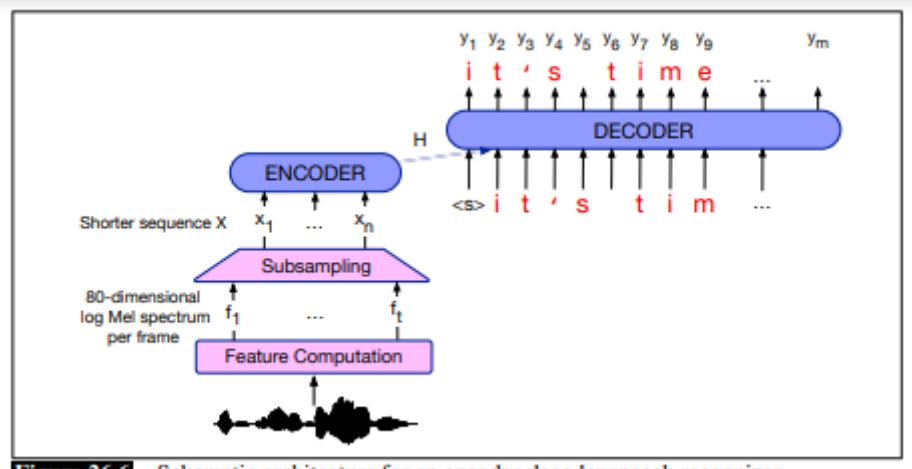


Figure 26.6 Schematic architecture for an encoder-decoder speech recognizer.

Η αρχιτεκτονική encoder-decoder είναι κατάλληλη όταν η είσοδος και η έξοδος διαφέρουν σημαντικά ως προς το μέγεθος , κάτι το οποίο συμβαίνει συχνά στην ομιλία καθώς ένα μια ακουστική ακολουθία μεγάλης διάρκειας συχνά αντιστοιχεί σε μια αρκετά συντομότερη ακολουθία από γράμματα ή λέξεις.

Προκειμένου να βελτιωθεί η επίδοση του συστήματος, μπορούμε να το εφοδιάσουμε με ένα επιπλέον **αρκετά μεγαλύτερο γλωσσικό μοντέλο**, καθώς είναι πολύ πιο εύκολο να συλλέξουμε για την εκπαίδευση του συστήματος πληροφορία υπό μορφή κειμένου, παρά γραπτή πληροφορία συνοδευόμενη από τον αντίστοιχο ήχο. [1]

Τέλος, ενναλακτική αρχιτεκτονική που μπορεί να βελτιώνει τις επιδόσεις του συστήματος ASR είναι hybrid Κρυφά Μαρκοβιανά Μοντέλα υποβοηθούμενα από Βαθιά Νευρωνικά Δίκτυα (**DNN-HMM**), τα οποία υπερτερούν έναντι των GMM-HMM στην αναγνώριση ομιλίας.

References

[1] [J&M] D. Jurafski, J. H. Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics." Prentice-Hall.

[2] Introduction to HMM-GMM acoustic modeling – Mael Fabien https://maelfabien.github.io/machinelearning/speech_reco_1/#

[3] Ψηφιακή Επεξεργασία Φωνής: Θεωρία και Εφαρμογές, Lawrence R. Rabiner, Ronald W. Schafer, 1η Έκδοση, Εκδόσεις Π.Χ.Πασχαλίδης, 2011

[4] Decoding graph construction in Kaldi – http://kaldi-asr.org/doc/graph.html#graph_disambig