Επεξεργασία Φωνής και Φυσικής Γλώσσας

2ο εργαστήριο: Αναγνώριση Φωνής με το KALDI TOOLKIT

<u>Στοιχεία Φοιτητή</u>

Ονοματεπώνυμο: Αντωνιάδης Παναγιώτης

Σχολή: ΗΜΜΥ

Εξάμηνο: 7ο

ΑΜ: 03115009

Ροή: Σ

1. Περιγραφή

Σκοπός της άσκησης αυτής είναι η υλοποίηση ενός συστήματος επεξεργασίας και αναγνώρισης φωνής με το εργαλείο Kaldi, το οποίο χρησιμοποιείται ευρέως στον ερευνητικό τομέα, αλλά και όχι μόνο, για την εκπαίδευση state-of-the-art συστημάτων αναγνώρισης φωνής. Πιο συγκεκριμένα, το σύστημα που θα αναπτύξουμε αφορά σε αναγνώριση φωνημάτων (Phone Recognition) από ηχογραφήσεις της USC-TIMIT. Μας δίνονται δεδόμενα audio από 4 διαφορετικούς ομιλητές, με τα αντίστοιχα transcriptions, ώστε να εκπαιδεύσουμε και να εκτιμήσετε το σύστημά μας. 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 αντιλαμβάνεται 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

MFCCs

Στην υλοποίηση ενός συστήματος αναγνώρισης φωνής για την εξαγωγή ακουστικών χαρακτηριστικών από τα δεδομένα μας συνήθως χρησιμοποιούμε τα Mel Frequency Spectral Coefficients ή MFCCs. Δεδομένου ενός σήματος φωνής η διαδικασία υπολογισμού αυτών των χαρακτηριστικών είναι η εξής:

- Αρχικά κάνουμε **προέμφαση** του σήματος φωνής που έχουμε για να ενισχύσουμε τις υψηλές συχνότητες. Με αυτό πετυχαίνουμε μία εξισορρόπηση του φάσματος, αποφεύγουμε υπολογιστικά λάθη στον μετασχηματισμό Fourier που θα κάνουμε αργότερα και συνήθως βελτιώνεται και ο σηματοθορυβικός λόγος (SNR).
- **Παραθυροποίηση** του σήματος σε πλαίσια με μήκος συνήθως 20-40 ms και επικάλυψη 50%. Αυτό είναι απαραίτητο διότι ένα σήμα φωνής δεν είναι στατικό στη διάρκεια του χρόνου και ο μετασχηματισμός Fourier πάνω σε αυτό δεν θα μας έδινε καμία χρήσιμη πληροφορία. Μπορούμε όμως να θεωρήσουμε ότι σε ένα πλαίσιο τα χαρακτηριστικά του σήματος δεν αλλάζουν και να εφαρμόσουμε τον μετασχηματισμό εκεί. Αφού χωρίσουμε, λοιπόν, το σήμα εφαρμόζουμε σε κάθε πλαίσιο συνήθως το Hamming παράθυρο το οποίο έχει το πιο βολικό για εμάς φάσμα σε σχέση με άλλα παράθυρα (πλαταίνει το φάσμα αλλά περιορίζονται οι πλευρικοί λοβοί).
- Στη συνέχεια εφαρμόζουμε το **μετασχηματισμό DFT** σε κάθε πλαίσιο το οποίο ονομάζεται και STFT και υπολογίζουμε το power spectrum.
- Το επόμενο βήμα είναι να εφαρμόσουμε ένα **filterbank** στο power spectrum του σήματος η κατασκευή του οποίου βασίζεται στον τρόπο λειτουργίας του ανθρώπινου αυτιού το οποίο

διαχωρίζει πιο εύκολα τις χαμηλές από τις υψηλές συχνότητες. Το filterbank αποτελείται συνήθως από 40 τριγωνικά φίλτρα όπου κάθε φίλτρο σε κλίμακα Mel έχει απόκριση 1 στην κεντρική του συχνότητα και μειώνεται γραμμικά μέχρι να μηδενιστεί στις κεντρικές συχνότητες των δύο γειτονικών του φίλτρων.

- Τέλος, εφαρμόζουμε τον μετασχηματισμό DCT για να μειώσουμε την υψηλή συσχέτιση των τελικών μας δεδομένων. Τα εξαγόμενα MFCCs είναι οι πρώτες 2 με 13 cepstral coefficients.

Ένας άλλος τρόπος εξαγωγής ακουστικών χαρακτηριστικών είναι να σταματήσουμε στον υπολογισμό της εξόδου του filter bank χωρίς να εφαρμόσουμε DCT μετασχηματισμό στο σήμα μας. Όπως αναφέρθηκε, το τελευταίο ήταν απαραίτητο για να μειωθεί η υψηλή συσχέτιση του σήματος μας η οποία είναι και ανεπιθύμητη όταν θα εφαρμόσουμε στη συνέχεια κάποιον αλγόριθμο μηχανικής μάθησης. Τώρα πια, ωστόσο, με την ανάπτυξη του Deep Learning και των νευρωνικών δικτύων η μείωση της συσχέτισης των τελικών δειγμάτων δεν είναι απαραίτητη διότι τα νευρωνικά δίκτυα είναι λιγότερο επιρρεπή σε αυτή. Έτσι, εφαρμόζοντας αλγορίθμους Deep Learning μπορούμε να αποφύγουμε το βήμα αυτό και έτσι, παράλληλα, να κρατήσουμε την πληροφορία που χανόταν λόγω του DCT ο οποίος είναι ένας γραμμικός μετασχηματισμός.

Τέλος, ένα ακόμη τρόπος βελτίωσης είναι να αποφύγουμε να εφαρμόσουμε μετασχηματισμό Fourier τελείως γιατί και αυτός σας ένας γραμμικός μετασχηματισμός αποκρύπτει χρήσιμη πληροφορία και να “μάθουμε” κατευθείαν από το σήμα στο χρόνο.

Γλωσσικό Μοντέλο (Language Model)

Το γλωσσικό μοντέλο ουσιαστικά περιέχει την a priori πιθανότητα να έρθει μία συγκεκριμένη λέξη ή (στην περίπτωσή μας) ένα φώνημα. Το δημιουργούμε με βάση ένα transcription των προτάσεων που έχουμε και αποτελείται από n-grams όπου n είναι ο αριθμός των προηγούμενων φωνημάτων από τις οποίες εξαρτάται η εμφάνιση ενός φωνήματος. Συνήθως χρησιμοποιούμε unigrams και bigrams μοντέλα τα οποία το KALDI τα συνδυάζει αυξάνοντας έτσι την ακρίβεια του μοντέλου (αλλά και την υπολογιστική πολυπλοκότητα).

Φωνητικό Μοντέλο (Acoustic Model)

Το φωνητικό μοντέλο περιέχει την σχέση μεταξύ των φωνητικών χαρακτηριστικών που εξάγουμε από το σήμα και των φωνημάτων που υπάρχουν στο αντίστοιχο transcription. Δηλώνει ουσιαστικά την πιθανότητα P(o|w) δηλαδή την πιθανότητα να έχουμε ορισμένα ακουστικά χαρακτηριστικά δεδομένου ότι έχουν προκύψει από κάποια συγκεκριμένα φωνήματα και αναπαριστάται με κρυφά μαρκοβιανά μοντέλα (HMM).

3. Βήματα προπαρασκευής

Αρχικά δημιουργούμε μέσα στον φάκελο **kaldi/egs** ένα φάκελο **usc** μέσα στο οποίο θα έχουμε ότι κάνουμε από εδώ και πέρα. Την υπόλοιπη διαδικασία της προπαρασκευής την αναλαμβάνει το αρχείο **prolab.sh** το οποίο τρέχοντας το με **./prolab.sh** δημιουργούνται όλα τα απαραίτητα αρχεία. Συγκεκριμένα, δημιουργούμε τον φάκελο **data** και τους υποφακέλους **data/train**, **data/dev**, **data/test**, μέσα στους οποίους θα δημιουργήσουμε αρχεία-δείκτες τα οποία θα περιγράφουν τα δεδομένα εκπαίδευσης, επαλήθευσης και αποτίμησης αντίστοιχα. Για κάθε έναν από τους 3 φακέλους πρέπει να δημιουργήσουμε τα παρακάτω αρχεία:

- **uttids:** Περιέχει στην κάθε του γραμμή ένα μοναδικό συμβολικό όνομα για κάθε πρόταση του συγκεκριμένου συνόλου δεδομένων. Απλά, αντιγράφουμε το αντίστοιχο αρχείο του φακέλου filesets του φακέλου που κατεβάσαμε.
- **utt2spk:** Περιέχει σε κάθε γραμμή τον ομιλητή που αντιστοιχεί σε κάθε πρόταση. Αυτό το επιτυγχάνουμε μέσω ενός python script όπου διαβάζει τα αντίστοιχα αρχεία uttids και σε κάθε γραμμή γράφει το χαρακτηριστικό του ομιλητή χωρισμένο με ένα κενό.

- **wav.scp:** Περιέχει τη θέση του αρχείου ήχου που αντιστοιχεί σε κάθε πρόταση. Αυτό επίσης το πετυχαίνουμε με ένα python script όπου σε κάθε γραμμή αφού γράψει το αντίστοιχο uttid γράφει το αντίστοιχο path του αρχείου ήχου.
- **text:** Περιέχει το κείμενο που αντιστοιχεί στην κάθε πρόταση.

Τέλος, πρέπει να αντικαταστήσουμε στο τελευταίο αρχείο text τις προτάσεις με φωνήματα διότι στη συνέχεια της άσκησης θα ασχοληθούμε με αυτά. Για να συμβεί αυτό μας έχει δοθεί ένα φωνητικό λεξικό με όνομα **lexicon.txt** το οποίο περιέχει για κάθε λέξη την αντίστοιχη ακολουθία φωνημάτων. Αρχικά, λοιπόν, δημιουργούμε ένα λεξικό στην python όπου για κάθε λέξη του φωνητικού λεξικού (κλειδί) έχει ως τιμή την ακολουθία φωνημάτων σε μία λίστα. Στη συνέχεια για κάθε έναν από τους τρεις φακέλους δημιουργούμε ένα προσωρινό αρχείο **text_temp** στο οποίο θα αποθηκεύσουμε το αρχείο που ζητείται. Για κάθε γραμμή του αρχείου text γράφουμε στο text_temp το utterance id της προτάσεις μετά προσθέτουμε το φώνημα της σιωπής **sil** και μετά για κάθε λέξη της πρότασης γράφουμε τα φωνήματα της χρησιμοποιώντας το λεξικό που δημιουργήσαμε και στο τέλος της πρότασης ξαναπροσθέτουμε το φώνημα της σιωπής **sil**. Στο τέλος, αντιγράφουμε το περιεχόμενο του text_temp στο αρχείο text το οποίο είναι και το τελικό μας αρχείο.

4. Βήματα κυρίως μέρους

▪ Προετοιμασία διαδικασίας αναγνώρισης φωνής για την USC-TIMIT

Αρχικά αντιγράφουμε τα αρχεία path.sh και cmd.sh στον φάκελο usc όπου έχουμε όλα τα αρχεία μας και θέτουμε την μεταβλητή KALDI_ROOT του path.sh στο directory που βρίσκεται ο φάκελος εγκατάστασης του Kaldi. Επίσης, στο cmd.sh αλλάζουμε τις τιμές των μεταβλητών train_cmd, decode_cmd και cuda_cmd σε run.pl.

Τα επόμενα βήματα τα αναλαμβάνει το αρχείο lab4_1.sh το οποίο αρχικά δημιουργεί τα ζητούμενα soft links και στην συνέχεια όλους τους απαραίτητους φακέλους που απαιτούνται για την συνέχεια.

▪ Προετοιμασία γλωσσικού μοντέλου

Όλη αυτή την διαδικασία την αναλαμβάνει το αρχείο lab4_2.sh το οποίο απλά το τρέχουμε με run ./lab4_2.sh.

Αρχικά, δημιουργούμε ορισμένα αρχεία που είναι απαραίτητα για την δημιουργία του γλωσσικού μοντέλου και τα αποθηκεύουμε στον φάκελο data/local/dict. Συγκεκριμένα, τα **silence_phones.txt** και **optional-silence.txt** τα οποία περιέχουν μόνο το φωνής της σιωπής **sil**. Ακόμη, θέλουμε ένα αρχείο **nonsilence_phones.txt** το οποίο θα περιέχει όλα τα υπόλοιπα φωνήματα. Αυτό το δημιουργούμε με ένα python script το οποίο αποθηκεύει σε μία λίστα όλα τα διαφορετικά φωνήματα από το lexicon.txt και τα γράφει ταξινομημένα το ένα κάτω από το άλλο στο αρχείο nonsilence_phones.txt. Στη συνέχεια, δημιουργούμε ένα αρχείο **lexicon.txt** το οποίο θα είναι το φωνητικό μας λεξικό και θα περιέχει μία αντιστοιχία ένα προς ένα κάθε φωνήματος με τον εαυτό του (συμπεριλαμβανομένου του φωνήματος της σιωπής). Το αρχείο αυτό δημιουργείται αντιγράφοντας δύο φορές το περιεχόμενο του nonsilence_phones.txt χωρισμένο με κενό και προσθέτοντας μία αντιστοιχία του φωνήματος της σιωπής με τον εαυτό του. Μετά, ταξινομούμε το αρχείο αλφαβητικά. Επίσης, δημιουργούμε για κάθε αρχείο text της προπαρασκευής ένα αρχείο lm_train.text (και lm_dev.text, lm_test.text αντίστοιχα) τα οποία περιέχει σε κάθε γραμμή ένα <s> πριν τα φωνήματα και ένα </s> στο τέλος της πρότασης. Τέλος, δημιουργούμε και ένα κενό αρχείο extra_questions.txt.

Στη συνέχεια, δημιουργούμε μία ενδιάμεση μορφή του γλωσσικού μοντέλου και την αποθηκεύουμε στον φάκελο data/local/lm_temp χρησιμοποιώντας την εντολή build-lm.sh του πακέτου IRSTLM που έχει εγκατασταθεί μαζί με το KALDI και συντάσσεται ως εξής:

build-lm.sh -i <αρχείο lm_train.text> -n <τάξη γλωσσικού μοντέλου> -o <αρχείο_εξόδου.ilm.gz>

Το επόμενο βήμα είναι να αποθηκευτεί το compiled γλωσσικό μοντέλο σε μορφή ARPA μέσω της εντολής compile-lm που συντάσσεται ως εξής:

compile-lm <αρχείο .ilm.gz> -t=yes /dev/stdout | grep -v unk | gzip -c > <αρχείο_εξόδου.arpa.gz>

Στη συνέχεια, δημιουργούμε το FST του λεξικού μας χρησιμοποιώντας τα παραπάνω αρχεία και στην εντολή prepate_lang.sh του KALDI.

Τέλος, δημιουργούμε το FST της γραμματικής μας χρησιμοποιώντας το timit_format_data.sh με κάποιες αλλαγές.

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

Για να υπολογίσουμε το perplexity στο validation και στο test set χρησιμοποιούμε την εντολή compile-lm προσθέτοντας και το –eval όρισμα. Τρέχοντας, λοιπόν, το perplexity.sh έχουμε τα παρακάτω αποτελέσματα:

- **unigram στο developement set**

```
# Calculating perplexity for the unigram model of the dev set #
inpfile: lm_dev_unigram.ilm.gz
outfile: lm_dev_unigram.ilm.blm
evalfile: ../dict/lm_dev.text
loading up to the LM level 1000 (if any)
dub: 10000000
OOV code is 42
OOV code is 42
Start Eval
OOV code: 42
%% Nw=6357 PP=32.51 PPwp=0.00 Nbo=0 Noov=0 OOV=0.00%
```

- **bigram το developement set**

```
# Calculating perplexity for the bigram model of the dev set #
inpfile: lm_dev_bigram.ilm.gz
outfile: lm_dev_bigram.ilm.blm
evalfile: ../dict/lm_dev.text
loading up to the LM level 1000 (if any)
dub: 10000000
OOV code is 42
OOV code is 42
Start Eval
OOV code: 42
%% Nw=6357 PP=15.26 PPwp=0.00 Nbo=0 Noov=0 OOV=0.00%
```

- **unigram στο test set**

```
# Calculating perplexity for the unigram model of the test set #
inpfile: lm_test_unigram.ilm.gz
outfile: lm_test_unigram.ilm.blm
evalfile: ../dict/lm_test.text
loading up to the LM level 1000 (if any)
dub: 10000000
OOV code is 42
OOV code is 42
Start Eval
OOV code: 42
%% Nw=6179 PP=32.00 PPwp=0.00 Nbo=0 Noov=0 OOV=0.00%
```

- **bigram στο test set**

```
# Calculating perplexity for the bigram model of the test set #
inpfile: lm_test_bigram.ilm.gz
outfile: lm_test_bigram.ilm.blm
evalfile: ../dict/lm_test.text
loading up to the LM level 1000 (if any)
dub: 10000000
OOV code is 42
OOV code is 42
Start Eval
OOV code: 42
%% Nw=6179 PP=14.65 PPwp=0.00 Nbo=0 Noov=0 OOV=0.00%
```

Παρατηρούμε ότι το perplexity μειώνεται όταν πάμε από unigram σε bigram μοντέλο πράγμα αναμενόμενο αφού η κατανομή πιθανοτήτων στο bigram μοντέλο είναι πιο αποτελεσματική.

▪ Εξαγωγή ακουστικών χαρακτηριστικών

Για να εξάγουμε τα **MFCCs** θα χρησιμοποιήσουμε τις συναρτήσεις make_mfcc.sh και compute_cmvn_stats.sh που μας παρέχει το KALDI. Αρχικά, βέβαια πρέπει να δημιουργήσουμε τα αρχεία spk2utt που θα χρειαστούν οι παραπάνω συναρτήσεις. Αυτό γίνεται αμέσως με την εντολή utt2spk_to_spk2utt.pl χρησιμοποιώντας στο αρχείο utt2spk (για κάθε ένα από τους 2 φακέλους). Όλη αυτή η διαδικασία γίνεται στο αρχείο lab4_3.sh.

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

Όταν θέλουμε να υλοποιήσουμε ένα σύστημα ASR σκοπός μας είναι να αφαιρέσουμε οτιδήποτε επηρεάζει το αρχικό σήμα του ομιλητή και το μεταβάλλει. Αν για παράδειγμα θεωρήσουμε ως x[n] το αρχικό σήμα του ομιλητή και h[n] το σύνολο των σημάτων που το επηρέασαν μέχρι να ηχογραφηθεί (θόρυβος κλπ) τότε το ηχογραφημένο σήμα είναι η συνέλιξή τους στο χρόνο x[n]⋆h[n] και το γινόμενο των φασμάτων τους στη συχνότητα X(f)H(f). Παίρνοντας, λοιπόν, το cepstrum του τελικού σήματος το γινόμενο γίνεται άθροισμα και έχουμε Y(f) = Xc(f) + Hc(f). Θεωρώντας ότι το σήμα h είναι στατικό τότε για κάθε πλαίσιο ισχύει: Yi(f) = Xi(f) + Hc(f). Παίρνοντας τον μέσο όλων των πλαισίων έχουμε:

$$(1/N)\sum_i(Yi[f])=H[f]+(1/N)\sum_i(Xi[f])$$

και ορίζοντας την διαφορά

$$Ri[f]=Yi[f]-(1/N)\sum_j(Y_j[f]) \;=$$

$$H[f]+Xi[f]-(H[f]+(1/N)\sum_j(X_j[f]))=Xi[f]-(1/N)\sum jXj[f]$$

Παρατηρούμε, λοιπόν, ότι έχουμε ένα σήμα χωρίς τις αλλοιώσεις από εξωτερικούς παράγοντες και αυτό είναι ο σκοπός που εξυπηρετεί το Ceptral Mean and Variance Normalization.

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

Θα χρησιμοποιήσουμε τις εντολές feat-to-dim και feat-to-len που παρέχει το KALDI. Συγκεκριμένα, το feat-to-dim επιστρέφει την διάσταση των χαρακτηριστικών που θα είναι ίδια για όλες τις προτάσεις και το feat-to-len επιστρέφει πόσα ακουστικά frames εξήχθησαν για κάθε πρόταση από το set που θα πάρει ως όρισμα. Τρέχοντας λοιπόν το lab4_3_question2.sh έχουμε το παρακάτω output:

```
feat-to-dim ark:mfcc_train/raw_mfcc_train.1.ark -
13
feat-to-len scp:data/train/feats.scp ark,t:data/train/feats.lengths
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
```

Έχουμε δηλαδή για τις 5 πρώτες προτάσεις 237, 377, 317, 399 και 338 frames αντίστοιχα και διάσταση 13 για τα χαρακτηριστικά.

▪ Εκπαίδευση ακουστικών μοντέλων και αποκωδικοποίηση προτάσεων

- Αρχικά θα εκπαιδεύσουμε ένα monophone GMM-HMM ακουστικό μοντέλο πάνω στα train δεδομένα και στη συνέχεια θα κάνουμε alignment των φωνημάτων. Αυτό γίνεται με τις εντολή train_mono.sh και align_si.sh και τρέχοντας το lab4_4_train.sh που τις περιέχει έχουμε την παρακάτω έξοδο:

```
panos@panos-Inspiron-3543:~/Documents/ece/7th-semester/flow-S/nlp/2hSeira/kaldi/egs/usc$ ./lab4_4_train.sh
Training the monophone model
steps/train_mono.sh data/train data/lang_test exp/mono0
steps/train_mono.sh: Initializing monophone system.
steps/train_mono.sh: Compiling training graphs
steps/train_mono.sh: Aligning data equally (pass 0)
steps/train_mono.sh: Pass 1
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 2
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 3
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 4
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 5
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 6
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 7
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 8
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 9
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 10
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 11
steps/train_mono.sh: Pass 12
```

```
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 13
steps/train_mono.sh: Pass 14
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 15
steps/train_mono.sh: Pass 16
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 17
steps/train_mono.sh: Pass 18
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 19
steps/train_mono.sh: Pass 20
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 21
steps/train_mono.sh: Pass 22
steps/train_mono.sh: Pass 23
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 24
steps/train_mono.sh: Pass 25
steps/train_mono.sh: Pass 26
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 27
steps/train_mono.sh: Pass 28
steps/train_mono.sh: Pass 29
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 30
steps/train_mono.sh: Pass 31
steps/train_mono.sh: Pass 32
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 33
steps/train_mono.sh: Pass 34
steps/train_mono.sh: Pass 35
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 36
steps/train_mono.sh: Pass 37
steps/train_mono.sh: Pass 38
steps/train_mono.sh: Aligning data
steps/train_mono.sh: Pass 39
steps/diagnostic/analyze_alignments.sh --cmd run.pl data/lang_test exp/mono0
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only 44.6416382253% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only 49.4197952218% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_alignments.sh: see stats in exp/mono0/log/analyze_alignments.log
3858 warnings in exp/mono0/log/align.*.*.log
2 warnings in exp/mono0/log/analyze_alignments.log
502 warnings in exp/mono0/log/acc.*.*.log
121 warnings in exp/mono0/log/update.*.log
exp/mono0: nj=4 align prob=-84.30 over 1.81h [retry=1.6%, fail=0.2%] states=125 gauss=999
steps/train_mono.sh: Done training monophone system in exp/mono0
Align the monophone model
steps/align_si.sh data/train data/lang_test exp/mono0 exp/mono0_ali
steps/align_si.sh: feature type is delta
steps/align_si.sh: aligning data in data/train using model from exp/mono0, putting alignments in exp/mono0_ali
steps/diagnostic/analyze_alignments.sh --cmd run.pl data/lang_test exp/mono0_ali
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only 45.7337883959% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only 48.5324232082% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_alignments.sh: see stats in exp/mono0_ali/log/analyze_alignments.log
steps/align_si.sh: done aligning data.
```

- Στη συνέχεια θα δημιουργήσουμε το γράφο HCLG του Kaldi σύμφωνα με τη γραμματική (G) του προηγούμενου βήματος. Αυτό γίνεται με την εντολή mkgraph.sh του KALDI. Θα δημιουργήσουμε τον γράφο 2 φορές μία για το unigram μοντέλο και μία για το bigram.
- Μετά, θα αποκωδικοποιήσουμε τις προτάσεις των validation και των test set δεδομένων με τον αλγόριθμο του Viterbi. Αυτό γίνεται με την εντολή decode.sh που παρέχεται από το KALDI.
- Τέλος, υπολογίζουμε το PER το οποίο στο KALDI αναφέρεται ως WER διότι συνήθως κάνουμε αναγνώριση λέξεων και όχι φωνημάτων το οποιο το αναλαμβάνει η εντολή best_wer.sh

Και τα τρία παραπάνω βήματα γίνονται τρέχοντας τα αρχεία lab4_4_unigram.sh για το unigram μοντέλο και lab4_4_bigram για το bigram μοντέλο. Τα αποτελέσματα φαίνονται παρακάτω (πρώτα το unigram και μετά το bigram). Χρειάστηκε να μετονομάσω το score_kaldi.sh σε score.sh γιατί έτσι μου έδειχνε ότι το θέλει τι script.

unigram model

```
WARNING: the --mono, --left-biphone and --quinphone options are now deprecated and ignored.
tree-info exp/mono0/tree
tree-info exp/mono0/tree
fsttablecompose data/lang_test/L_disambig.fst data/lang_test/G.fst
fstpushspecial
fstdeterminizestar --use-log=true
fstminimizeencoded
fstisstochastic data/lang_test/tmp/LG.fst
0.000519182 0.000486742
fstcomposecontext --context-size=1 --central-position=0 --read-disambig-syms=data/lang_test/phones/disambig.int --write-disambig-syms=data/lang_test/t
mp/disambig_ilabels_1_0.int data/lang_test/tmp/ilabels_1_0.20714 data/lang_test/tmp/LG.fst
fstisstochastic data/lang_test/tmp/CLG_1_0.fst
0.000519182 0.000486742
make-h-transducer --disambig-syms-out=exp/mono0/graph_nosp_tgpr/disambig_tid.int --transition-scale=1.0 data/lang_test/tmp/ilabels_1_0 exp/mono0/tree
exp/mono0/final.mdl
fstrmsymbols exp/mono0/graph_nosp_tgpr/disambig_tid.int
fsttablecompose exp/mono0/graph_nosp_tgpr/Ha.fst data/lang_test/tmp/CLG_1_0.fst
fstrmepslocal
fstdeterminizestar --use-log=true
fstminimizeencoded
fstisstochastic exp/mono0/graph_nosp_tgpr/HCLGa.fst
0.000976562 -0.000362418
add-self-loops --self-loop-scale=0.1 --reorder=true exp/mono0/final.mdl exp/mono0/graph_nosp_tgpr/HCLGa.fst
steps/decode.sh exp/mono0/graph_nosp_tgpr data/dev exp/mono0/decode_dev
decode.sh: feature type is delta
steps/diagnostic/analyze_lats.sh --cmd run.pl exp/mono0/graph_nosp_tgpr exp/mono0/decode_dev
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0/decode_dev/log/analyze_alignments.log
Overall, lattice depth (10,50,90-percentile)=(10,42,598) and mean=302.1
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0/decode_dev/log/analyze_lattice_depth_stats.log
local/score.sh --cmd run.pl data/dev exp/mono0/graph_nosp_tgpr exp/mono0/decode_dev
local/score.sh: scoring with word insertion penalty=0.0,0.5,1.0
%WER 51.39 [ 3173 / 6174, 88 ins, 1685 del, 1400 sub ] exp/mono0/decode_dev/wer_7_0.0
steps/decode.sh exp/mono0/graph_nosp_tgpr data/test exp/mono0/decode_test
decode.sh: feature type is delta
```

```
steps/diagnostic/analyze_lats.sh --cmd run.pl exp/mono0/graph_nosp_tgpr exp/mono0/decode_test
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0/decode_test/log/analyze_alignments.log
Overall, lattice depth (10,50,90-percentile)=(8,44,494) and mean=240.6
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0/decode_test/log/analyze_lattice_depth_stats.log
local/score.sh --cmd run.pl data/test exp/mono0/graph_nosp_tgpr exp/mono0/decode_test
local/score.sh: scoring with word insertion penalty=0.0,0.5,1.0
%WER 49.04 [ 2940 / 5995, 68 ins, 1587 del, 1285 sub ] exp/mono0/decode_test/wer_7_0.0
```

bigram model

```
panos@panos-Inspiron-3543:~/Documents/ece/7th-semester/flow-S/nlp/2hSeira/kaldi/egs/usc$ ./lab4_4_bigram.sh
WARNING: the --mono, --left-biphone and --quinphone options are now deprecated and ignored.
tree-info exp/mono0/tree
tree-info exp/mono0/tree
fstpushspecial
fstminimizeencoded
fsttablecompose data/lang_test/L_disambig.fst data/lang_test/G.fst
fstdeterminizestar --use-log=true
fstisstochastic data/lang_test/tmp/LG.fst
-0.00855164 -0.009256
fstcomposecontext --context-size=1 --central-position=0 --read-disambig-syms=data/lang_test/phones/disambig.int --write-disambig-syms=data/lang_test/t
mp/disambig_ilabels_1_0.int data/lang_test/tmp/ilabels_1_0.24701 data/lang_test/tmp/LG.fst
fstisstochastic data/lang_test/tmp/CLG_1_0.fst
-0.00855164 -0.00925601
make-h-transducer --disambig-syms-out=exp/mono0/graph_nosp_tgpr/disambig_tid.int --transition-scale=1.0 data/lang_test/tmp/ilabels_1_0 exp/mono0/tree
exp/mono0/final.mdl
fsttablecompose exp/mono0/graph_nosp_tgpr/Ha.fst data/lang_test/tmp/CLG_1_0.fst
fstdeterminizestar --use-log=true
fstrmepslocal
fstrmsymbols exp/mono0/graph_nosp_tgpr/disambig_tid.int
fstminimizeencoded
fstisstochastic exp/mono0/graph_nosp_tgpr/HCLGa.fst
0.000433958 -0.010419
HCLGa is not stochastic
add-self-loops --self-loop-scale=0.1 --reorder=true exp/mono0/final.mdl exp/mono0/graph_nosp_tgpr/HCLGa.fst
steps/decode.sh exp/mono0/graph_nosp_tgpr data/dev exp/mono0/decode_dev
decode.sh: feature type is delta
steps/diagnostic/analyze_lats.sh --cmd run.pl exp/mono0/graph_nosp_tgpr exp/mono0/decode_dev
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only 44.262295082% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only 37.1584699454% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0/decode_dev/log/analyze_alignments.log
Overall, lattice depth (10,50,90-percentile)=(3,17,121) and mean=69.9
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0/decode_dev/log/analyze_lattice_depth_stats.log
local/score.sh --cmd run.pl data/dev exp/mono0/graph_nosp_tgpr exp/mono0/decode_dev
local/score.sh: scoring with word insertion penalty=0.0,0.5,1.0
%WER 46.27 [ 2857 / 6174, 115 ins, 1226 del, 1516 sub ] exp/mono0/decode_dev/wer_7_0.0
steps/decode.sh exp/mono0/graph_nosp_tgpr data/test exp/mono0/decode_test
decode.sh: feature type is delta
```

```
steps/diagnostic/analyze_lats.sh --cmd run.pl exp/mono0/graph_nosp_tgpr exp/mono0/decode_test
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only 46.7391304348% of the time at utterance begin.  This may not be optimal.
analyze_phone_length_stats.py: WARNING: optional-silence sil is seen only 41.847826087% of the time at utterance end.  This may not be optimal.
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0/decode_test/log/analyze_alignments.log
Overall, lattice depth (10,50,90-percentile)=(3,17,115) and mean=50.6
steps/diagnostic/analyze_lats.sh: see stats in exp/mono0/decode_test/log/analyze_lattice_depth_stats.log
local/score.sh --cmd run.pl data/test exp/mono0/graph_nosp_tgpr exp/mono0/decode_test
local/score.sh: scoring with word insertion penalty=0.0,0.5,1.0
%WER 43.57 [ 2612 / 5995, 106 ins, 1144 del, 1362 sub ] exp/mono0/decode_test/wer_7_0.0
```

Έχουμε, λοιπόν, τα παρακάτω αποτελέσματα:

- dev set με unigram μοντέλο

```
%WER 51.39 [ 3173 / 6174, 88 ins, 1685 del, 1400 sub ] exp/mono0/decode_dev/wer_7_0.0
```

- test set με unigram μοντέλο

```
%WER 49.04 [ 2940 / 5995, 68 ins, 1587 del, 1285 sub ] exp/mono0/decode_test/wer_7_0.0
```

- dev set με bigram μοντέλο

```
%WER 46.27 [ 2857 / 6174, 115 ins, 1226 del, 1516 sub ] exp/mono0/decode_dev/wer_7_0.0
```

- test set με bigram μοντέλο

```
%WER 43.57 [ 2612 / 5995, 106 ins, 1144 del, 1362 sub ] exp/mono0/decode_test/wer_7_0.0
```

Παρατηρούμε, λοιπόν, ότι όπως ήταν αναμενόμενο βάζοντας το bigram μοντέλο το PER μειώθηκε αισθητά.

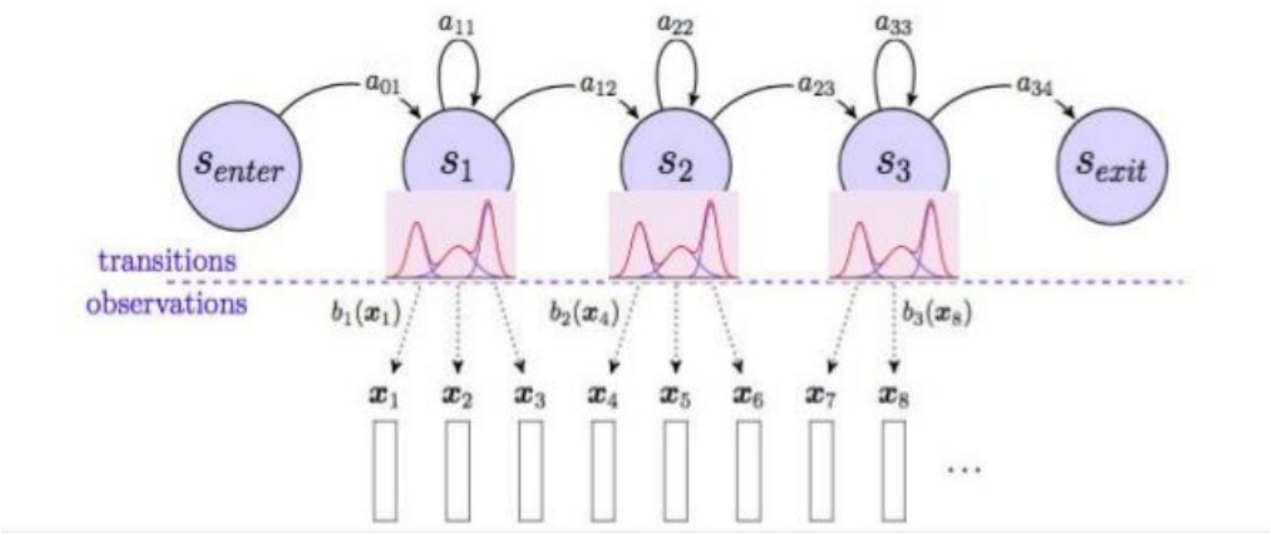
Ερώτημα 4: Εξηγήστε τη δομή ενός ακουστικού μοντέλου GMM-HMM. Τι σκοπό εξυπηρετούν τα μαρκοβιανά μοντέλα στη συγκεκριμένη περίπτωση και τι τα μίγματα γκαουσιανών? Με ποιό τρόπο γίνεται η εκπαίδευση ενός τέτοιου μοντέλου?

Η δομή ενός ακουστικού μοντέλου είναι οι εξής:

- Το κομμάτι του HMM που μοντελοποιεί σε ένα αυτόματο τις πιθανότητες μετάβασης από ένα φώνημα σε ένα άλλο.

- Το κομμάτι του GMM εκπαιδεύεται και ομαδοποιεί στο αυτόματο ορισμένες υποκατηγορίες. Η εκπαίδευση γίνεται έχοντας (όπως στην περίπτωσή μας ένα train set όπου διαθέτει και τις μεταβάσεις από ένα φώνημα σε ένα άλλο μέσω του transcription και των ακουστικών χαρακτηριστικών που προέκυψαν μέσω των αντίστοιχων αρχείων ήχου).

Στην παρακάτω εικόνα βλέπουμε οπτικά ένα GMM-HMM:



Ακουστικό μοντέλο GMM-HMM

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?

Έστω O τα features που έχουμε εξάγει από το ακουστικό μοντέλο και W η λέξη που πιθανόν έχουμε. Τότε από τον τύπο του Bayes γνωρίζουμε ότι:

$$\mathbf{P(W \mid O) = (P(O \mid W) * P(W)) / (P(O)}$$

όπου P(W | O) είναι η πιθανότητα να έχουμε την λέξη W ενώ έχουμε παρατηρήσει τα features O, P(O | W) είναι η πιθανότητα να παρατηρήσουμε τα features O ενώ έχει ειπωθεί η λέξη W (ακουστικό μοντέλο), P(W) είναι η πιθανότητα να έχουμε την λέξη W (γλωσσικό μοντέλο) και P(O) η πιθανότητα να παρατηρήσουμε τα features O.

Η πιο πιθανή λέξη δεδομένης μίας ακολουθίας ακουστικών χαρακτηριστικών ισούται με:

$$\mathbf{argmax_{\sigma\varepsilon\ \acute{o}\lambda\alpha\ \tau\alpha\ w}\ P(W \mid O) = argmax_{\sigma\varepsilon\ \acute{o}\lambda\alpha\ \tau\alpha\ w}\ (P(O \mid W) * P(W))}$$

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

Ο γράφος HCLG του Kaldi προκύπτει από την σύνθεση των τεσσάρων παρακάτω γράφων:

- Grammar: που είναι το γλωσσικό μας μοντέλου.
- Lexicon: που στην περίπτωσή μας είναι μία αντιστοίχηση των φωνημάτων στον εαυτό τους αφού θέλουμε να κάνουμε αναγνώριση φωνημάτων.
- Context dependency: mapping από context dependend labels σε φωνήματα.

- Hidden Markov Model: mapping από μεταβάσεις ενός κρυφού μαρκοβιανού μοντέλου σε context dependend labels.

Βιβλιογραφία

- https://dsp.stackexchange.com/questions/19564/cepstral-mean-normalization
- https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html
- https://www.semanticscholar.org/topic/Cepstral-Mean-and-Variance-Normalization/551159
- https://github.com/kaldi-asr/kaldi