Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ροή Σ: Επεξεργασία Φωνής και Φυσικής Γλώσσας

(10ο Εξάμηνο)

Εργαστηριακή Άσκηση 2

Δημήτρης Μπακάλης 03118163

Εισαγωγή:

Στη συγκεκριμένη εργαστηριακή άσκηση θα ασχοληθούμε με προβλήματα επεξεργασίας φωνής, χρησιμοποιώντας το λογισμικό KALDI. Πιο συγκεκριμένα, αφού εγκαταστήσουμε αυτό το πακέτο και δημιουργήσουμε μερικά βασικά αρχεία θα δημιουργήσουμε γλωσσικά μοντέλα (unigrams, bigrams), θα εξάγουμε χαρακτηριστικά για τα δεδομένα μας (Mel-Frequency Cepstral Coefficients) και θα εκπαιδεύσουμε ορισμένα ακουστικά μοντέλα , για να δούμε πώς ανταποκρίνονται στο συγκεκριμένο πρόβλημα. Τέλος, στην αρχιτεκτονική GMM-HMM (Gaussian Mixture Model – Hidden Markov Model) που θα βασιστεί η εργασία, θα επιχειρήσουμε να αντικαταστήσουμε το GMM με ένα DNN (Deep Neural Network) και να πραγματοποιήσουμε τις αντίστοιχες συγκρίσεις σχετικά με την επίδοση.

Χρήσιμες Έννοιες:

- MFCCs (Mel-Frequency Cepstral Coefficients):

  Τα MFCCs αποτελούν χαρακτηριστικά , τα οποία χρησιμοποιούνται κατά κόρον σε προβλήματα ASR (Automatic Speech Recognition) και εξάγονται από τα εξής βήματα:

  - Pre-emphasis
  - Framing
  - Windowing
  - Fast Fourier Transform (FFT)
  - Mel Filtering
  - Logarithm
  - Discrete Cosine Transform (DCT)

  Από τα εξαγόμενα coefficients, τα 12-13 πρώτα (13 στη συγκεκριμένη περίπτωση) είναι τα MFCCs.

- Γλωσσικά Μοντέλα:

  Είναι μοντέλα, τα οποία προβλέπουν τις κατανομές πιθανότητας λέξεων (ή φωνημάτων στη δική μας περίπτωση). Σε αυτή την εργασία θα χρησιμοποιηθούν unigrams και bigrams, ωστόσο, υπάρχουν και άλλα τέτοια μοντέλα όπως (n-grams με n >= 2, RNNs, LSTMs, Trasformers κλπ).

- Ακουστικά Μοντέλα:

  Είναι μοντέλα, τα οποία μοντελοποιούν τις σχέσεις μεταξύ ακουστικών χαρακτηριστικών και γλωσσικού περιεχομένου (λέξεις, φωνήματα κλπ). Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκαν GMM-HMM και DNN-HMM μοντέλα.

Μέρος 3 (Βήματα προπαρασκευής)

Τα βήματα της προπαρασκευής συνοψίζονται στο script mk_files.sh, στο οποίο περιέχονται οι εντολές για τη δημιουργία καθενός από τα αρχεία της εκφώνησης. Αρχικά, στο αχρείο δημιουργείται ο φάκελος data και οι αντίστοιχοι υποφάκελοι data/train,dev,test. Στη συνέχεια σε κάθε έναν από τους παραπάνω υποφακέλους δημιουργούνται τα αρχεία uttids, utt2spk, wav.scp και text. Αναλυτικότερα:

Για το uttids:

Στο συγκεκριμένο αρχείο περιέχονται τα ids κάθε πρότασης του συνόλου δεδομένων. Προκειμένου να δημιουργηθεί το αρχείο, απλά αντιγράφεται το περιεχόμενο των αντίστοιχων αρχείων στον φάκελο filesets που δίνεται. Το περιεχόμενό του συνοψίζεται στην ακόλουθη εικόνα:

```
1  f1_003
2  f1_004
3  f1_005
4  f1_007
5  f1_008
6  f1_009
7  f1_010
8  f1_012
9  f1_013
10 f1_015
```

Για το utt2spk:

Στο συγκεκριμένο αρχείο περιέχονται τα ids της κάθε πρότασης και αυτό του αντίστοιχου ομιλητή. Το script, το οποίο καλείται από το mk_files.sh, και δημιουργεί αυτό το αρχείο είναι το mk_utt2spk.py και, απλά διαβάζει το uttids και απομονώνει τους 2 πρώτους χαρακτήρες κάθε γραμμής, καθώς αποτελούν το id του ομιλητή. Το περιεχόμενό του συνοψίζεται στην ακόλουθη εικόνα:

```
1  f1_003 f1
2  f1_004 f1
3  f1_005 f1
4  f1_007 f1
5  f1_008 f1
6  f1_009 f1
7  f1_010 f1
8  f1_012 f1
9  f1_013 f1
10 f1_015 f1
```

Για το wav.scp:

Στο συγκεκριμένο αρχείο περιέχονται τα ids κάθε πρότασης και το path στο οποίο είναι αποθηκευμένα το είναι το ./wav/uttid.wav, με δεδομένο ότι βρισκόμαστε στο directory usc. Το script, το οποίο καλείται από το mk_files.sh, και δημιουργεί αυτό το αρχείο είναι το mk_wavs.py. Το περιεχόμενό του συνοψίζεται στην ακόλουθη εικόνα:

```
f1_003 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_003.wav
f1_004 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_004.wav
f1_005 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_005.wav
f1_007 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_007.wav
f1_008 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_008.wav
f1_009 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_009.wav
f1_010 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_010.wav
f1_012 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_012.wav
f1_013 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_013.wav
f1_015 /home/dimitris/Desktop/SLP/lab2/kaldi/egs/usc/wav/f1_015.wav
```

Για το text:

Στο συγκεκριμένο αρχείο περιέχονται τα ids κάθε πρότασης και ,αρχικά, η αντίστοιχη πρόταση σε text μορφή (αυτή η πληροφορία υπάρχει στο αρχείο transcriptions.txt). Στη συνέχεια, μέσω του αρχείου lexicon.txt αντιστοιχίζουμε κάθε λέξη στα φωνήματα που την αποτελούν, προσθέτοντας το ειδικό φώνημα σιωπής στην αρχή και στο τέλος κάθε πρότασης. Το script, το οποίο καλείται από το mk_files.sh, και δημιουργεί αυτό το αρχείο είναι το mk_text.py. Το περιεχόμενό του συνοψίζεται στην ακόλουθη εικόνα:

```
f1_003 sil sh iy ih z th ih n er dh ae n ay ae m sil
f1_004 sil b r ay t s ah n sh ay n sh ih m er z aa n dh iy ow sh ah n sil
f1_005 sil n ah th ih ng ih z ae z ah f eh n s ih v ae z ih n ah s ah n s sil
f1_007 sil w eh r w er y uw w ay l w iy w er ah w ey sil
f1_008 sil aa r y ao r g r ey d z hh ay er ao r l ow er dh ae n n ae n s iy z sil
f1_009 sil hh iy w ih l ah l aw ah r eh r l ay sil
f1_010 sil w ih l r aa b ah n w eh r ah y eh l ow l ih l iy sil
f1_012 sil b ih f ao r th er z d iy z ih g z ae m r iy v y uw eh v er iy f ao r m y ah l ah sil
f1_013 sil dh iy m y uw z iy ah m hh ay er z m y uw z ih sh ah n z eh v er iy iy v n ih ng sil
f1_015 sil k aa r l l ih v z ih n ah l ay v l iy hh ow m sil
```

Μέρος 4

4.1 Προετοιμασία διαδικασίας αναγνώρισης φωνής για τη USC-TIMIT

Σε αυτό το βήμα, αρχικά, θα αντιγράψουμε τα αρχεία path.sh και cmd.sh από τη διαδικασία για τη Wall Street Journal και θα κάνουμε τις αντίστοιχες αλλαγές, όπως ορίζονται από την εκφώνηση. Σε κάθε bash script από εδώ και πέρα θα κάνουμε source το αρχείο path.sh, ώστε να μπορούμε να χρησιμοποιήσουμε τις εντολές του kaldi.

Τα επόμενα μέρη αυτού του βήματος αφορούν στη δημιουργία ορισμένων symbolic links (για τους φακέλους steps, utils και το αρχείο score_kaldi.sh της wsj) και τη δημιουργία των φακέλων data/lang, data/local/dict, data/local/lm_tmp, data/local/nist_lm. Οι εντολές για την πραγματοποίηση αυτής της διαδικασίας περιέχονται στο script 4_1.sh.

4.2 Προετοιμασία γλωσσικού μοντέλου

Στο συγκεκριμένο βήμα, αρχικά, θα δημιουργήσουμε μερικά αρχεία, στα οποία θα βασιστεί το γλωσσικό μοντέλο. Ειδικότερα, τα αρχεία αυτά είναι:

- silence_phones.txt, optional_silence.txt: Θα περιέχεται απλά το φώνημα σιωπής sil.
- nonsilence_phones.txt: Θα περιέχονται όλα τα υπόλοιπα φωνήματα (sorted), δηλαδή όλα τα distinct φωνήματα που υπάρχουν στο lexicon.txt.
- lexicon.txt: Μία 1-1 αντιστοίχιση όλων των φωνημάτων (συμπεριλαμβανομένου του sil) με τον εαυτό τους.
- lm_train/dev/test.text: Θα περιέχονται όλες οι προτάσεις των αρχείων text των directories train, dev και test, με τις ειδικές μονάδες <s>, </s> στην αρχή και στο τέλος, αντίστοιχα, κάθε πρότασης (τα nonsilence_phones.txt και lm_train/dev/test.text δημιουργούνται από το script mk_dict.py).
- extra_questions.txt: Ένα κενό αρχείο.

Στη συνέχεια, μέσω της εντολής build-lm.sh του πακέτου IRLSTM, θα δημιουργήσουμε αρχεία, τα οποία αποτελούν μία ενδιάμεση μορφή του γλωσσικού μοντέλου, βασισμένη στο αρχείο lm_train.text (θα δημιουργηθούν unigram και bigram μοντέλα). Τα αρχεία αυτά γίνονται, έπειτα, compiled, μέσω της εντολής compile-lm, με αποτέλεσμα, το γλωσσικό μοντέλο να είναι, πλέον, σε μορφή ARPA.

Αφού αποθηκεύσουμε το γλωσσικό μοντέλο, θα χρησιμοποιήσουμε την εντολή prepare_lang.sh του kaldi, για να δημιουργήσουμε το FST του λεξικού της γλώσσας (θα αποθηκευτεί στο data/lang/L.fst).

Τα επόμενα βήματα είναι να ταξινομήσουμε τα αρχεία wav.scp, utt2spk και text (που είχαμε δημιουργήσει ως μέρος της προπαρασκευής), μέσω της εντολής sort και να δημιουργήσουμε το αρχείο spk2utt, εκτελώντας το script utt2spk_to_spk2utt.pl του φακέλου utils.

Τέλος, θα εκτελέσουμε και το δοθέν script timit_format_data.sh, προκειμένου να δημιουργηθεί και το FST της γραμματικής G.fst.

Όλη η διαδικασία που περιεγράφηκε σε αυτό το βήμα θα πραγματοποιηθεί εκτελώντας το script 4_2.sh.

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

Το perplexity αποτελεί μία μετρική για κάνουμε το evaluation για τα γλωσσικά μοντέλα που δημιουργήσαμε. Μικρότερο perplexity σηματοδοτεί και μικρότερη εντροπία, οπότε όσο μικρότερη είναι η τιμή του, τόσο το καλύτερο. Ο μαθηματικός τύπος για τον υπολογισμό του είναι ο ακόλουθος:

- PP = $\sqrt[N]{\frac{1}{P(p1,p2,\dots,pN)}}$

Με P(p_1, p_2, …, p_N) = $\prod_{i=1}^{N} P(p_i)$ για unigram
και P(p_1, p_2, …, p_N) = P(p_1) $\prod_{i=2}^{N} P(p_i)P(\,p_i|p_{i-1})$ για bigram

Στη συγκεκριμένη περίπτωση το PP υπολογίστηκε μέσω της εντολής compile-lm του kaldi (compute_perplexity.sh) και προέκυψαν τα αντίστοιχα αποτελέσματα για τα validation και test stes:

	Perplexity	
	Validation set	Test set
unigram	32.43	31.98
bigram	17.06	36.89

Όπως ήταν αναμενόμενο παρατηρείται σημαντική μείωση του perplexity στο bigram σε σχέση με το unigram και στα δύο sets.

4.3 Εξαγωγή ακουστικών χαρακτηριστικών

Στο συγκεκριμένο βήμα θα εξάγουμε τα χαρακτηριστικά, που θα χρησιμοποιηθούν για την εκπαίδευση των γλωσσικών μοντέλων στο επόμενο βήμα. Πιο συγκεκριμένα θα χρησιμοποιηθούν τα χαρακτηριστικά Mel-Frequency Cepstral Coefficients (MFCCs), τα οποία εξαχθούν και για τα 3 sets, μέσω των εντολών του kaldi make_mfcc.sh και compute_cmvn_stats.sh. Για να πραγματοποιηθεί η εξαγωγή των MFCCs, αρκεί να εκτελεστεί το script 4_3.sh.

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

Το Cepstral Mean and Variance Normalization (CMVN) χρησιμοποιείται προκειμένου να βελτιστοποιηθούν τα MFCCs που έχουμε για features. Πιο συγκεκριμένα, αυτή η κανονικοποίηση είναι ιδιαίτερα χρήσιμη σε περιπτώσεις που τα recordings χαρακτηρίζονται

από υψηλά επίπεδα θορύβου και επηρεάζουν πολύ την επίδοση του μοντέλου. Η κανονικοποίηση αποτελείται από 2 βήματα:

- Cepstral Mean Normalization (CMN): Σε αυτό το βήμα αφαιρείται η μέση τιμή του συγκεκριμένου segment των MFCCs.

- Cepstral Variance Normalization (CVN): Τέλος, διαιρούμε με την αντίστοιχη διασπορά που προκύπτει και έχουμε τα κανονικοποιημένα feature vectors.

Τα παραπάνω συνοψίζονται στους ακόλουθους μαθηματικούς τύπους:

$x\hat{}_{t-D} = \frac{x_{t-D} - \mu_t(i)}{\sigma_t(i)}$, με

- $\mu_t(i) = \frac{1}{N}\sum_{j=1}^{N} x_j(i)$

- $\sigma_t(i) = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_j(i) - \mu_t(i))^2}$

Όπου t-D είναι ο χρόνος (D το delay) και i είναι το στοιχείο του feature vector.

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

Για να εξάγουμε τη διάσταση και τον αριθμό των ακουστικών frames για τις πρώτες 5 λέξεις θα χρησιμοποιήσουμε τις εντολές του kaldi feat-to-dim και feat-to-len (dim_len_of_frames.sh), αντίστοιχα. Η διάσταση των χαρακτηριστικών προκύπτει 13, ενώ τα frames φαίνονται στην ακόλουθη εικόνα:

```
f1_003 317
f1_004 371
f1_005 399
f1_007 328
f1_008 464
```

4.4 Εκπαίδευση ακουστικών μοντέλων και αποκωδικοποίηση προτάσεων

Αφού εξάγουμε τα MFCCs, το επόμενο βήμα είναι να προχωρήσουμε στην εκπαίδευση ενός monophone GMM-HMM acoustic model στα train δεδομένα μας. Αυτό θα γίνει μέσω του script train_mono.sh του φακέλου steps. Όπως φαίνεται και από την εκφώνηση, όταν ολοκληρωθεί η εκπαίδευση θα χρησιμοποιήσουμε τη γραμματική του G.fst, καθώς και το αρχείο mkgraph.sh του φακέλου utils, για να δημιουργηθεί ο γράφος HCLG.

Τέλος, μέσω του αλγορίθμου Viterbi θα αποκωδικοποιήσουμε τις προτάσεις των dev και test set, τόσο για το unigram, όσο και για το bigram μοντέλο (μέσω της εντολής decode.sh του φακέλου steps) και θα πραγματοποιήσουμε το τελικό evaluation του κάθε μοντέλου συγκρίνοντας τη μετρική Phone Error Rate (PER), που συνοψίζεται στον παρακάτω μαθηματικό τύπο:

- PER = $100\,\frac{\text{insertions} + \text{substitutions} + \text{deletions}}{\#\text{phonemes}}$

Τα αποτελέσματα για το monophone acoustic model παρατίθενται στο παρακάτω πινακάκι:

	Phone Error Rate	
	Validation set	Test set
unigram	52.43%	51.66%
bigram	47.28%	45.39%

Σημείωση: Στη διαδικασία του scoring περιέχονται 2 υπερπαράμετροι, οι οποίες είναι η ελάχιστη και η μέγιστη τιμή LM-weight για lattice rescoring, όπως φαίνεται και στο script score_kaldi.sh:

```
--min_lmwt <int>                 # minumum LM-weight for lattice rescoring "
--max_lmwt <int>                 # maximum LM-weight for lattice rescoring "
```

Στη συγκεκριμένη περίπτωση, οι υπερπαράμετροι λαμβάνουν τις τιμές 7 και 17, αντίστοιχα.

Στη συνέχεια, αφού παράγουμε τα τελικά alignments από το monophone, τα χρησιμοποιούμε για να εκπαιδεύσουμε ένα triphone μοντέλο (2,000 leaves, 10,000 gaussians) και επαναλαμβάνουμε την παραπάνω διαδικασία. Το αντίστοιχο πινακάκι που προκύπτει είναι το ακόλουθο:

	Phone Error Rate	
	Validation set	Test set
unigram	41.15%	39.06%
bigram	36.89%	34.66%

Συγκρίνοντας τα πινακάκια παρατηρούμε την αισθητή μείωση του PER (της τάξης του 10%) στην περίπτωση του triphone, καθώς και τη σταθερά καλύτερη επίδοση των bigram μοντέλων (μείωση της τάξης του 5% έναντι του αντίστοιχου unigram).

Σημείωση: Όλη η παραπάνω διαδικασία μπορεί να πραγματοποιηθεί εκτελώντας το script 4_4.sh.

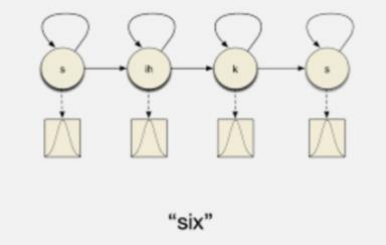
Ερώτημα 4: Εξηγήστε τη δομή ενός ακουστικού μοντέλου GMM-HMM. Τι σκοπό εξυπηρετούν τα μαρκοβιανά μοντέλα στη συγκεκριμένη περίπτωση και τι τα μίγματα γκαουσιανών? Με ποιό τρόπο γίνεται η εκπαίδευση ενός τέτοιου μοντέλου? Περιγράψτε τη διαδικασία εκπαίδευσης ενός μονοφωνικού μοντέλου.

Ένα ακουστικό μοντέλο GMM-HMM αποτελείται από δύο κύρια components, τα οποία είναι το GMM (Gaussian Mixture Model) και το HMM (Hidden Markov Model). Πιο συγκεκριμένα, ο ρόλος του κάθε component είναι ο ακόλουθος:

- GMM: Χρησιμοποιείται για τη μοντελοποίηση των κατανομών πιθανοτήτων των χαρακτηριστικών (MFCCs), το οποίο επιτυγχάνεται με χρήση ενός μίγματος από γκαουσιανές κατανομές.

- HMM: Χρησιμοποιείται για την εξαγωγή των sequences από states φωνημάτων, στα οποία αντιστοιχεί μία πιθανότητα, βάσει των GMMs. Επομένως, μέσω του HMM, επιχειρούμε την τελική πιο πιθανή αλληλουχία φωνημάτων.

Οι trainable παράμετροι του μοντέλου είναι τόσο, αυτές των GMMs (mean, variance), όσο και αυτές του HMM (πίνακας μεταβάσεων, αρχικές πιθανότητες) και εκπαιδεύονται με τον αλγόριθμος Expectation-Maximization (EM). Ένα μονοφωνικό, τέτοιο, μοντέλο οπτικά αποτυπώνεται όπως στην παρακάτω εικόνα:



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?

Με S τα states και Ph τα φωνήματα, εξάγουμε από τα GMMs τις πιθανότητες P(S|Ph). Προκειμένου να υπολογίσουμε τις αντίστοιχες a posteriori πιθανότητες χρησιμοποιούμε τον τύπο του Bayes. Επομένως, έχουμε:

- $P(Ph|S) = P(S|Ph)*P(Ph)/P(S)$

Στη συνέχεια, θέλουμε να μεγιστοποιήσουμε αυτή την πιθανότητα του φωνήματος δεδομένων των ακουστικών χαρακτηριστικών, άρα το αποτέλεσμα δίνεται από τον τύπο:

- $\text{argmax}_{Ph} (P(Ph|S)) = \text{argmax}_{Ph} (P(S|Ph)*P(Ph)/P(S)) = \text{argmax}_{Ph} (P(S|Ph)*P(Ph))$

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

Ο γράφος HCLG του kaldi αποτελείται από τη σύνθεση των παρακάτω components:

1. HMM: Ένα HMM μοντέλο για την πρόβλεψη της ακολουθίας φωνημάτων, όπως περιεγράφηκε παραπάνω.

2. Context Dependency: Το συγκεκριμένο component αφορά το context των φωνημάτων για τις μεταβάσεις του HMM.

3. Lexicon: Αντιστοιχίζει τις λέξεις σε φωνήματα (L.fst).

4. Grammar: Το γλωσσικό μοντέλο που χρησιμοποιείται (G.fst).

4.5 Μοντέλο DNN-HMM με PyTorch

Στο συγκεκριμένο μέρος θα βασιστούμε σε μεγάλο μέρος στον κώδικα που μας δίνεται. Αρχικά, θα εξάγουμε τα triphone alignments από το bigram μοντέλο (επειδή αυτό είχε καλύτερη επίδοση) για train, validation και test sets. Στη συνέχεια, θα εξάγουμε τα cmvn stats όπως υποδεικνύεται από τον κώδικα στο run_dnn.sh.

Έπειτα, θα διαβάσουμε τα δεδομένα μέσω του script timt_dnn.sh, το οποίο χρησιμοποιεί την κλάση TorchSpeechDataset του torch_dataset.py, προκειμένου να δημιουργηθούν οι dataloaders για την εκπαίδευση του μοντέλου. Το επόμενο βήμα είναι να δημιουργήσουμε το μοντέλο, το οποίο αποτελείται από:

- Input layer διαστάσεων 128*195 (batch_size*n_features)
- Hidden layer διάστασης 128
- 1D Batch Normalization
- ReLU activation
- Dropout layer (p=0.2)
- Output layer

Για την εκπαίδευση χρησιμοποιήθηκε adam optimizer (με learning rate = 0.001), Cross Entropy για loss function, καθώς και early stopping (με patience 3), για να αποφευχθεί το overfitting.

Αξίζει να σημειωθεί πως χρησιμοποιήθηκε ένα αρκετά μικρό μοντέλο, λόγω περιορισμών στην μνήμη του VM. Για τον ίδιο λόγο, χρησιμοποιήθηκαν τα 4000 πρώτα δείγματα του training set, αφού μετά από αυτά σκοτωνόταν η διεργασία.

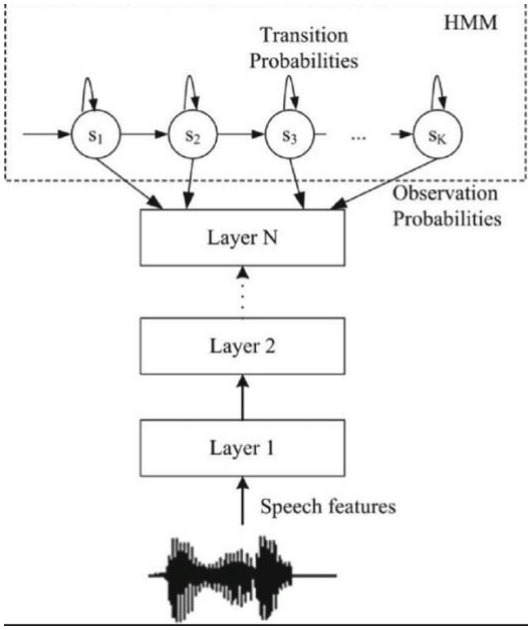
Το τελικό PER (Phone Error Rate) μετά την εκπαίδευση του νευρωνικού ήταν 37.26%. Θα μπορούσαμε, ωστόσο, να επιτύχουμε ένα αρκετά καλύτερο PER με ένα πιο βαθύ νευρωνικό περισσότερων παραμέτρων.

Σημείωση: Όλη η παραπάνω διαδικασία μπορεί να πραγματοποιηθεί εκτελώντας το script 4_5.sh.

Ερώτημα 7: Εξηγήστε σε τι διαφέρει το DNN-HMM από το GMM-HMM και πώς συνδυάζεται το DNN με ένα HMM. Που πιθανώς να μας οφελεί η προσθήκη DNN έναντι των GMM ? Θα μπορούσαμε να έχουνε εκπαιδεύσει εξαρχής ένα DNN-HMM?

Η διαφορά μεταξύ των δύο μοντέλων σχετίζεται με το πως γίνεται πλέον το mapping μεταξύ ακουστικών χαρακτηριστικών και φωνημάτων πιθανότητες, καθώς το μίγμα γκαουσιανών αντικαθίσταται από ένα βαθύ νευρωνικό δίκτυο, το οποίο, ανάλογα και με το πλήθος των παραμέτρων του, μπορεί να δημιουργήσει πιο περίπλοκα mappings.

Επιπλέον, αλλάζει και ο τρόπος εκπαίδευσης, αφού, πλέον, η εκπαίδευση πραγματοποιείται με backpropagation και. adam optimizer για τα βάρη του δικτύου. Οπτικά, ένα μοντέλο DNN-HMM έχει την παρακάτω μορφή:



Ερώτημα 8: Εξηγήστε τη λειτουργία του Batch Normalization.

Το batch normalization είναι η διαδικασία, κατά την οποία κανονικοποιούμε το input ενός layers σε ένα νευρωνικό δίκτυο, ώστε να έχει μηδενική μέση τιμή και μοναδιαία διασπορά. Η συγκεκριμένη τεχνική έχει πολλαπλά οφέλη στην διαδικασία της εκπαίδευσης, όπως τα ακόλουθα:

- Ταχύτερη εκπαίδευση: Παρόλο που προστίθεται υπολογιστική πολυπλοκότητα σε κάθε layer, η σύγκλιση επιτυγχάνεται αρκετά γρηγορότερα. Άλλος ένας λόγος που επιταχύνεται η εκπαίδευση είναι μπορεί να χρησιμοποιηθεί μεγαλύτερο learning rate.

- Τυχαιότητα initialization: Η χρήση batch normalization επιτρέπει εντελώς τυχαία initializations των βαρών, το οποίο σε ένα βαθύ δίκτυο είναι σημαντικό.

- Overfitting: Πολλές φορές το batch normalization εισάγει λίγο θόρυβο στο δίκτυο (λειτουργεί σαν regularization), με αποτέλεσμα να συμβάλλει θετικά στη γενίκευσή του και να αποφεύγεται το overfitting.