Επεξεργασία Φωνής και Φυσικής Γλώσσας

Εργαστήριο Αναγνώρισης Φωνής με το *Kaldi Toolkit*

Κωνσταντίνος Κατσικόπουλος 03120103
Άρης Μαρκογιαννάκης 03120085

29 Απριλίου 2024

1 Περιγραφή

2 Θεωρητικό Υπόβαθρο

2.1 MFCCs

Οι συντελεστές $Mel - frequency\ Cepstral\ Coefficients$ ($MFCCs$) είναι μια ευρέως χρησιμοποιούμενη τεχνική εξαγωγής χαρακτηριστικών στην επεξεργασία και αναγνώριση ομιλίας. Τα βήματα για τον υπολογισμό των συντελεστών είναι τα εξής:

- Προέμφαση: Εφαρμόζουμε ένα φίλτρο στο σήμα για να ενισχύσουμε τις υψηλές συχνότητες. Με αυτό τον τρόπο εξισσοροπούμε το συχνοτικό φάσμα του σήματος, αποφεύγουμε τα αριθμητικά λάθη στον Μετασχηματισμό $Fouries$ και βελτιώνουμε το SNR.
- $Framing$: Χωρίζουμε το σήμα σε $frames$ μεγέθους $20ms - 40ms$ με περίπου 50% επικάλυψη, καθώς το συχνοτικό περιεχόμενο του σήματος αλλάζει με την πάροδο του χρόνου και δεν θα παίρναμε κάποια χρήσιμη πληροφορία εξετάζοντας όλο το σήμα μαζί.
- Παραθυροποίηση: Έπειτα σε κάθε $frame$ εφαρμόζουμε μία συνάρτηση παραθύρου(π.χ. $Hamming$) για να αντιμετωπίσουμε την υπόθεση που κάνει ο FFT ότι τα δεδομένα είναι άπειρα και για να μειώσουμε τη φασματική διαρροή.
- FFT και Φάσμα Ισχύος: Έπειτα σε κάθε $frame$ εφαρμόζουμε τον $N - point\ FFT$ (συνήθως με N 256 ή 512) και για κάθε πλαίσιο x_i υπολογίζουμε το φάσμα ισχύος ως: $P = \frac{FFT(x_i)^2}{N}$
- $FilterBanks$: Εφαρμόζουμε τριγωνικά φίλτρα, συνήθως 40, σε κλίμακα Mel στο φάσμα ισχύος για να εξάγουμε τις μπάντες συχνοτήτων. Η κλίμακα Mel έχει ως στόχο να μιμηθεί τη μη γραμμική αντίληψη του ήχου από το ανθρώπινο αυτί, με μεγαλύτερη διακριτική ικανότητα στις χαμηλές συχνότητες και μικρότερη στις υψηλές συχνότητες.
- DCT: Έπειτα εφαρμόζουμε DCT μετασχηματισμό για να αποσύσχετίσουμε τους συντελεστές του $fitler\ bank$ και να πάρουμε μία πιό συμπιεσμένη αναπαράσταση του $filter\ bank$.
- $Mean\ Normalization$: Για να εξισορροπήσουμε το φάσμα και να βελτιώσουμε το SNR μπορούμε να αφαιρέσουμε τον μέσο όρο των $frames$ από κάθε $frame$.

Με την έλευση της Βαθιάς Μάθησης στα συστήματα ομιλίας οι $MFCCs$ μπορεί να μην είναι η βέλτιστη επιλογή αφού τα νευρωνικά δίκτυα δεν είναι τόσο ευαίσθητα σε υψηλά συσχετισμένες εισόδους και επομένως ο DCT δεν είναι ένα απαρραίτητο βήμα. Επίσης ο DCT είναι ένας γραμμικός μετασχηματισμός και επομένως ανεπιθύμητος, καθώς απορρίπτει ορισμένες πληροφορίες στα σήματα ομιλίας που είναι εξαιρετικά μη γραμμικά. Επειδή ο Μετασχηματισμός Φουριερ είναι επίσης μια γραμμική πράξη, μπορούμε να τον αγνοήσουμε και να προσπαθήσουμε να μάθουμε απευθείας από το σήμα στο πεδίο του χρόνου.

2.2 Γλωσσικό Μοντέλο

Ως γλωσσικά μοντέλα στην παρούσα εργασία χρησιμοποιούμε *unigams* και *bigrams*. Στα *unigrams* η πιθανότητα εμφάνισης ενός φωνήματος/λέξης είναι ανεξάρτητη από όλα τα υπόλοιπα φωνήματα/λέξεις του κειμένου ενώ στα *bigrams* η πιθανότητα εμφάνισης ενός φωνήματος/λέξης εξαρτάται μόνο από το ακριβώς προηγούμενο φώνημα. Και τα δύο μοντέλα δεν είναι τα καλύτερα δυνατά καθώς αδυνατούν να εντοπίσουν σχέσεις ανάμεσα σε φωνήματα/λέξεις που απέχουν αρκετά μεταξύ τους μέσα στο κείμενο.

2.3 Ακουστικό Μοντέλο

Ως ακουστικό μοντέλο στην παρούσα εργασία χρησομοποιούμε το HMM. Το κρυφό μοντέλο *Markov* είναι ένα πιθανοτικό μοντέλο που χρησιμοποιείται για τη μοντελοποίηση ακολουθιών παρατηρήσεων. Αποτελείται από ένα σύνολο κρυφών καταστάσεων, καθεμία από τις οποίες συνδέεται με μια κατανομή πιθανότητας επί των παρατηρούμενων συμβόλων, και ένα σύνολο πιθανοτήτων μετάβασης μεταξύ των καταστάσεων. Το HMM υποθέτει ότι τα παρατηρούμενα δεδομένα παράγονται από μια διαδικασία με κρυφές καταστάσεις που εξελίσσονται με την πάροδο του χρόνου σύμφωνα με μια μαρκοβιανή διαδικασία, και τα παρατηρούμενα σύμβολα εκπέμπονται από κάθε κατάσταση με βάση την κατανομή πιθανότητας εκπομπής της.
Τα $HMMs$ μπορούν να μοντελοποιήσουν αποτελεσματικά πολύπλοκα διαδοχικά μοτίβα με σχετικά λίγες παραμέτρους, καθιστώντας τα υπολογιστικά εφικτά για πολλές εφαρμογές και επίσης παρέχουν ένα σαφές πιθανολογικό πλαίσιο για τη μοντελοποίηση ακολουθιών, επιτρέποντας την απλή ερμηνεία υποκείμενων διαδικασιών. Όμως τα $HMMs$ υποθέτουν ότι οι παρατηρήσεις είναι υπό όρους ανεξάρτητες δεδομένης της κρυφής κατάστασης, κάτι που συνήθως δεν ισχύει και έχουν σταθερή τοπολογία και έτσι δεν μπορούν να μοντελοποιήσουν σύνθετες εξαρτήσεις μεταξύ των δεδομένων.

3 Βήματα Προπαρασκευής

Αρχικά μέσα στον φάκελο $\sim /kaldi/egs$ φτιάχνουμε τον φάκελο *usc* και στην συνέχεια μέσα στο *usc* τους φακέλους $data/train$, $data/dev$ και $data/dev$ όπου θα αποθηκεύσουμε τα δεδομένα μας.
Μέσα σε κάθε έναν από τους 3 παραπάνω φακέλους υπάρχουν τα αρχεία:

- *uttids*: Περιέχει στην κάθε του γραμμή ένα μοναδικό συμβολικό όνομα για κάθε πρόταση του συγκεκριμένου συνόλου δεδομένων. Στην ουσία περιέχει αυτούσιο το περιεχόμενο του αντίστοιχου αρχείου $usc/filesets/\{training, testing, validation\}.txt$ που μας δόθηκε στην εκφώνηση. Τα αρχεία δημιουργήθηκαν με το **uttids.sh**.

- *utt2spk*: Περιέχει σε κάθε γραμμή τον ομιλητή που αντιστοιχεί σε κάθε πρόταση. Οι ομιλητές είναι οι $f1, f5, m1, m3$. Τα αρχεία δημιουργήθηκαν με το **utt2spks.sh**.

- *wav.scp*: Περιέχει τη θέση του αρχείου ήχου που αντιστοιχεί σε κάθε πρόταση. Αντιστοιχούμε το κάθε *utterance_id* με το κατάλληλο αρχείο του φακέλου *wav* που μας δόθηκε. Τα αρχεία δημιουργήθηκαν με το **wav.sh**.

- *text*: Περιέχει το κείμενο που αντιστοιχεί στην κάθε πρόταση. Αντιστοιχούμε το κάθε *utterance_id* με την κατάλληλη πρόταση στο αρχείο *transcriptions.txt*.Τα αρχεία δημιουργήθηκαν με το **text.sh**.

Τέλος, χρησιμοποιώντας το **text_to_phoneme.py** μετατρέπουμε, στα αρχεία *text*, τις λέξεις της αγγλικής γλώσσας σε αλληλουχίες φωνημάτων κάνοντας την αντιστοίχιση με την βοήθεια του *lexicon.txt*

4 Βήματα Κυρίως Μέρους

4.1 Προετοιμασία διαδικασίας αναγνώρισης φωνής για τη USC – TIMIT

1. Από τη διαδικασία *wsj* παίρνουμε τα αρχεία *path.sh* και *cmd.sh*, θέσαμε $KALDI\_ROOT =\sim /kaldi$ στο *path.sh* και αλλάζουμε τις τιμές των μεταβλητών $train_cmd, decode_cmd$ και $cuda_cmd$ σε *run.pl* στο *cmd.sh*.

2. Έπειτα φτιάχνουμε *soft links* μέσα στο φάκελο της δικής σας διαδικασίας με ονόματα *steps* και *utils* τα οποία θα δείχνουν στους αντίστοιχους φακέλους της *wsj* χρησιμοποιώντας την εντολή:

$ln\ -s\ wsj/s5/\{steps/utils\}\ usc/\{steps, utils\}$.

3. Έπειτα φτιάχνουμε τον φάκελο *local* μέσα στο *usc* και μέσα σε αυτόν ένα $softlink$ που να δείχνει στο αρχείο $score_kaldi.sh$ πουβρίσκεται μέσα στο *steps*.

4. Έπειτα φτιάχνουμε τον φάκελο $conf$ και βάζουμε μέσα το αρχείο $mfcc.conf$ το οποίο περιέχει την συχνότητα δειγματοληψίας και θα χρησιμοποιηθεί για τον υπολογισμό των $MFCCs$.

5. Έπειτα φτιάχνουμε τους φακέλους $data/lang$, $data/local/dict$, $data/local/lm\_tmp$ και $data/local/nist\_lm$.

4.2 Προετοιμασία γλωσσικού μοντέλου

1. Αρχικά φτιάχνουμε στο $data/local/dict$ τα αρχεία $silence\_phones.txt$, $optional\_silence.txt$, $nonsilence\_phones.txt$, $lexicon.txt$, $lm\_train.txt$ και $extra\_questions.txt$ με τον τρόπο που αναφέρονται στην εκφώνηση.

2. Έπειτα με την χρήση του **boo.sh** φτιάχνουμε την ενδιάμεση μορφή του γλωσσικού μοντέλου($unigram$ και $bigram$) χρησιμοποιώντας την εντολή:
$build - lm.sh\ -i$ <αρχείο $lm\_train.text$ > -n <τάξη γλωσσικού μοντέλου> -o <αρχείο_εξόδου.$ilm.gz$ >

3. Έπειτα με την χρήση του **hoo.sh** μεταγλωττίζουμε το γλωσσικό μοντέλο και το αποθηκεύουμε σε μορφή $ARPA$. Χρησιμοποιούμε την εντολή:
$compile - lm$ <αρχείο $.ilm.gz$ > $-t = yes\ /dev/stdout$, $grep\ -v\ unk$, $gzip\ -c$ > <αρχείο_εξόδου.$arpa.gz$ >

4. Στη συνέχεια με την χρήση του **langset.sh** δημιουργούμε το FST του λεξικού της γλώσσας($L.FST$).

5. Έπειτα με την χρήση του **sort_files.sh** ταξινομούμε τα αρχεία $wav.scp$, $text$ και $utt2spk$ των φακέλων $data/train$, $data/text$ και $data/dev$.

6. Έπειτα με τη χρήση του **spk2utt.sh** δημιουργούμε τα αρχεία $spk2utt$, χρησιμοποιώντας το $utils/utt2spk\_to\_spk2utt.pl$.

7. Τέλος χρησιμοποιούμε το ελαφρώς τροποποιημένο **timit_format_data.sh** για να φτιάξουμε το FST της γραμματικής($G.FST$).

Ερώτημα 1

Χρησιμοποιώντας το **perplexity.sh** υπολογίζουμε το $perplexity$ στα dev και $test\ set$ με την εντολή:
$compile - lm\ path/to/lm\_output\_train\{u,b\}.ilm.gz --eval = path/to/lm\_\{train,dev\}.txt --dub = 10000000$

```
# Calculating perplexity for the bigram model of the dev set #
inpfile: lm_output_trainb.ilm.gz
outfile: lm_output_trainb.ilm.blm
evalfile: ../dict/lm_dev.txt
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=4930 PP=17.06 PPwp=0.00 Nbo=52 Noov=0 OOV=0.00%
# Calculating perplexity for the unigram model of the dev set #
inpfile: lm_output_trainu.ilm.gz
outfile: lm_output_trainu.ilm.blm
evalfile: ../dict/lm_dev.txt
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=4930 PP=32.43 PPwp=0.00 Nbo=0 Noov=0 OOV=0.00%
# Calculating perplexity for the bigram model of the test set #
inpfile: lm_output_trainb.ilm.gz
outfile: lm_output_trainb.ilm.blm
evalfile: ../dict/lm_test.txt
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=12795 PP=16.89 PPwp=0.00 Nbo=240 Noov=0 OOV=0.00%
# Calculating perplexity for the unigram model of the test set #
inpfile: lm_output_trainu.ilm.gz
outfile: lm_output_trainu.ilm.blm
evalfile: ../dict/lm_test.txt
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=12795 PP=31.98 PPwp=0.00 Nbo=0 Noov=0 OOV=0.00%
```

Το *perplexity* είναι ένα μέτρο του πόσο καλά ένα γλωσσικό μοντέλο προβλέπει την επόμενη λέξη/φώνημα σε ένα δείγμα κειμένου. Αντικατοπτρίζει πόσο έκπληκτο ή αβέβαιο είναι το μοντέλο όταν συναντά νέα δεδομένα. Αν ένα γλωσσικό μοντέλο έχει χαμηλό *perplexity*, αυτό σημαίνει ότι είναι καλό στο να προβλέπει την επόμενη λέξη/φώνημα σε μια ακολουθία με βάση αυτά που έχει μάθει από τα δεδομένα εκπαίδευσης. Παρατηρούμε ότι το *bigram* μοντέλο παρουσιάζει καλύτερο *pereplexity* από το *unigram* μοντέλο τόσο $dev(PP = 17.06/PP = 32.43)$ όσο και στο $test(PP = 16.89/PP = 31.98)$ *set*. Αυτό ήταν αναμενόμενο καθώς τα *unigram* μοντέλα εξετάζουν κάθε λέξη μεμονωμένα, χωρίς να λαμβάνουν υπόψη τα περιβάλλοντα συμφραζόμενα αυτής. Από την άλλη πλευρά, τα μοντέλα *bigram* λαμβάνουν υπόψη την προηγούμενη λέξη κατά την πρόβλεψη της επόμενης λέξης. Αυτή η ενσωμάτωση των συμφραζομένων επιτρέπει στα μοντέλα *bigram* να καταγράφουν καλύτερα τις εξαρτήσεις και τους συσχετισμούς μεταξύ γειτονικών λέξεων, με αποτέλεσμα κάνουν ακριβέστερες προβλέψεις.

4.3 Εξαγωγή ακουστικών χαρακτηριστικών

Χρησιμοποιώντας το **ex4_3.sh** εξάγουμε τα $MFCCs$ για τα *train*, *test* και *dev set* και πραγματοποιούμε το λεγόμενο *Cepstral Mean and Variance Normalization*, κάνοντας χρήση των εντολών *steps/make_mfcc.sh* και *steps/compute_cmvn_stats.sh*.

Ερώτημα 2

Το *Cepstral Mean and Variance Normalization* εξυπηρετεί τον σκοπό της κανονικοποίησης των $MFCCs$, αφαιρώντας τον μέσο όρο και κλιμακώνοντας με την τυπική απόκλιση σε όλα τα $frames$. Αυτή η κανονικοποίηση βοηθά στη μείωση της μεταβλητότητας των χαρακτηριστικών ομιλίας που προκαλείται από διαφορές στις συνθήκες καταγραφής, όπως τα διαφορετικά επίπεδα θορύβου στο *background* ή τα χαρακτηριστικά του μικροφώνου.

Έστω $x[n]$ το σήμα ήχου του ομιλητή και $h[n]$ η κρουστική απόκριση του καναλιού που γίνεται η ηχογράφηση.

Το τελικό σήμα της ηχογράφησης είναι το $y[n] = x[n] * h[n]$

Παίρνοντας τον Μετασχηματισμό $Fourier$ έχουμε $Y[f] = X[f] \cdot H[f]$

Έπειτα υπολογίζουμε το *cepstrum* παίρνοντας τον λογάριθμο: $Y[q] = logY[f] = log(X[f] \cdot H[f]) = X[q] \cdot H[q]$, όπου q είναι το $quefrency$.

Τώρα γνωρίζουμε ότι στο πεδίο *cepstral* οι όποιες παραμορφώσεις/αλλοιώσεις υπάρχουν στο πεδίο του χρόνου, εμφανίζονται με την μορφή αθροίσματος. Παίρνοντας τώατα την το *cesptrum* του $i-$οστού $frame$ έχουμε:

$Y_i[q] = X_i[q] + H[q]$, και ο μέσος όρος των *cepstrum* των $frames$ είναι:

$\mu = \sum_{i=1}^{N} Y_i[q] = \sum_{i=1}^{N} X_i[q] + H[q]$

Τέλος παίρνοντας την διαφορά:

$R_i[q] = Y_i[q] - \mu = X_i[q] + H[q] - \sum_{i=1}^{N} X_i[q] - H[q] = X_i[q] - \sum_{i=1}^{N} X_i[q]$

Βλέπουμε ότι με αυτό τον τρόπο εξουδετερώσαμε στην επίδραση του θορύβου και μπορούμε ακόμη να διαιρέσουμε το κάθε $R_i[q]$ με την τυπική απόκλιση $\sigma = \sqrt{\frac{\sum_{i=1}^{N}(X_i[q]-\mu)^2}{N}}$ για να κάνουμε το $Variance\ Normalization$ που μας εξασφαλίζει ότι όλα τα δεδομένα θα βρίσκονται στην ίδια κλίμακα και έτσι δεν θα κυριαρχούν τα δείγματα με τις πολύ υψηλές τιμές.

Ερώτημα 3

Χρησιμοποιώντας το **ex4_3_3.sh** υπολογίζουμε τη διάσταση των χαρακτηριστικών με την εντολή $feat-to-dim$ και τον αριθμό των $frames$ των 5 πρώτων προτάσεων του $training\ set$ με την εντολή $feat-to-len$.

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ ./ex4_3_3.sh
Executing command: feat-to-dim scp:data/train/data/raw_mfcc_train.1.scp-
feat-to-dim scp:data/train/data/raw_mfcc_train.1.scp -
13
---------------------------------------
Executing command: feat-to-len scp:data/train/feats.scp ark, t:data/train/feats.lengths
feat-to-len scp:data/train/feats.scp ark,t:data/train/feats.lengths
---------------------------------------
First 5 lines of data/train/feats.lengths:
f1_003 317
f1_004 371
f1_005 399
f1_007 328
f1_008 464
```

Παρατηρούμε ότι ο κάθε $MFCC$ έχει διάσταση 13 και οι πρώτες 5 προτάσεις του $training\ set$ έχουν 317, 371, 399, 328 και 464 $frames$ αντίστοιχα.

4.4 Εκπαίδευση ακουστικών μοντέλων και αποκωδικοποίηση προτάσεων

1. Αρχικά χρησιμοποιούμε το **ex4_4_1.sh** για να εκπαιδεύσουμε ένα $monophone\ GMM-HMM$ μοντέλο πάνω στα $training$ δεδομένα, χρησιμοποιώντας το $steps/train\_mono.sh$.

2. Έπειτα χρησιμοποιούμε το **ex4_4_2.sh** για να φτιάξουμε τον γράφο $HCLG$ του $Kaldi$ τόσο για $unigram$ όσο και για $bigram$ χρησιμοποιώντας το $utils/mkgraph.sh$.

3. Έπειτα χρησιμοποιούμε το **ex4_4_3.sh** για να αποκωδικοποιήσουμε τις προτάσεις των $test$ και $dev\ set$ με τον αλγόριθμο $Viterbi$, χρησιμοποιώντας τόσο $unigram$ γλωσσικό μοντέλο όσο και $bigram$ και κάνοντας χρήση του $steps/decode.sh$.

4.
PER με $unigram$ στο $test\ set$, $monophone$

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ cat exp/mono/decode_test_unigram/scoring_kaldi/best_wer
%WER 52.10 [ 6456 / 12392, 122 ins, 3753 del, 2581 sub ] exp/mono/decode_test_unigram/wer_7_0.0
```

PER με $bigram$ στο $test\ set$, $monophone$

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ cat exp/mono/decode_test_bigram/scoring_kaldi/best_wer
%WER 45.17 [ 5597 / 12392, 185 ins, 2583 del, 2829 sub ] exp/mono/decode_test_bigram/wer_7_0.0
```

PER με $unigram$ στο $dev\ set$, $monophone$

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ cat exp/mono/decode_dev_unigram/scoring_kaldi/best_wer
%WER 52.82 [ 2526 / 4782, 91 ins, 1359 del, 1076 sub ] exp/mono/decode_dev_unigram/wer_7_0.0
```

PER με $bigram$ στο $dev\ set$, $monophone$

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ cat exp/mono/decode_dev_bigram/scoring_kaldi/best_wer
%WER 46.72 [ 2234 / 4782, 104 ins, 948 del, 1182 sub ] exp/mono/decode_dev_bigram/wer_7_0.0
```

Παρατηρούμε ότι με το $bigram$ πετυχαίνουμε καλύτερα PER(45.17/46.72) από ότι με το $unigram$(52.18/52.82) στα $test/dev$ αντίστοιχα.
Οι υπερπαράμετροι της διαδικασίας $scoring$ είναι οι min_lmwt και max_lmwt, που καθορίζουν το ελάχιστο και το μέγιστο βάρος, αντίστοιχα, του γλωσσικού μοντέλου κατά την διάρκεια του $lattice\ rescoring$, δηλαδή στην ουσία καιθορίζουν την επιρροή του γλωσσικού μοντέλου στο τελευταίο στάδιο της αναγνώρισης. Υπάρχει επίσης η υπερπαράμετρος $word\_ins\_penalty$, η οποία τιμωρεί την προσθήκη λέξεων/φωνημάτων στο $hypothesis\ word/phoneme\ string$ καθώς τα λάθη λέξεων/φωνημάτων συχνά προκαλούνται από την εισαγωγή σύντομων λέξεων/φωνημάτων με ευρύ $context$. Και στις παραπάνω 4 περιπτώσεις τα καλύτερα $score$ τα πήραμε για $lmwt = 7$ και $word\_ins\_penalty = 0.0$.

5. Χρησιμοποιούμε το **ex4_4_5.sh** όπου αρχικά κάνουμε *align* το *monophone* μοντέλο με το *steps/align_si.sh*, εκπαιδεύουμε το *triphone* μοντέλο με το *steps/train_deltas.sh* και έπειτα φτιάχνουμε τον $HCLG$ γράφο και κάνουμε *decode* όπως στα προηγούμενα ερωτήματα.

PER με *unigram* στο *test set*, *triphone*

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ cat exp/tri/decode_test_unigram/scoring_kaldi/best_wer
%WER 39.06 [ 4840 / 12392, 447 ins, 1819 del, 2574 sub ] exp/tri/decode_test_unigram/wer_7_0.0
```

PER με *bigram* στο *test set*, *triphone*

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ cat exp/tri/decode_test_bigram/scoring_kaldi/best_wer
%WER 35.99 [ 4460 / 12392, 537 ins, 1317 del, 2606 sub ] exp/tri/decode_test_bigram/wer_7_0.0
```

PER με *unigram* στο *dev set*, *thiphone*

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ cat exp/tri/decode_dev_unigram/scoring_kaldi/best_wer
%WER 40.51 [ 1937 / 4782, 215 ins, 620 del, 1102 sub ] exp/tri/decode_dev_unigram/wer_8_0.0
```

PER με *bigram* στο *dev set*, *triphone*

```
arismarkogi@aris-laptop:~/kaldi/egs/usc$ cat exp/tri/decode_dev_bigram/scoring_kaldi/best_wer
%WER 36.41 [ 1741 / 4782, 211 ins, 501 del, 1029 sub ] exp/tri/decode_dev_bigram/wer_9_0.0
```

Παρατηρούμε ότι χρησιμοποιώντας *triphone* το *Error Rate* έχει μειωθεί και για *unigram* και για *bigram* τόσο στο *test* όσο και στο *dev set*, ενώ πάλι με το *bigram* πετυχαίνουμε καλύτερα $PER(35.99/36.41)$ από ότι με το $unigram(39.06/40.51)$ στα $test/dev$ αντίστοιχα.
Σε όλα τα *best_wer* έχουμε $word\_ins\_penalty = 0.0$, ενώ στο στο *test set* με *unigram* και *bigram* έχουμε $lmwt = 7$, στο *dev set* με *unigram* έχουμε $lmwt = 8$ και στο *dev set* με *bigram* έχουμε $lmwt = 9$.

Ερώτημα 4

Σε ένα $GMM - HMM$ κάθε κατάσταση του HMM σχετίζεται με ένα GMM που μοντελοποιεί την κατανομή πιθανότητας των παρατηρούμενων ακουστικών χαρακτηριστικών για την συγκεκριμένη κατάσταση.

HMM: Το HMM είναι ένα στατιστικό μοντέλο που χρησιμοποιείται για την περιγραφή της πιθανοτικής μετάβασης μεταξύ μιας ακολουθίας παρατηρήσιμων καταστάσεων. Στην αναγνώριση ομιλίας, τα $HMMs$ χρησιμοποιούνται για τη μοντελοποίηση των χρονικών εξαρτήσεων μεταξύ φωνημάτων στο σήμα ομιλίας. Κάθε φώνημα αντιπροσωπεύεται από μια κατάσταση στο HMM και οι μεταβάσεις μεταξύ αυτών των καταστάσεων υπαγορεύονται από τις πιθανότητες μετάβασης.

GMM: Τα GMM μοντελοποιούν την κατανομή πιθανότητας του διανύσματος χαρακτηριστικών δεδομένου ενός φωνήματος. Με αυτό τον τρόπο μπορούμε να μετρήσουμε την απόσταση μεταξύ ενός φωνήματος και του $frame$ που παρατηρείται.

Η εκπαίδευση του $GMM - HMM$ γίνεται με τον αλγόριθμο $Expectation - Maximization$. Στο Ε-βήμα για κάθε ακολουθία παρατηρήσεων υπολογίζουμε τις *posterior* πιθανότητες της κάθε κατάστασης δεδομένων των παρατηρήσεων και στο Μ-βήμα ενημερώνουμε τις παραμέτρους του GMM και του HMM έτσι ώστε να μεγιστοποιείται η συνάρτηση πιθανοφάνειας που υπολογίζεται στο Ε-βήμα.

Όταν εκπαιδεύουμε ένα *monophone* βλέπουμε το $frame$ σαν μία αλληλουχία μεμονωμένων φωνημάτων και προσπαθούμε να βρούμε ποια είναι η πιο πιθανή τέτοια ακολουθία.

Ερώτημα 5

Έστω O ένα σύνολο παρατηρήσεων από το ακουστικό μοντέλο και έστω w κάποια λέξη από το λεξικό μας, έστω W. Τότε:

$P[w|O] = \frac{P[O|w] \cdot P[w]}{P[O]}$

Επομένως για να βρούμε την πιο πιθανλη λέξη $\widetilde{w}$ δεδομένης μιας ακολουθίας χαρακτηριστικών έχουμε:

$$\widetilde{w} = \underset{w \in W}{argmax}\{P[w|O]\} = \underset{w \in W}{argmax}\{P[O|w] \cdot P[w]\}$$

Ερώτημα 6

Η γενική εικόνα για την δομή του γράφου $HCLG$ είναι ότι κατασκευάζεται ως $HCLG = H \circ C \circ L \circ G$, όπου:

- G είναι ένας *acceptor* που κωδικοποιεί την γραμματική ή το γλωσσικό μοντέλο.
- L είναι το λεξικό. Τα σύμβολα εισόδου είναι φωνήματα και τα σύμβολα εξόδου είναι λέξεις(στην περίπτωσή μας φωνήματα).
- C αναπαριστά την εξάρτηση από το περιεχόμενο. Τα σύμβολα εισόδου αναπαριστούν φωνήματα εξαρτώμενα από το περιεχόμενο και τα σύμβολα εξόδου είναι φωνήματα.
- H περιέχει τους ορισμούς του HMM. Τα σύμβολα εισόδου είναι μεταβάσεις του HMM και τα σύμβολα εξόδου είναι φωνήματα εξαρτώμενα από το περιεχόμενο.

Αυτή είναι η τυπική συνταγή. Ωστόσο, υπάρχουν πολλές λεπτομέρειες που πρέπει να συμπληρωθούν. Θέλουμε να διασφαλίσουμε ότι η έξοδος είναι *determinized* και ελαχιστοποιημένη, και προκειμένου το $HCLG$ να είναι *determinizable* πρέπει να εισάγουμε σύμβολα αποσαφήνισης. Επίσης θέλουμε να διασφαλίσουμε ότι ο $HCLG$ είναι, όσο περισσότερο μπορούμε, στοχαστικός και γι΄ αυτό προσπαθούμε να διασφαλίσουμε ότι σε κάθε βήμα κατασκευής του $HCLG$ διατηρείται η στοχαστικότητα.

4.5 Μοντέλο DNN – HMM με PyTorch

1. Αρχικά, για το *alignment* του *triphone* χρησιμοποιούμε το **ex4_5_1.sh** το οποίο κάνει *align* το μοντέλο με το καλύτερο $WER - PER$, δηλαδή εκείνο με *bigram*.

2-6. Στην συνέχεια συμπληρώνοντας κατάλληλα τον κώδικα στα αρχεία **timit_dnn.py**, **torch_dnn.py**, **torch_dataset.py**, **run_dnn.sh**, **decode_dnn.sh**, **extract_posteriors.py** τρέχουμε το **run_dnn.sh** που τόσο εκπαιδεύει το *aligned triphone* όσο και εξάγει τα $WER - PER$ από το εκπαιδευμένο DNN.

$WER - PER$ χρησιμοποιώντας $DNN - HMM$

```
arismarkogi@aris-laptop:~/kaldi/egs/usc_final$ cat exp/tri_bg/decode_dnn/scoring_kaldi/best_wer
%WER 34.60 [ 4250 / 12282, 356 ins, 1442 del, 2452 sub ] [PARTIAL] ./exp/tri_bg/decode_dnn/wer_2_0.5
```

Παρατηρούμε ότι με την χρήση $DNN-HMM$, το $WER-PER$ μειώνεται σε 34.60. Με την χρήση $GMM-HMM$ το καλύτερο $WER - PER$ που είχαμε πετύχει στο *test set* ήταν 35.99.

Ερώτημα 7
Ένα $GMM - HMM$ διαφέρει από ένα $DNN - HMM$ στην διαχείριση των παρατηρήσεων, συγκεκριμένα, οι παρατηρήσεις σε ένα $GMM - HMM$ μοντελοποιούνται ως μικτές κατανομές Γκαουσιανών, καθώς τα GMM είναι στατιστικά μοντέλα που προσπαθούν να κατανείμουν κάθε παρατήρηση σε μια Γκαουσιανή κατανομή. Από την άλλη πλευρά, τα DNN προσπαθούν να προβλέψουν την κατανομή των παρατηρήσεων. Η προσθήκη DNN συνεπώς μπορεί να ωφελήσει στην αναγνώριση κάποιων πιο σύνθετων και μη γραμμικών μοτίβων στα δεδομένα που θα οδηγήσουν σε μια ακριβέστερη πρόβλεψη της κατανομής τους. Μια εκπαίδευση ενός $DNN - HMM$ εξαρχής θα ήταν εφικτή απλά θα χρειάζονταν περισσότερα δεδομένα και χρόνο για να είναι το ίδιο αποδοτική με το $GMM - HMM \rightarrow DNN - HMM$ *pipeline* που έχει ήδη μάθει κάποιες αναπαραστάσεις από GMM.
Ερώτημα 8
Το *Batch Normalization*:

- Χωρίζει τα δεδομένα σε $mini - batches$
- Υπολογίζει τη μέση τιμή και τη διακύμανση για κάθε χαρακτηριστικό ξεχωριστά και χρησιμοποιεί αυτές τις παραμέτρους για να κανονικοποιήσει το κάθε $mini - batch$.
- Εφαρμόζεται ένας γραμμικός μετασχηματισμός της μορφής $y_i^k = \gamma^k \hat{x_i}^k + \beta^k$ όπου τα γ,β είναι παράμετροι που μαθαίνει ο *optimizer* μας

Πρακτικά δίνει μια λύση στο πρόβλημα του *exploding/vanishing gradient* και βοηθάει στην ομαλοποίηση της εκπαίδευσης του μοντέλου μας.