Επεξεργασία Φωνής και Φυσικής Γλώσσας

2η εργαστηριακή άσκηση
Αναγνώριση φωνής με το Kaldi Toolkit

Ηλίας Κουμουκέλης, el18157
Νικόλαος Παγώνας, el18175

1 Εισαγωγή

Ο κώδικας που έχει γραφτεί έχει δοκιμαστεί με Python `3.7.13`. Προτείνεται τα αρχεία python να εκτελεστούν σε ένα καινούργιο environment με την έκδοση αυτή, για να μην υπάρξουν θέματα με τυχόν πακέτα/βιβλιοθήκες.

Ο κώδικας που έχουμε γράψει βρίσκεται στον φάκελο `usc/scripts/` (εκτός από το `main.sh` που βρίσκεται έξω από το `scripts/`, στον `usc/`) και είναι χωρισμένος σε βήματα:

- `3.py`
- `4.1.py`
- `4.2.py`
- `4.3.py`
- `4.4.py`

Για να εκτελεστεί όλος ο κώδικας μαζί, πρέπει να εκτελεστεί το `main.sh` μέσα από τον φάκελο `usc/` (επομένως η εντολή `pwd` πρέπει να εκτυπώνει `path/to/usc/`, αλλιώς θα υπάρξουν σφάλματα κατά την εκτέλεση).

Επιπλέον, η default συμπεριφορά του `main.sh` είναι να ξανακατεβάζει τόσο τα δεδομένα από το Google Drive link, όσο και τα απαραίτητα αρχεία από το slp-ntua/slp-labs. Προτείνουμε να διατηρήσετε αυτή τη συμπεριφορά. Τα αρχεία κάνουν λίγα δευτερόλεπτα για να κατέβουν, και έτσι εξασφαλίζεται ότι το `main.sh` θα βρει τα αρχεία που χρειάζεται στα σωστά σημεία.

Τέλος, έχουμε γράψει κάποια convenience scripts, τα οποία χρησιμοποιούν τα παραπάνω αρχεία-βήματα:

```
build-lm-wrapper.sh       # source path.sh before calling build-lm.sh
clean.sh                  # clean everything except main.sh and scripts/
compile-lm-wrapper.sh     # source path.sh before calling compile-lm.sh
```

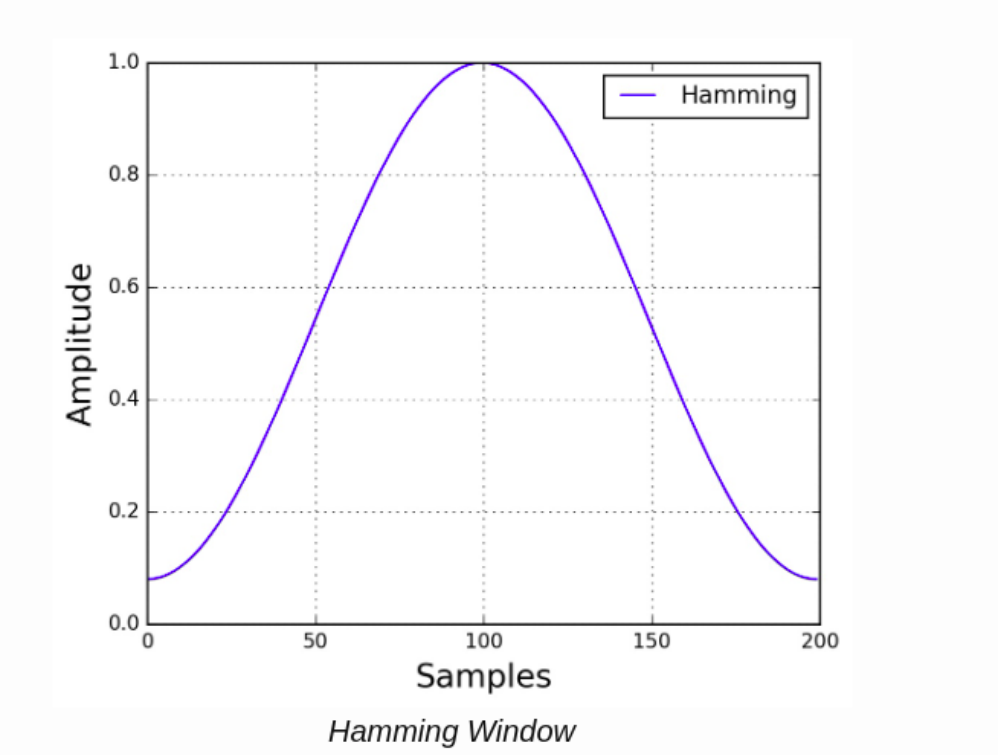
```
download.sh                 # download dependencies (Google Drive/GitHub)
helpers.py                  # for running bash commands from within python
prepare_lang-wrapper.sh     # source path.sh before calling prepare_lang.sh
timit_format_data.sh        # taken from timit/ recipe
```

2 Θεωρητικό υπόβαθρο

Mel-frequency Cepstral Coefficients (MFCCs)

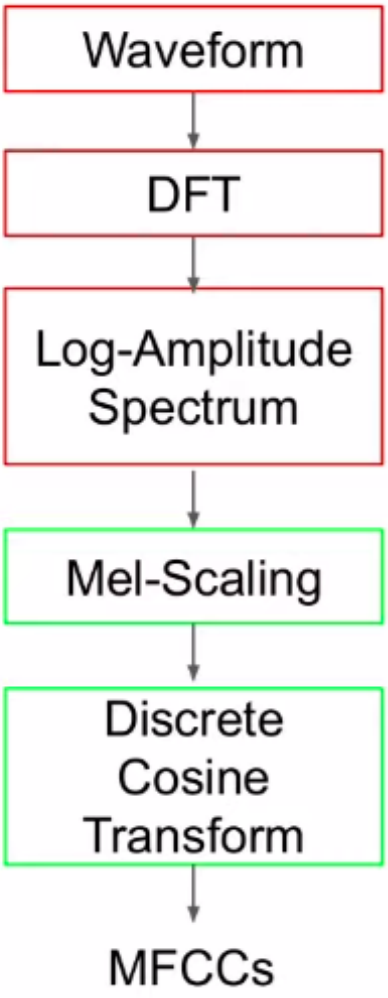
Στην αναγνώριση φωνής ένας από τους πιο γνωστούς τρόπους για την εξαγωγή χαρακτηριστικών και την αναπαράσταση τους είναι τα MFCCs. Τα Mel Frequency Cepstral Coefficients είναι βασισμένα στο Cepstrum (αντίστροφος μετασχηματισμός Fourier του λογαρίθμου του εκτιμώμενου σήματος). Η διαδικασία εξαγωγής των χαρακτηριστικών είναι η εξής:

- **Pre-Emphasis:** Ο σκοπός σε αυτό το βήμα είναι η ενίσχυση των υψηλών συχνοτήτων, επειδή η ενέργεια στις υψηλές συχνότητες των φωνηέντων είναι αρκετά μικρότερη σε σχέση με τις χαμηλές. Αυτό έχει αποτέλεσμα να εντοπίζονται πιο εύκολα, και έτσι βελτιώνεται η ακρίβεια του μοντέλου.
- **Framing and Windowing:** Η ομιλία είναι ένα μη στατικό δείγμα, επομένως αν υπολογίσουμε τον μετασχηματισμό Fourier σε όλο το σήμα δεν μπορούμε να εξάγουμε φασματικά χαρακτηριστικά, αφού χάνεται μεγάλη πληροφορία του φάσματος. Συνεπώς, επιλέγουμε ένα μικρό χρονικό παράθυρο, στη διάρκεια του οποίου εφαρμόζουμε τη συνάρτηση παραθύρου. Συνήθως χρησιμοποιούμε το παράθυρο Hamming, το οποίο -όπως φαίνεται στο παρακάτω σχήμα- έχει μια σταθερή ομαλοποίηση στις άκρες του, και έτσι δεν χάνεται πληροφορία στην αρχή και στο τέλος του.



Hamming Window

- **Fourier-Transform:** Έπειτα υλοποιούμε τον Διακριτό Μετασχηματισμό Fourier μέσω του FFT- N σημείων (σε όλα τα παράθυρα πλήθους N) και παίρνουμε τον Short-Time-Fourier-Transform ολόκληρου του σήματος.
- **Mel-filter-bank:** Ο τρόπος που τα αυτιά μας αντιλαμβάνονται τον ήχο δεν είναι γραμμικός, οπότε οι αλλαγές στις χαμηλές συχνότητες είναι πιο αισθητές σε σχέση με τις υψηλές. Η κλίμακα Mel μετατρέπει τις συχνότητες λαμβάνοντας υπόψιν αυτή την ιδιαιτερότητα. Έχει παρατηρηθεί ότι η εφαρμογή της στην εξαγωγή χαρακτηριστικών βελτιώνει την αναγνώριση φωνής σε θορυβώδη σήματα.
- **DCT:** Ο διακριτός μετασχηματισμός συνημιτόνου εκφράζει μία πεπερασμένη ακολουθία δεδομένων ως άθροισμα συνημιτόνων διαφορετικών συχνοτήτων. Σε αντίθεση με τον μετασχηματισμό Fourier, δεν έχει φανταστικές συνιστώσες, με αποτέλεσμα να είναι πιο γρήγορος ο υπολογισμός του. Παρά το γεγονός ότι έχει μόνο πραγματικές τιμές, έχει αποδειχθεί ότι δεν χάνει πληροφορία από το σήμα γιατί είναι γραμμικός. Με την εφαρμογή του στον λογάριθμο της κλίμακας Mel παίρνουμε 13 cepstral συντελεστές deltas και deltas-deltas (πρώτη και δεύτερη παράγωγος των MFCCs αντίστοιχα).



Γλωσσικά Μοντέλα (Language Models)

Τα γλωσσικά μοντέλα καθορίζουν την πιθανότητα μίας λέξης με χρήση ανάλυσης δεδομένων. Για να γίνει η διάκριση ανάμεσα σε λέξεις και προτάσεις που είναι ομόηχες χρησιμοποιεί διάφορες πιθανοτικές προσεγγίσεις και επιλέγει τον συνδυασμό λέξεων που είναι πιο πιθανός σε μια πρόταση. Μερικές από αυτές τις προσεγγίσεις είναι:

- **N-gram**: Δημιουργεί μία πιθανοτική κατανομή για μια ακολουθία από N στοιχεία.

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(w_n|w_{1:n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

- **Unigram και Bigram**: Αποτελούν ειδικές περιπτώσεις του n-gram με $n = 1$ και $n = 2$ και είναι και αυτά που χρησιμοποιήσαμε στην εργασία. Στο bigram η κάθε λέξη εξαρτάται από μία προηγούμενη κατάσταση ενώ στο unigram από την a-priori πιθανότητα εμφάνισης της λέξης.

  Unigram:

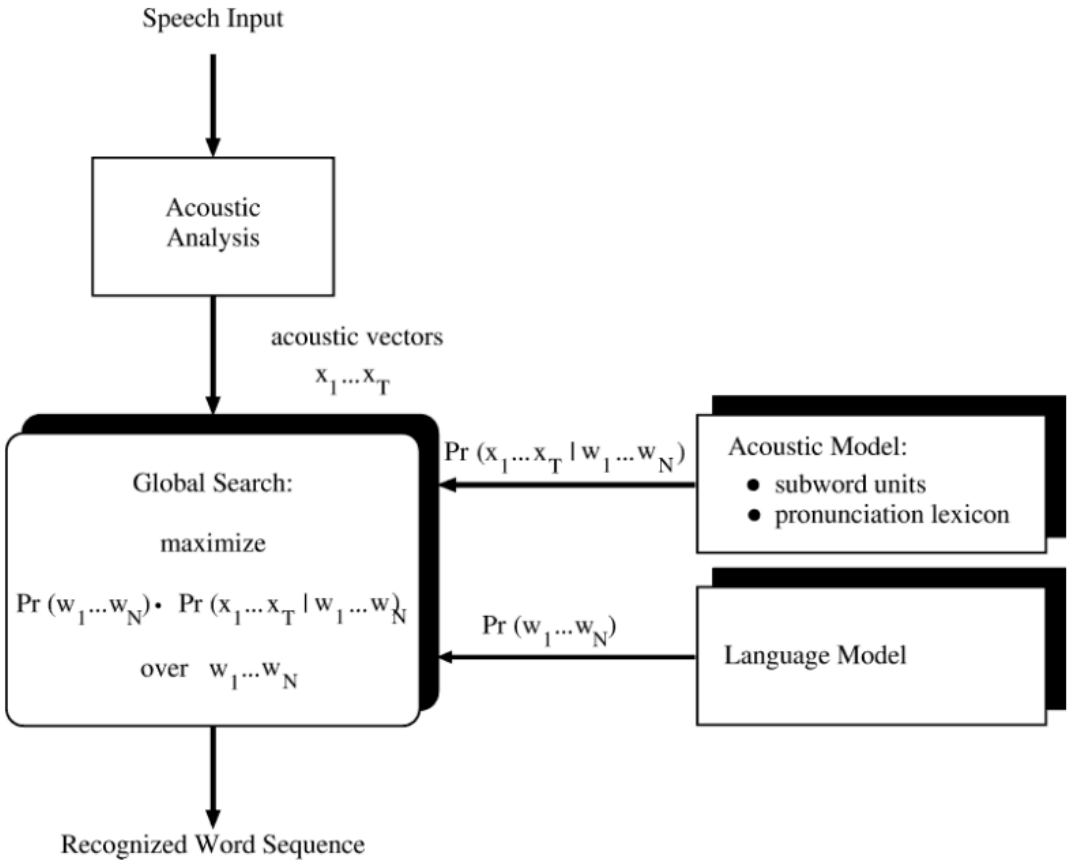
$$P(w_n|w_1, w_2, .., w_n) = P(w_n)$$

  Bigram:

$$P(w_n|w_1, w_2, ..., w_n) = P(w_n|w_{n-1})$$

Φωνητικά Μοντέλα (Acoustic Models)

Τα φωνητικά μοντέλα αποτελούν αναπαραστάσεις του σήματος σε φωνήματα. Με βάση τις Γκαουσιανές μίξεις, παράγονται ορισμένα γνωρίσματα τα οποία αποτελούν τα χαρακτηριστικά του φωνητικού μοντέλου. Στην εκπαίδευση, κάθε πρόταση διασπάται σε επιμέρους λέξεις και κάθε λέξη στα αντίστοιχα φωνήματα, και έτσι δημιουργούνται οι αναπαραστάσεις. Επομένως γίνεται η εκτίμηση των παραμέτρων μεταξύ των καταστάσεων του μοντέλου και δημιουργείται ένας πίνακας μεταβάσεων κάθε φωνήματος και οι πιθανότητες των παρατηρήσεων που αποτελούν τα Hidden Markov Models (HMMs). Τέλος, με την υλοποίηση του αλγορίθμου forward-backward γίνεται η εκτίμηση για το ποια είναι η πιθανή ακολουθία φωνημάτων, ώστε να γίνει η αναγνώριση.



3 Βήματα προπαρασκευής

1.

Εγκαθιστούμε το Kaldi σύμφωνα με τις οδηγίες που μας δίνονται.

2.

Εξοικειωνόμαστε με το Kaldi, με βάση τα tutorials που δίνονται.

3.

Με χρήση του `download.sh`, κατεβάζουμε τα δεδομένα από το Google Drive link που μας δίνεται.

4.

Με χρήση του `3.py` κατασκευάζουμε τον αρχικό σκελετό ως εξής:

- Μέσα στον φάκελο `egs/` δημιουργούμε έναν φάκελο `usc/`.
- Δημιουργούμε τον φάκελο `data` και τους υποφακέλους `data/train, data/dev, data/test`.
- Μέσα σε καθέναν από αυτούς τους 3 φακέλους δημιουργούμε τα αρχεία:
  - `uttids`

  - `utt2spk`
  - `wav.scp`
  - `text`

  Η μορφή και η χρήση τους περιγράφεται αναλυτικά στην εκφώνηση της άσκησης.

- Με χρήση του `lexicon.txt` αντικαθιστούμε τις λέξεις των προτάσεων με τις αντίστοιχες ακολουθίες φωνημάτων. Για τον σκοπό αυτό μετατρέπουμε όλους τους χαρακτήρες σε lower case και αφαιρούμε τους ειδικούς χαρακτήρες, εκτός από τα single quotes. Επίσης προσθέτουμε το φώνημα `sil` (σιωπή) στην αρχή και στο τέλος της κάθε πρότασης.

4 Βήματα κυρίως μέρους

4.1 Προετοιμασία διαδικασίας αναγνώρισης φωνής για τη USC-TIMIT

Με χρήση του `4.1.py` εκτελούμε τα βήματα που ακολουθούν:

1.

Από τη διαδικασία `wsj/` παίρνουμε τα αρχεία `path.sh` και `cmd.sh`. Στο `path.sh`, θέτουμε κατάλληλα τη μεταβλητή `KALDI_ROOT` και στο `cmd.sh` θέτουμε τις μεταβλητές `train_cmd, decode_cmd` και `cuda_cmd` ίσες με `run.pl`.

2.

Δημιουργούμε soft links με ονόματα `steps` και `utils` μέσα στο φάκελο `usc/`, που δείχνουν στους αντίστοιχους φακέλους της `wsj`.

3.

Δημιουργούμε τον φάκελο `local`, και μέσα σ'αυτόν ένα soft link με όνομα `score.sh` που δείχνει στο `steps/score_kaldi.sh`.

4.

Δημιουργούμε τον φάκελο `conf/` και μέσα σε αυτόν αντιγράφουμε το αρχείο `mfcc.conf` που βρίσκεται στο slp-ntua/slp-labs.

5.

Τέλος, δημιουργούμε τους φακέλους:

`data/lang, data/local/dict, data/local/lm_tmp, data/local/nist_lm`

4.2 Προετοιμασία γλωσσικού μοντέλου

Με χρήση του `4.2.py` εκτελούμε τα βήματα που ακολουθούν:

1.

Αρχικά στον φάκελο `data/local/dict` αποθηκεύουμε τα βασικά αρχεία που χρησιμεύουν για τη δημιουργία του γλωσσικού μοντέλου.

- `silence_phones.txt` και `optional_silence.txt`, που περιέχουν μόνο το φώνημα της σιωπής (`sil`).
- Το `nonsilence_phones.txt` το οποίο περιέχει όλα τα υπόλοιπα φωνήματα ταξινομημένα, και ένα σε κάθε γραμμή.
- Το `lexicon.txt`, το οποίο στην δική μας περίπτωση είναι απλά μία 1-1 αντιστοιχία κάθε φωνήματος με τον εαυτό του.
- Τα `lm_{train,dev,test}.text`, που προκύπτουν από τα αντίστοιχα αρχεία `text` (βλ. προπαρασκευή), αν προσθέσουμε σε κάθε πρόταση τα `<s>` και `</s>` στην αρχή και στο τέλος αντίστοιχα.
- Τέλος, δημιουργούμε το (κενό) αρχείο `extra_questions`.

2.

Μέσα στον φάκελο `data/local/lm_tmp` δημιουργούμε την ενδιάμεση μορφή του γλωσσικού μοντέλου. Συγκεκριμένα, θα χρησιμοποιήσουμε την εντολή `build-lm.sh` του πακέτου `IRSTLM`. Για τον σκοπό αυτό δημιουργούμε το αρχείο `build-lm-wrapper.sh`, το οποίο κάνει source το `path.sh` ώστε να γίνει διαθέσιμη η εντολή `build-lm.sh`, και ύστερα την καλεί. Δημιουργούμε τόσο unigram όσο και bigram μοντέλα βάζοντας την παράμετρο `-n` ίση με 1 και 2 αντίστοιχα.

3.

Μέσα στον φάκελο `data/local/nist_lm` αποθηκεύουμε το compiled γλωσσικό μοντέλο σε μορφή ARPA. Για τον σκοπό αυτό δημιουργούμε το αρχείο `compile-lm-wrapper.sh`, το οποίο κάνει source το `path.sh` ώστε να γίνει διαθέσιμη η εντολή `compile-lm.sh`, και ύστερα την καλεί. Προκύπτουν τα αρχεία `lm_phone_ug.arpa.gz` (unigram μοντέλο) και `lm_phone_bg.arpa.gz` (bigram μοντέλο).

4.

Μέσα στον φάκελο `data/lang` δημιουργούμε το `L.fst` (FST του λεξικού της γλώσσας). Για τον σκοπό αυτό δημιουργούμε το αρχείο `prepare_lang-wrapper.sh`, το οποίο κάνει source το `path.sh` ώστε να γίνει διαθέσιμη η εντολή `prepare_lang.sh`, και ύστερα την καλεί.

5.

Χρησιμοποιούμε την εντολή `sort` για να ταξινομήσουμε τα αρχεία `wav.scp, text, utt2spk` στους φακέλους `data/{train,dev,test}`.

6.

Εκτελούμε το `utils/utt2spk_to_spk2utt.pl` και έτσι δημιουργούμε το αρχείο `spk2utt`.

7.

Τέλος, δημιουργούμε το `G.fst` (FST της γραμματικής), με βάση τη διαδικασία `timit` (αρχείο `local/timit_format_data.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 είναι η μέτρηση για το πόσο καλά μια πιθανοτική κατανομή μπορεί να προβλέψει ένα δείγμα και δείχνει το πόσο περιπλέκεται το μοντέλο μας (όσο μικρότερο τόσο το καλύτερο). Έχουμε:

	Unigram	**Bigram**
Dev	56.23	27.02
Test	55.15	26.39

Παρατηρούμε ότι το bigram αποδίδει πολύ καλύτερα από το unigram (το perplexity πέφτει σχεδόν στο μισό). Αυτό είναι λογικό, γιατί το unigram δεν χρησιμοποιεί την προηγούμενη κατάσταση για να υπολογίσει την πιθανότητα του κάθε φωνηέντου αλλά μόνο την a-priori πιθανότητα εμφάνισης του.

4.3 Εξαγωγή ακουστικών χαρακτηριστικών

Με χρήση του `4.3.py` - το οποίο καλεί τις εντολές `make_fcc.sh` και `compute_cmvn_stats.sh` - εξάγουμε τα MFCCs και για τα 3 σετ.

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

Το CMVN είναι μία υπολογιστικά αποδοτική τεχνική κανονικοποίησης του σήματος που χρησιμοποιείται στην αναγνώριση φωνής. Ο στόχος της εύρωστης εξαγωγής στοιχείων είναι η ελαχιστοποίηση της παραμόρφωσης που δημιουργείται στα σήματα φωνής λόγω του θορύβου, ώστε να γίνεται η βέλτιστη εξαγωγή χαρακτηριστικών. Αυτό επιτυγχάνεται με την κανονικοποίηση της μέσης τιμής και της διασποράς στις τιμές 0 και 1 αντίστοιχα. Η κανονικοποίηση αυτή επιτυγχάνει την ομαλοποίηση της διασποράς, επειδή καθώς σχηματίζονται οι συντελεστές, οι μικρές αλλαγές και ο θόρυβος μπορεί να προκαλέσουν μεγάλο variance με αποτέλεσμα λανθασμένη αναγνώριση του φωνήματος.

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

Με βάση το αρχείο `data/train/utt2num_frames` έχουμε:

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

Η διάσταση των χαρακτηριστικών προκύπτει από τη σχέση:

$$\text{Διάσταση} = 13 \times \left(\frac{\text{Διάρκεια}}{\text{Frame shift}} \right)$$

Η διάρκεια βρίσκεται στο αρχείο `data/train/utt2dur`, το frame shift στο `data/train/frame_shift`, ενώ 13 είναι το πλήθος των συντελεστών του Mel-Frequency Cepstrum που κρατάμε κάθε φορά.

4.4 Εκπαίδευση ακουστικών μοντέλων και αποκωδικοποίηση προτάσεων

Με χρήση του `4.4.py` εκτελούμε τα βήματα που ακολουθούν:

1.

Με την εντολή `steps/train_mono.sh` εκπαιδεύουμε ένα monophone GMM-HMM ακουστικό μοντέλο πάνω στα `train` δεδομένα.

2.

Δημιουργούμε το γράφο HCLG σύμφωνα με την γραμματική G του προηγούμενου βήματος. Χρησιμοποιούμε το `utils/mkgraph.sh` (τόσο για unigrams όσο και για bigrams).

3.

Χρησιμοποιούμε τον αλγόριθμο Viterbi για να αποκωδικοποιήσουμε τις προτάσεις τόσο των validation όσο και των test δεδομένων. Χρησιμοποιούμε το `steps/decode.sh`.

4.

Χρησιμοποιούμε το `local/score.sh` και τα αποτελέσματα που βρίσκονται στα αρχεία:

```
exp/mono/decode_dev_ug/scoring_kaldi/best_wer
exp/mono/decode_dev_bg/scoring_kaldi/best_wer
exp/mono/decode_test_ug/scoring_kaldi/best_wer
exp/mono/decode_test_bg/scoring_kaldi/best_wer
```

Η μετρική PER (Phone Error Rate) είναι:

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

Με βάση τα αποτελέσματα στα παραπάνω αρχεία έχουμε:

PER for monophone model

	Unigram	**Bigram**
Dev	52.66%	45.98%
Test	51.59%	45.01%

5.

Κάνουμε alignment των φωνημάτων χρησιμοποιώντας το monophone μοντέλο (`steps/align_si.sh`), και με βάση αυτά τα alignments εκπαιδεύουμε ένα triphone μοντέλο (`steps/train_deltas.sh`). Δημιουργούμε τον γράφο HCLG και πραγματοποιούμε αποκωδικοποίηση ξανά. Αυτή τη φορά τα αποτελέσματα βρίσκονται στα αρχεία:

```
exp/tri1/decode_dev_ug/scoring_kaldi/best_wer
exp/tri1/decode_dev_bg/scoring_kaldi/best_wer
exp/tri1/decode_test_ug/scoring_kaldi/best_wer
exp/tri1/decode_test_bg/scoring_kaldi/best_wer
```

Έχουμε:

PER for triphone model

	Unigram	**Bigram**
Dev	40.00%	36.66%
Test	39.07%	35.03%

Υπάρχει αρκετά σημαντική βελτίωση σε σχέση με το monophone μοντέλο. Επίσης εδώ υπάρχει μικρότερη διαφορά μεταξύ unigram και bigram μοντέλου.

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 ποιο τρόπο γίνεται η εκπαίδευση ενός τέτοιου μοντέλου; Περιγράψτε τη διαδικασία εκπαίδευσης ενός μονοφωνικού μοντέλου.

Τα GMMs είναι ένας τρόπος να προσδιοριστούν και να ομαδοποιηθούν οι καταστάσεις του αυτόματου σε κατηγορίες με βάση την πιθανότητα ενός φωνήματος να βρίσκεται σε μία λέξη. Αν το μοντέλο μας χρησιμοποιούσε μόνο Gaussian-Mixture-Model, αυτό θα σήμαινε ότι θα κατηγοριοποιούσε κάθε φώνημα χωρίς να λαμβάνει υπόψιν προηγούμενες καταστάσεις, οπότε δεν θα μπορούσε να δει αν ταιριάζει κάποια πρόβλεψη στην εκάστοτε λέξη η πρόταση, προκειμένου να ελαχιστοποιήσει την πιθανότητα λάθους. Γι' αυτό χρησιμοποιεί και τα HMMs, στατιστικά μοντέλα τα οποία υποθέτουν κρυφές καταστάσεις (τοποθέτηση γλώσσας, ποια συντακτική θέση έχει η λέξη στην οποία βρίσκεται το φώνημα, πιθανότητες μετάβασης από μία ακολουθία N φωνημάτων σε ένα άλλο). Ο τρόπος που γίνεται η εκπαίδευση είναι με μια ειδική κατηγορία του Expectation-Maximization, τον forward-backward, ο οποίος μας επιτρέπει να υπολογίσουμε την πλήρη υπό συνθήκη πιθανοφάνεια μιας σειράς καταστάσεων, δεδομένης μιας ακουλουθίας παρατηρήσεων. Αρχικά κάνει μια εκτίμηση για την μέση τιμή και διασπορά των δεδομένων και έπειτα τις ξανά υπολογίζει επαναληπτικά με βάση τις προηγούμενες τιμές και τις καταστάσεις μετάβασης, οπότε μεγαλώνει η πιθανοφάνεια και συγκλίνει σε μία τιμή. Έτσι, εκπαιδεύουμε ταυτόχρονα τόσο τα transition, όσο και τα emission probabilities του 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;

Το Μπεϋζιανό μοντέλο για την εύρεση πιθανότητας ενός φωνήματος είναι με βάση την a posteriori για κάθε πιθανό φώνημα δεδομένης μια ακουλουθίας λέξεων και φωνημάτων (χαρακτηριστικών).

$$P(W|X) = \frac{P(X|W) \cdot P(W)}{P(X)}$$

όπου:

- $P(W|X)$ η a-posteriori πιθανότητα,
- $P(X|W)$ η πιθανότητα ανίχνευσης χαρακτηριστικών δεδομένου του φωνήματος W_i επομένως το φωνητικό μοντέλο(acoustic model)
- $P(W)$ η a-priori που είναι η πιθανότητα εμφάνισης των χαρακτηριστικών(language model)

Έπειτα από τα i πιθανά φωνήματα που έχουν προκύψει, βρίσκουμε την μέγιστη πιθανοφάνεια και προκύπτει το εκτιμώμενο φώνημα.

$$W = \text{argmax}_{0 \leq i \leq n} \{P(W_i|X)\} = \text{argmax}_{0 \leq i \leq n} \left\{ \frac{P(X|W_i) \cdot P(W_i)}{P(X)} \right\}$$

Αφού θέλουμε να δούμε για ποιο W_i μεγιστοποιείται η παραπάνω συνάρτηση, μπορούμε να απαλείψουμε τον όρο $P(X)$, γιατί παραμένει ίδιος για όλα τα i. Επομένως, έχουμε το εξής:

$$W = \text{argmax}_{0 \leq i \leq n} \{P(X|W_i) \cdot P(W_i)\}$$

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

Ο γράφος HCLG είναι αναπτυγμένο γράφημα αποκωδικοποίησης που αναπαριστά τον αποδοχέα γραμματικής, το λεξικό φωνημάτων, τα HMMs και τα context-dependency. Το output είναι ένα fst με word-ids και ως input δέχεται word-ids.

- `H.fst` είναι το FST που περιέχει πληροφορίες για τα HMMs.
- `C.fst` είναι το FST που κάνει τα φωνήματα σε contex-dependency.
- `L.fst` είναι λεξικό που παίρνει ως είσοδο μία λέξη και μας δίνει σαν έξοδο τα επιμέρους φωνήματα.
- `G.fst` είναι ο αποδοχέας της γραμματικής.