ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάθημα Προπτυχιακών Σπουδών:
Επεξεργασία Σημάτων Φωνής και Ήχου
Ακαδημαϊκό έτος: 2023 - 2024
Εξάμηνο: 8ο
Τελική Εργασία

Φοιτητής: Κοξάνογλου Θεόδωρος, Π20094
Επιβλέπων Καθηγητής: Πικράκης Άγγελος
Έκδοση Εργασίας: Πειραιάς, Ιούλιος 2024

Περιεχόμενα

Εκφώνηση

Εργασία Εαρινού Εξαμήνου 2023-2024

- **Ημερομηνία παράδοσης:** Ημερομηνία εξέτασης του μαθήματος, ώρα 23:59μμ. Η εργασία συμμετέχει με βάρος 40% στον τελικό βαθμό.
- Η εργασία είναι ατομική και παραδίδεται μέσω της πλατφόρμας e-class. Αποδεκτές γλώσσες είναι οι Matlab και Python. Στα παραδοτέα συμπεριλαμβάνονται:
    1. η τεκμηρίωση της εργασίας σε αρχείο pdf, στην πρώτη σελίδα της οποίας αναγράφεται το ονοματεπώνυμο του φοιτητή/φοιτήτριας και ο ΑΜ του/της. Θα μηδενιστούν οι εργασίες που δεν περιέχουν τεκμηρίωση ή στοιχεία φοιτητή/φοιτήτριας.
    2. τα αρχεία source code σε ένα συμπιεσμένο αρχείο με όνομα source2023.zip (ή .rar ή άλλη σχετική κατάληξη).
    3. οποιαδήποτε άλλα συνοδευτικά αρχεία κρίνετε απαραίτητα σε ένα συμπιεσμένο αρχείο με το όνομα auxiliary2023.zip (ή .rar ή άλλη σχετική κατάληξη).

- Η εργασία θα είναι η ίδια και τον Σεπτέμβριο.
- Η αντιγραφή ή η χρήση generative bots οδηγεί σε μηδενισμό.

Πρώτη άσκηση:

Καλείστε να υλοποιήσετε ένα σύστημα που προχωρά στην κατάτμηση μιας πρότασης σε λέξεις, χρησιμοποιώντας υποχρεωτικά έναν ταξινομητή background vs foreground της επιλογής σας. Δηλαδή, δοθείσης μιας ηχογράφησης ενός ομιλητή, το σύστημα επιστρέφει τα χρονικά όρια των λέξεων που ειπώθηκαν (σε δευτερόλεπτα). Επίσης, παρέχετε συνοδευτικό πρόγραμμα το οποίο αναπαράγει τις λέξεις που εντοπίστηκαν. Το πλήθος των λέξεων στην πρόταση δεν είναι εκ των προτέρων γνωστό, αλλά μπορείτε να υποθέσετε ότι υπάρχει μικρό διάστημα απουσίας ομιλίας μεταξύ λέξεων.

Θα πρέπει να υλοποιήσετε και να συγκρίνετε τις επιδόσεις των παρακάτω ταξινομητών: Least Squares, SVM, RNN και MLP τριών επιπέδων (προσδιορίστε τον αριθμό νευρώνων ανά επίπεδο). Η σύγκριση θα γίνει όπως προβλέπεται σε δυαδικά συστήματα ταξινόμησης.

Δεύτερη Άσκηση:

Από τις λέξεις που προκύπτουν, υπολογίστε τη μέση θεμελιώδη συχνότητα του ομιλητή.

- Πρέπει να εξηγήσετε ποια δεδομένα χρησιμοποιήσατε κατά τον έλεγχο και την εκπαίδευση του συστήματος. Αν είναι δικά σας, πώς τα δημιουργήσατε και αν είναι open source, πώς αξιοποιούνται.
- Προσπαθήστε να μην εξαρτάται το σύστημα από τα χαρακτηριστικά της φωνής του ομιλητή, αλλά να είναι όσο το δυνατόν ανεξάρτητο ομιλητή.

Προσοχή!!!: Δεν μπορείτε να χρησιμοποιήσετε **συνελικτικά** νευρωνικά δίκτυα. **Δεν** είναι αποδεκτή η χρήση έτοιμων web services ή APIs για speech recognition. **Δεν** μπορείτε να χρησιμοποιήσετε **transfer learning** από ήδη εκπαιδευμένα δίκτυα. Οι αντίστοιχες λύσεις μηδενίζονται.

Καλή επιτυχία!

Ανάλυση Εργασίας

Πρώτη άσκηση

Για υλοποιήσω ένα σύστημα που προχωρά στην κατάτμηση μιας πρότασης σε λέξεις, θα χρειαστεί πρώτα να το εκπαιδεύσω.

Εύρεση κατάλληλων ηχητικών για εκπαίδευση

Μετά από αρκετή αναζήτηση κατέληξα στη χρήση των δεδομένων που είναι για ανάπτυξη εφαρμογών της VOiCES (Voices Obscured in Complex Environmental Settings). Αυτό το σύνολο δεδομένων/κειμένων κυκλοφορεί ως open source, με την άδεια Creative Commons BY 4.0, ελεύθερο για εμπορική και ακαδημαϊκή χρήση.

Από τα τέσσερα συμπιεσμένα αρχεία που έχουν στη βάση τους:

- VOiCES_release.tar.gz (417.5 GiB)
- **VOiCES_devkit.tar.gz (27.5 GiB)**
- VOiCES_competition.tar.gz (19.5 GiB)
- recording_data.tar.gz (56 MiB)

Χρησιμοποίησα το δεύτερο το οποίο περιλαμβάνει:

- **Για εκπαίδευση:** 12.800 ηχητικά παραδείγματα
- **Για δοκιμές:** 6.400 ηχητικά παραδείγματα

Το VOiCES_devkit αποτελεί ένα μικρότερο δείγμα του πλήρους συνόλου δεδομένων VOiCES_release. Περιλαμβάνει όλους τους ομιλητές του πλήρους συνόλου, αλλά για κάθε ομιλητή επιλέχθηκαν τυχαία δύο ηχογραφήσεις από το **Librispeech**.

Από το συγκεκριμένο dataset χρησιμοποιήθηκαν τα ηχητικά που ήταν από το **rm1**. Η επιλογή των ηχητικών και η ταξινόμησή τους στους κατάλληλους φακέλους έγινε με την χρήση της **create_datasets()** στο αρχείο **load_dataset.py**.

Το dataset που έχω για την εκπαίδευση αποτελείται από δύο φακέλους: **background_sound**, **foreground_sound**. Στο **background_sound** υπάρχουν ηχητικά που έχουν μόνο θόρυβο υποβάθρου τύπου: οχλαγωγία, ήχοι από τηλεόραση, μουσική και ήχοι δωματίου. Στο **foreground_sound** υπάρχουν ηχητικά που ακούγονται ομιλητές με και χωρίς ήχους υποβάθρου. Τα ηχητικά αυτά προέρχονται από τις ηχογραφήσεις του Librispeech που δεν έχουν μεγάλες παύσεις μεταξύ τους, επομένως δεν θα χρειαστούν επεξεργασία πριν την εκπαίδευση των συστημάτων.

Για την εύκολη δοκιμή των μοντέλων δημιούργησα ένα τρίτο dataset **test**, το οποίο θα περιλαμβάνει μόνο ηχητικά αρχεία ομιλίας που είναι για **testing** από το **rm1** του dataset.

Το dataset που δημιούργησα περιλαμβάνει πολλούς ομιλητές ώστε τα συστήματα που θα δημιουργήσω να είναι **ανεξάρτητα του ομιλητή**.

Εξαγωγή χαρακτηριστικών των ηχητικών εκπαίδευσης

Η εξαγωγή χαρακτηριστικών (**feature extraction**) είναι η διαδικασία μετατροπής των ηχητικών σημάτων σε μια αντιπροσωπευτική μορφή, κατάλληλη για την εκπαίδευση και ανάλυση αλγορίθμων μηχανικής μάθησης.

Το feature extraction το εφαρμόζω με χρήση της συνάρτησης **extract_features()** που βρίσκεται στο αρχείο **feature_extraction.py**.

Διαδικασία εξαγωγής χαρακτηριστικών

1. Φόρτωση του dataset: φορτώνω τα dataset που δημιούργησα στο προηγούμενο βήμα:
   - background_sound
   - foreground_sound

2. Ορισμός καθολικών παραμέτρων: που θα εφαρμοστούν σε όλα τα ηχητικά κατά την επεξεργασία τους.
   - **`N_MFCC = 20:`** Ο αριθμός των Mel-frequency cepstral coefficients (**MFCCs**) που θα εξαχθούν.
   - **`N_MELS = 96:`** Ο αριθμός των ζωνών **Mel** που θα δημιουργηθούν.
   - **`N_FFT = 512:`** Το μέγεθος του παραθύρου **FFT** (Fast Fourier Transform).
   - **`HOP_LENGTH = 256:`** Ο αριθμός των δειγμάτων ανάμεσα σε διαδοχικά frames.

3. Φόρτωση και εξαγωγή χαρακτηριστικών: Για κάθε αρχείο ήχου, φορτώνουμε το σήμα και εξάγουμε τα χαρακτηριστικά MFCC και το mel spectrogram με τεχνική κινούμενου παραθύρου - συνάρτηση **load_and_extract_features()** στο αρχείο **feature_extraction.py**.
   - Εφαρμόζουμε **Hamming windowing** στο σήμα για να μειώσουμε την παραμόρφωση στα άκρα του παραθύρου.
   - Υπολογίζουμε τα **MFCC** χαρακτηριστικά για κάθε frame.
   - Υπολογίζουμε το **mel-spectrogram** και το μετατρέπουμε σε κλίμακα **dB** με τη συνάρτηση **librosa.power_to_db()**.

   Κάθε mel-spectrogram θα πρέπει να έχει και ένα συγκεκριμένο label ώστε να μπορούν τα συστήματα ταξινόμησης που θα δημιουργήσω να αναγνωρίσουν τη διαφορά μεταξύ του background και του foreground sound. Για τον λόγο αυτό, χώρισα το dataset με τέτοιο τρόπο ώστε να μπορέσω εύκολα να ορίσω τα labels στα ηχητικά που προορίζονται για εκπαίδευση. Τα mel-spectrograms των ηχητικών που βρίσκονται στο **background_sound** θα έχουν **label 0** και τα mel-spectrograms των ηχητικών που βρίσκονται στο **foreground_sound** θα έχουν **label 1**.

4. Ανακάτεμα των δεδομένων: Έχοντας συγκεντρώσει όλα τα mel-spectrograms με τα αντίστοιχα labels ανακατεύονται τυχαία για να αποφευχθεί η μεροληψία κατά την εκπαίδευση του μοντέλου ταξινόμησης. Έχω βάλει boolean όρισμα στη συνάρτηση **extract_features()** για να δοκιμάσω την αποδοτικότητα των ταξινομητών όταν ανακατεύω τα δεδομένα.

5. Αποθήκευση των δεδομένων ταξινόμησης: αποθηκεύω τις λίστες σε ένα συμπιεσμένο αρχείο numpy **features.npz** ώστε να χρησιμοποιηθεί μελλοντικά από τα συστήματα ταξινόμησης.

Εκπαίδευση δυαδικών συστημάτων ταξινόμησης

Για τον κάθε ένα ταξινομητή έχω δημιουργήσει ένα διαφορετικό python αρχείο που περιλαμβάνει τρεις συναρτήσεις:

- **train:** για την εκπαίδευση και την αποθήκευση του εκπαιδευμένου μοντέλου
- **load_model:** για την φορτωση του εκπαιδευμένου μοντέλου
- **predict:** για την εξαγωγή των προβλέψεων του εκπαιδευμένου μοντέλου

Σε ορισμένα αρχεία ταξινομητών έχω προσθέσει επιπλέον συναρτήσεις τις οποίες θα τις αναλύσω παρακάτω.

Τα εκπαιδευμένα μοντέλα των ταξινομητών αποθηκεύονται στο φάκελο: **auxiliary2024/output/classifiers.**

Εκπαίδευση SVM

Το αρχείο του κώδικα βρίσκεται στον φάκελο **classifiers** και ονομάζεται **svm.py**.

1. Προετοιμασία Δεδομένων
    - Φορτώνω τα χαρακτηριστικά και τα αντίστοιχα label τους από το αρχείο **features.npz** χρησιμοποιώντας τη συνάρτηση **load_features()** από το **feature_extraction.py**.
    - Επειδή το SVM δεν έχει ενσωματωμένο τρόπο να ελέγχει την ακρίβεια του, αυτή υπολογίζεται χειροκίνητα.
    - Διαχωρίζω τα δεδομένα σε σύνολα εκπαίδευσης (90%) και δοκιμών (10%) για να αξιολογήσω την απόδοση του μοντέλου.
2. Εκπαίδευση του μοντέλου SVM
    - Αρχικοποιώ έναν SVM ταξινομητή με έναν γραμμικό πυρήνα από την βιβλιοθήκη sklearn.svm το **LinearSVC** και (όχι το SVC(kernel='linear') το οποίο προσδιορίζεται για μικρότερο όγκο δεδομένων προς εκπαίδευση).
    - Ο ταξινομητής εκπαιδεύεται με το σύνολο που προορίζεται για εκπαίδευση.
3. Αξιολόγηση μοντέλου:
    - Τα χαρακτηριστικά που διατηρήθηκαν για έλεγχο χρησιμοποιούνται από τον ταξινομητή για να προβλέψει τα labels τους.
    - Η ακρίβεια του μοντέλου υπολογίζεται συγκρίνοντας τα labels που προέβλεψε με τα αρχικά labels.
4. Αποθήκευση μοντέλου:
    - Αποθηκεύω το μοντέλο που μόλις εκπαίδευσα σε ένα αρχείο **svm_model.pkl** χρησιμοποιώντας την βιβλιοθήκη **joblib**.

Εκπαίδευση MLP

Το αρχείο του είναι το **mlp.py** μέσα στον φάκελο classifiers.

1. Προετοιμασία Δεδομένων
    - Φορτώνω τα χαρακτηριστικά και τις αντίστοιχες ετικέτες τους από το αρχείο **features.npz** χρησιμοποιώντας τη συνάρτηση **load_features()** από το **feature_extraction.py**.
2. Εκπαίδευση του μοντέλου SVM
    - Αρχικοποιώ έναν MLP ταξινομητή με τρία κρυφά επίπεδα με 256, 128 και 64 **νευρώνες** αντίστοιχα χρησιμοποιώντας τη βιβλιοθήκη **sklearn.neural_network**.
    - Ρυθμίζω τον ταξινομητή να εκπαιδεύεται με μέγιστο αριθμό **300** επαναλήψεων και να χρησιμοποιεί τυχαίο αριθμό **seed=0** για την αναπαραγωγιμότητα των αποτελεσμάτων. Επίσης, ενεργοποιώ την επιλογή **early_stopping** για να σταματά η εκπαίδευση όταν δεν βελτιώνεται η απόδοση.

- Ο ταξινομητής εκπαιδεύεται με το σύνολο που προορίζεται για εκπαίδευση.

3. Αξιολόγηση μοντέλου:
   - Η απόδοση του μοντέλου αξιολογείται αυτόματα κατά την εκπαίδευση μέσω των μετρικών που παρέχονται από τον MLP ταξινομητή, όπως η ακρίβεια (**accuracy**).
4. Αποθήκευση μοντέλου:
   - Αποθηκεύω το εκπαιδευμένο μοντέλο σε ένα αρχείο **mlp_model.pkl** χρησιμοποιώντας τη βιβλιοθήκη **joblib**.

Εκπαίδευση RNN

Το αρχείο του κώδικα βρίσκεται στον φάκελο **classifiers** και ονομάζεται **rnn.py**.

1. Προετοιμασία Δεδομένων
   - Φορτώνω τα χαρακτηριστικά και τα αντίστοιχα label τους από το αρχείο **features.npz** χρησιμοποιώντας τη συνάρτηση **load_features()** από το **feature_extraction.py**.
   - Τα δεδομένα αυτά διαμορφώνονται σε κατάλληλη μορφή για το μοντέλο RNN:
     - Εύρεση Κατάλληλου Διαιρέτη: Η συνάρτηση **get_divisor** βρίσκει έναν κατάλληλο διαιρέτη του πλήθους των χρονικών βημάτων (timesteps) στα χαρακτηριστικά. Αυτός ο διαιρέτης χρησιμοποιείται για να χωρίσει τις ακολουθίες των χαρακτηριστικών σε ίσα μέρη (γιατί δεν θα μπορούσα να τροφοδοτήσω το μοντέλο με άνισα “αρχεία”), αντιπροσωπεύοντας έναν πλασματικό αριθμό ηχητικών αρχείων που με εξυπηρετεί.
     - Αναδιαμόρφωση Χαρακτηριστικών και Ετικετών: Τα χαρακτηριστικά και οι ετικέτες αναδιαμορφώνονται σε έναν πίνακα τριών διαστάσεων, όπου:
       1. Η πρώτη διάσταση αντιστοιχεί στα ηχητικά αρχεία,
       2. Η δεύτερη στα χρονικά βήματα κάθε αρχείου και
       3. Η τρίτη στα χαρακτηριστικά mel-spectrogram.

     Η παραπάνω προεπεξεργασία έγινε ώστε τα δεδομένα να έχουν τη σωστή μορφή και μέγεθος για την εκπαίδευση του RNN.

2. Εκπαίδευση του μοντέλου RNN:
   - Αρχικοποιώ ένα RNN μοντέλο χρησιμοποιώντας τη βιβλιοθήκη **tensorflow.keras**.
   - Προσθέτω τα απαραίτητα επίπεδα στο μοντέλο:
     - Input Layer: Δέχεται τις ακολουθίες των χαρακτηριστικών mel-spectrogram.
     - SimpleRNN: Εφαρμόζει έναν απλό επαναληπτικό νευρώνα με συνάρτηση ενεργοποίησης σιγμοειδή **(sigmoid)**, ο οποίος επεξεργάζεται τις ακολουθίες διατηρώντας την πλήρη ακολουθία των κρυφών καταστάσεων **(hidden states)** για κάθε χρονικό βήμα ώστε να λαμβάνει υπόψη τη χρονική εξάρτηση των δεδομένων (**return_sequences=True**).
     - Dense: Παράγει την τελική πρόβλεψη για κάθε χρονικό βήμα (background ή foreground), χρησιμοποιώντας μια συνάρτηση ενεργοποίησης σιγμοειδή **(sigmoid)** για την **έξοδο πιθανοτήτων**.
   - Το μοντέλο μεταγλωττίζεται με τον βελτιστοποιητή Adam (**Adaptive Moment Estimation** - προσαρμόζει αυτόματα το learning rate), την **binary-crossentropy** ως συνάρτηση απώλειας και την **accuracy** ως μετρική αξιολόγησης.
   - Το μοντέλο εκπαιδεύεται για έναν καθορισμένο αριθμό **epochs** (10) με το σύνολο που προορίζεται για εκπαίδευση.
3. Αξιολόγηση μοντέλου:
   - Χρησιμοποιώ ένα ποσοστό (10%) των δεδομένων για επικύρωση κατά τη διάρκεια της εκπαίδευσης. Η επικύρωση βοηθά στην παρακολούθηση της απόδοσης του μοντέλου και γίνεται αυτόματα από το ίδιο το μοντέλο.

4. Αποθήκευση μοντέλου:
    - Αποθηκεύω το μοντέλο που μόλις εκπαίδευσα σε ένα αρχείο **rnn_model.keras**.

Εκπαίδευση Least Squares

Το αρχείο του κώδικα βρίσκεται στον φάκελο **classifiers** και ονομάζεται **least_squares.py**.

1. Προετοιμασία Δεδομένων
    - Φορτώνω τα χαρακτηριστικά και τα αντίστοιχα label τους από το αρχείο **features.npz** χρησιμοποιώντας τη συνάρτηση **load_features()** από το **feature_extraction.py**.
    - Τα χαρακτηριστικά με τα αντίστοιχα labels τους μετατρέπονται σε μορφή συμβατή με τον **Tensorflow** μέσω της συνάρτησης **preprocess_data()** και προστίθεται ένα **bias term** στα χαρακτηριστικά μέσω της **add_bias_term()**.
2. Εκπαίδευση του μοντέλου Least Squares
    - Χρησιμοποιώ τη συνάρτηση **tf.linalg.lstsq()** της **TensorFlow** για να λύσω το σύστημα εξισώσεων και να υπολογίσω τα βάρη του μοντέλου. Εφαρμόζω το μοντέλο στα δεδομένα επικύρωσης για να λάβω τις προβλέψεις, οι οποίες μετατρέπονται σε **0** και **1**.
3. Αξιολόγηση μοντέλου:
    - Η απόδοση του μοντέλου αξιολογείται μέσω της ακρίβειας (**accuracy**) που υπολογίζεται συγκρίνοντας τις προβλέψεις με τις πραγματικές ετικέτες του συνόλου επικύρωσης.
4. Αποθήκευση μοντέλου:
    - Αποθηκεύω τα εκπαιδευμένα βάρη του μοντέλου σε ένα αρχείο **ls_model.pkl** χρησιμοποιώντας τη βιβλιοθήκη joblib

Χρήση μοντέλων δυαδικών ταξινομητών

1. Τοποθετώ στο πρόγραμμα **main.py** (χειροκίνητα στον κώδικα) ή στο **main_replacement.py** (στη γραμμή εντολών μέσα από την αντίστοιχη επιλογή του μενού) τη διεύθυνση του ηχητικού αρχείου που θα ήθελα να αναλύσω.
2. Μέσω της συνάρτησης **transcribe_audio()** του αρχείου **speech_to_text.py** εξάγω από το αρχείο τα χρονικά διαστήματα στα οποία υπάρχει ομιλία σε **seconds** και τα αποθηκεύουμε στην μεταβλητή **intervals_original** - ground truth - ως λίστα πλειάδων (**start_time, end_time**).
3. Κάνω εξαγωγή χαρακτηριστικών του αρχείο μέσω της συνάρτησης **load_and_extract_features()** από το αρχείο **feature_extraction.py**.
4. Έχοντας τα features του ηχητικού αρχείου, τα περνάω και από τους τέσσερις ταξινομητές που δημιούργησα - μέσω της αντίστοιχης συνάρτησης **predict() για κάθε ταξινομητή**.
5. Οι ταξινομητές με την σειρά τους κάνουν ο καθένας την πρόβλεψη τους.
6. Περνάω τα αποτελέσματα τους από φίλτρο μεσαίας τιμής (**median filter**) **L.**
7. Μετατρέπω τα διαστήματα ομιλίας σε seconds και τα αποθηκεύω και αυτά ως λίστα πλειάδων (**start_time, end_time**) στη μεταβλητή **{ονομασία_ταξινομητή}_predictions_median**,
8. Τα αναπαράγω μέσω της συνάρτησης **show_predictions()** χρησιμοποιώντας την βιβλιοθήκη **pydub.playback**
9. Μετά το τέλος της αναπαραγωγής συγκρίνω τα διαστήματα που βρήκε ο κάθε ταξινομητής με τα σωστά διαστήματα μέσω της συνάρτησης **calculate_accuracy()** από το αρχείο **speech_to_text.py**.
10. Εκτυπώνω τα αποτελέσματα **στην κονσόλα** και **σε ένα txt αρχείο**.

Τα αποτελέσματα εκτυπώνονται στον φάκελο: **auxiliary2024/output/results**

Συμπεράσματα

Δοκίμασα όλα τα δυαδικά μοντέλα που δημιούργησα με 8 διαφορετικά ηχητικά αρχεία του από το φάκελο tests. Παρατήρησα ότι σε όλα τα μοντέλα δυαδικής ταξινόμησης που δημιούργησα έχουν μέτρια ακρίβεια ως προς τον εντοπισμό των λέξεων σε ένα ηχητικό αρχείο. Τα αποτελέσματα έδειξαν ότι και τα τέσσερα μοντέλα ταξινομητών εμφάνιζαν αδυναμία στην αναγνώριση του background sound, ενώ η συνολική τους ακρίβεια ήταν μέτρια (30%-40%).

Το μοντέλο RNN ήταν το μοντέλο που εμφάνιζε την μεγαλύτερη ακρίβεια στις προβλέψεις background και foreground sounds. Το δεύτερο μοντέλο με την μεγαλύτερη ακρίβεια ήταν το MLP, ενώ χαμηλή ακρίβεια εμφάνισαν τα δύο τελευταία μοντέλα SVM και Least Squares, με τον Least Squares να εμφανίζει την χειρότερη απόδοση.

Θα μπορούσαν να βελτιωθούν οι επιδόσεις των ταξινομητών κάνοντας προσθήκες ή τροποποιήσεις όπως:

1. Προσθήκη περισσότερων ηχητικών στο training dataset. Λόγω της υπολογιστικής ισχύς που είχε ο υπολογιστής μου δεν μπόρεσα να χρησιμοποιήσω όλο το training dataset του VOiCES. Σε ένα άλλο μηχάνημα μεγαλύτερης υπολογιστικής ισχύς θα μπορούσε να χρησιμοποιήσει ολόκληρο το dataset και να βελτιώσει την πρόβλεψη των ταξινομητών για τον εντοπισμό των λέξεων.

2. Εφαρμογή κάποιου επιπλέον φίλτρου για την αφαίρεση του θορύβου από το ηχητικό αρχείο πριν το εισάγουμε στους ταξινομητές.

3. Αλλαγή των παραμέτρων στα μοντέλα εκπαίδευσης των ταξινομητών. Για παράδειγμα τα hidden layers νευρώνων του MLP μοντέλο από 256, 128, 64 θα μπορούσαν να γίνουν 512, 256, 128 ή 96, 48, 24 και συγκριθούν τα αποτελέσματα.

Οδηγίες Εκτέλεσης Προγράμματος

Το πρόγραμμα δημιουργήθηκε μέσω του **Pycharm IDE 2024.1.4 (Professional Edition) Linux Mint**.

Προαπαιτούμενα

Βεβαιωθείτε ότι έχετε εγκατεστημένο το Anaconda ή το Miniconda στο σύστημά σας. Μπορείτε να το κατεβάσετε από εδώ.

Οδηγίες Εγκατάστασης

1. Κλωνοποιήστε το git repository και μεταβείτε στο root directory (**δεν χρειάζεται αυτό το βήμα**):

   git clone https://github.com/thkox/speech-and-audio-processing
   cd https://github.com/thkox/speech-and-audio-processing

2. Κατεβάστε το dataset VOiCES_devkit.tar.gz από εδώ ή μέσα από το onedrive εδώ.

   Αποσυμπιέστε το dataset στο **auxiliary2024/input/**. Τα αρχεία θα πρέπει να έχουν την ακόλουθη δομή:

   ```
   auxiliary2024/input
   ├── VOiCES_devkit
   │      ├── distant-16k
   │      ├── references
   │      ├── source-16k
   ```

   Σημείωση:
   - Δεν χρειάζεστε λογαριασμό **AWS** για να κατεβάσετε το σύνολο δεδομένων. Μπορείτε να το κατεβάσετε απευθείας από τον παραπάνω σύνδεσμο χρησιμοποιώντας **aws cli**.
     - Το σύνολο δεδομένων είναι μεγάλο (περίπου **30GB** το αρχείο **.tar.gz** και περίπου **40GB** ο ασυμπίεστος φάκελος), βεβαιωθείτε ότι έχετε αρκετό χώρο στο σύστημά σας.
   - Αν **δεν** θέλετε να κατεβάσετε το **VOiCES_devkit.tar.gz**, αρκεί να κατεβάσετε το **datasets.zip** που θα βρείτε στο onedrive εδώ.
     - Αποσυμπιέστε το dataset στο **auxiliary2024/** και κάντε αντικατάσταση τον φάκελο datasets που είναι κενός.
     - Για την εκτέλεση του **main.py**: Έχω κάνει σχόλια την γραμμή **9** στη main.py **ld.create_datasets()**, ώστε να μην εμφανιστεί πρόβλημα κατά την εκτέλεση του προγράμματος.
     - Για την εκτέλεση του **main_menu.py**: Αρκεί να μην επιλέξετε την επιλογή **Load the necessary dataset** γιατί το έχουμε ήδη δημιουργήσει.

3. Για να δημιουργήσετε το περιβάλλον conda στο φάκελο του project:

   python setup.py

Ενεργοποίηση του Περιβάλλοντος Conda

Αφού ολοκληρωθεί η δημιουργία του περιβάλλοντος conda, μπορείτε να το ενεργοποιήσετε χρησιμοποιώντας την παρακάτω εντολή:

Σε Windows, MacOS και Linux

conda activate speech-and-audio-processing

Το αρχείο environment.yml καθορίζει τις απαιτούμενες εξαρτήσεις:

- Python 3.11
- TensorFlow 2.16.1
- NumPy 1.25.*
- librosa 0.10.*
- pandas
- inquirer
- speechrecognition
- pydub

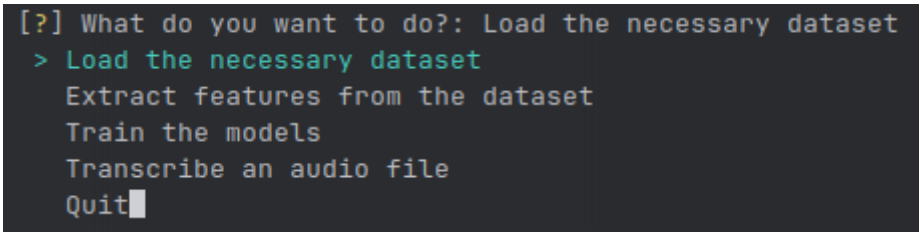
Εκτέλεση του προγράμματος

Αφού ενεργοποιήσετε το περιβάλλον conda, μπορείτε να εκτελέσετε την εφαρμογή είτε χρησιμοποιώντας είτε το **main.py** είτε το **main_replacement.py**.

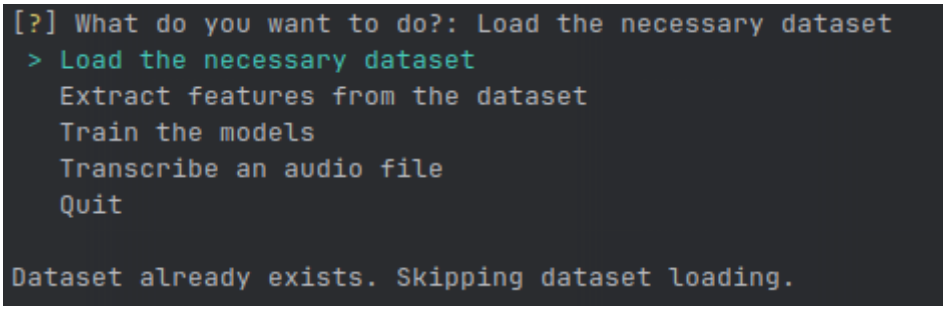
- Το **main.py** είναι το βασικό πρόγραμμα που εκτελεί ολόκληρη τη διαδικασία από τη φόρτωση του συνόλου δεδομένων, την εξαγωγή χαρακτηριστικών, την εκπαίδευση των μοντέλων και την πρόβλεψη. Βρίσκεται μέσα στο directory **source2024**.
- Το **main_replacement.py** είναι ένα διαδραστικό μενού που σας επιτρέπει να επιλέξετε ποιο τμήμα της διαδικασίας θέλετε να εκτελέσετε. Παρέχει επιλογές για τη φόρτωση του συνόλου δεδομένων, την εξαγωγή χαρακτηριστικών, την εκπαίδευση των μοντέλων και την πρόβλεψη ενός ηχητικού αρχείου. Βρίσκεται στο directory **root** του project.

Παράδειγμα Εκτέλεσης Προγράμματος

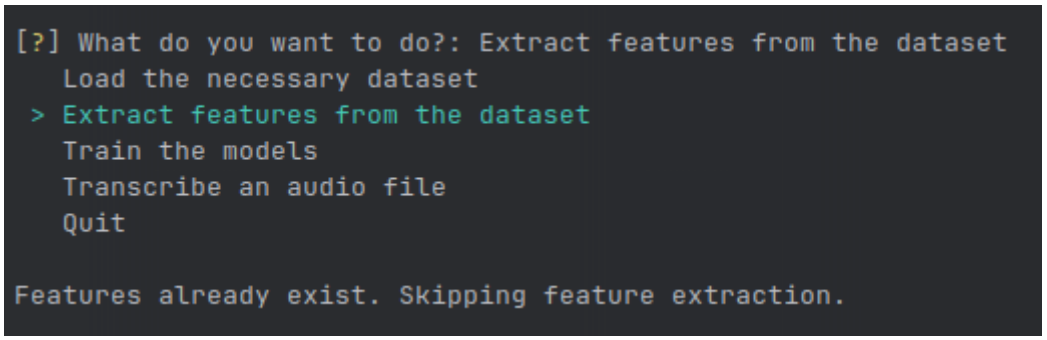
Το παράδειγμα είναι η εκτέλεση του διαδραστικού **main_replacement.py**:



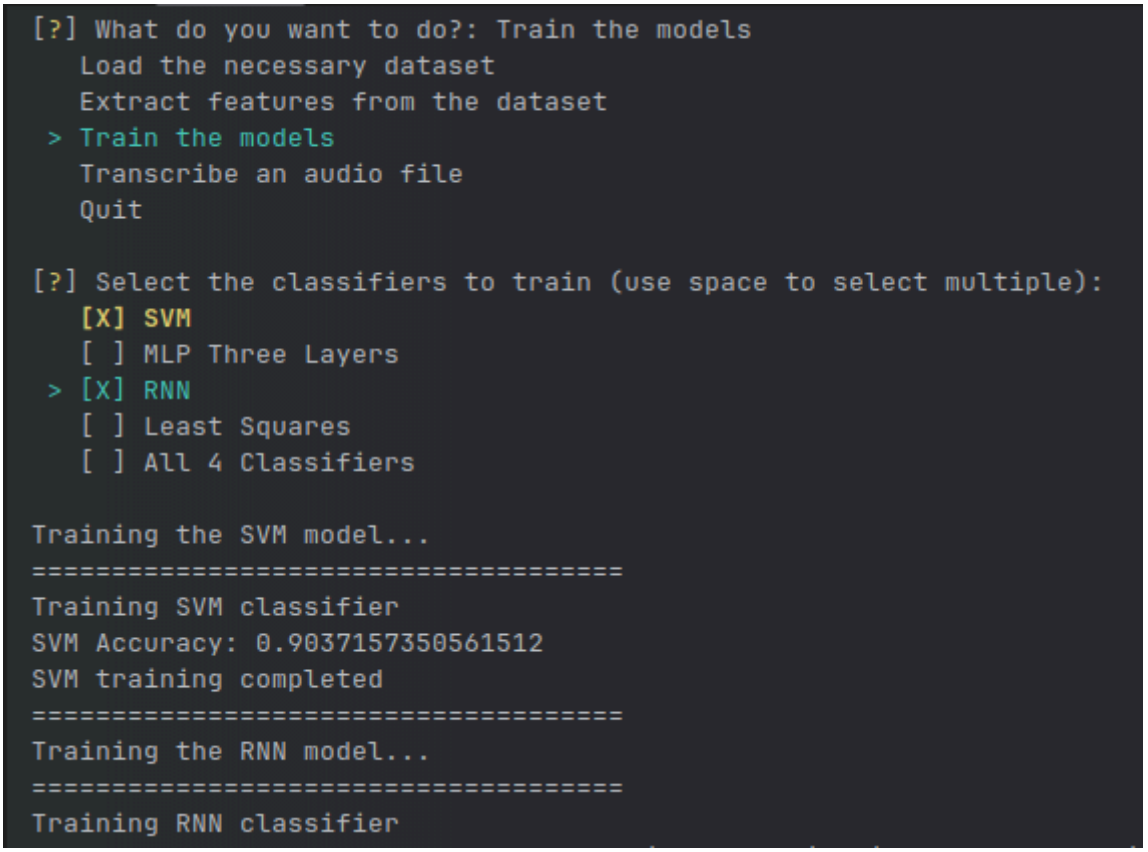
Αρχικό Μενού



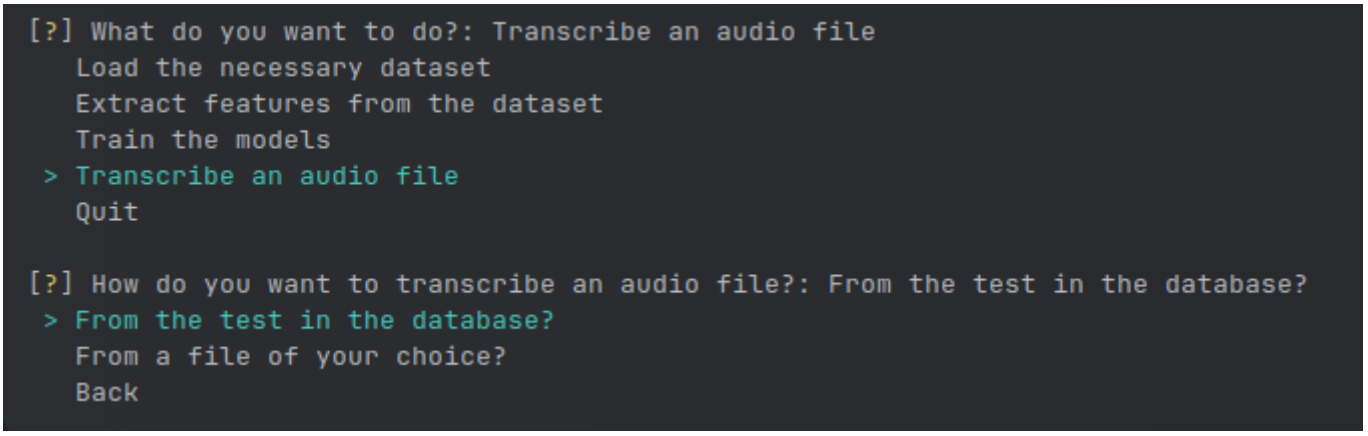
Επιλογή **Load the necessary dataset** με το αντίστοιχο μήνυμα εμφάνισης όταν έχουμε ήδη δημιουργήσει το κατάλληλο dataset.



Επιλογή **Extract features from the dataset** με το αντίστοιχο μήνυμα εμφάνισης όταν έχουμε ήδη εξάγει τα χαρακτηριστικά που θέλουμε από το dataset.



Επιλογή **Train the models** με το αντίστοιχο υπομενού όπου ο χρήστης θα πρέπει να να επιλέξει με το πλήκτρο space τα μοντέλα που θέλει να εκπαιδεύσει.



Επιλογή **Transcribe an audio file** με το αντίστοιχο υπομενού που δίνει τη δυνατότητα στο χρήστη να διαλέξει είτε ένα ηχητικό από το φάκελο test είτε να γράψει το μονοπάτι του αρχείου που βρίσκεται στον δίσκο του για να το ελέγξει.

```
[?] Please select the audio file to transcribe:: Lab41-SRI-VOiCES-rm1-none-sp6446-ch078193-sg0011-mc01-stu-clo-dg160.wav
 > Lab41-SRI-VOiCES-rm1-none-sp6446-ch078193-sg0011-mc01-stu-clo-dg160.wav
   Lab41-SRI-VOiCES-rm1-tele-sp1263-ch139804-sg0021-mc01-stu-clo-dg100.wav
   Lab41-SRI-VOiCES-rm1-none-sp0207-ch122801-sg0022-mc01-stu-clo-dg150.wav
   Lab41-SRI-VOiCES-rm1-none-sp7061-ch088086-sg0040-mc01-stu-clo-dg020.wav
   Lab41-SRI-VOiCES-rm1-tele-sp8465-ch246942-sg0002-mc01-stu-clo-dg110.wav
   Lab41-SRI-VOiCES-rm1-babb-sp5622-ch041172-sg0001-mc01-stu-clo-dg010.wav
   Lab41-SRI-VOiCES-rm1-tele-sp4899-ch032658-sg0012-mc01-stu-clo-dg070.wav
   Lab41-SRI-VOiCES-rm1-musi-sp0154-ch123998-sg0018-mc01-stu-clo-dg040.wav
   Lab41-SRI-VOiCES-rm1-babb-sp2196-ch174172-sg0037-mc01-stu-clo-dg040.wav
   Lab41-SRI-VOiCES-rm1-none-sp0166-ch000352-sg0020-mc01-stu-clo-dg100.wav
   Lab41-SRI-VOiCES-rm1-tele-sp5622-ch044586-sg0026-mc01-stu-clo-dg130.wav
   Lab41-SRI-VOiCES-rm1-babb-sp1379-ch130529-sg0020-mc01-stu-clo-dg140.wav
   Lab41-SRI-VOiCES-rm1-none-sp2952-ch000408-sg0023-mc01-stu-clo-dg160.wav
```

Εμφάνιση όλων των αρχείων που βρίσκονται στο φάκελο test.

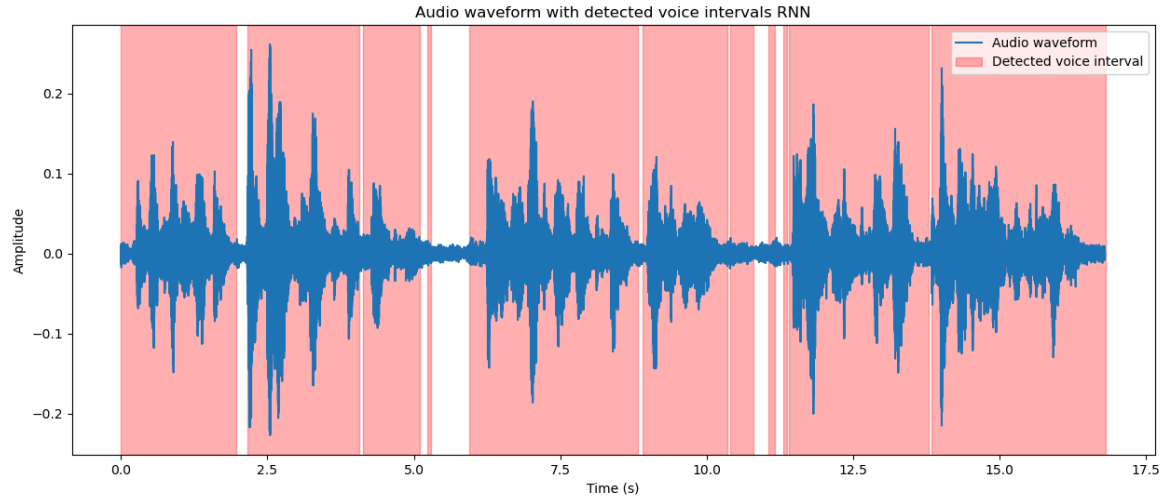
```
[?] How do you want to transcribe an audio file?: From a file of your choice?
   From the test in the database?
 > From a file of your choice?
   Back

Please provide the absolute path to the audio file to transcribe.
File path: auxiliary2024/datasets/test/Lab41-SRI-VOiCES-rm1-babb-sp0175-ch129587-sg0019-mc01-stu-clo-dg000.wav
Transcribing audio file: auxiliary2024/datasets/test/Lab41-SRI-VOiCES-rm1-babb-sp0175-ch129587-sg0019-mc01-stu-clo-dg000.wav
Loading SVM model from auxiliary2024/output/classifiers/svm_model.pkl
SVM model loaded successfully
Loading MLP model from auxiliary2024/output/classifiers/mlp_model.pkl
MLP model loaded successfully
Loading RNN model from auxiliary2024/output/classifiers/rnn_model.keras
```

Εμφάνιση μηνύματος που προτρέπει να τον χρήστη να γράψει στην κονσόλα το απόλυτο μονοπάτι του αρχείου που θα ήθελε να δοκιμάσει.

```
RNN model loaded successfully
Loading Least Squares model from auxiliary2024/output/classifiers/ls_model.pkl
Least Squares model loaded successfully
Loading and extracting features from auxiliary2024/datasets/test/Lab41-SRI-VOiCES-rm1-none-sp6446-ch078193-sg0011-mc01-stu-clo-dg160.wav
```

Τα πρώτα μηνύματα που εμφανίζονται όταν ο χρήστης έχει πατήσει enter για να ξεκινήσουν οι προβλέψεις των ταξινομητών.



Μετά την πρόβλεψη των διαστημάτων στα οποία εντοπίζεται ομιλία, ανά ταξινομητή, εμφανίζεται στην οθόνη ένα νέο παράθυρο με το audio waveform του ηχητικού (μπλε χρώμα) μαζί με τα διαστήματα που εντόπισε ο εκάστοτε ταξινομητής (στη φωτογραφία είναι ο **RNN**).

```
Original intervals:[(0.0, 1.354), (1.828, 2.723), (4.923, 6.4239999999999995)]
Predicted intervals:[(0.0, 3.104), (3.168, 3.2), (3.328, 4.512), (4.608, 4.704), (4.784, 4.848), (4.912, 5.52),
Background accuracy: 7.6766
Voice accuracy: 56.5762
Overall accuracy: 21.2609
```

```
Playing interval: 15.072 to 15.824 seconds
Input #0, wav, from '/tmp/tmpkbbv86_f.wav':   0KB sq=    0B f=0/0
  Duration: 00:00:00.74, bitrate: 256 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 16000 Hz, 1 channels, s16, 256 kb/s
   0.56 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0

No text transcribed for this segment.


Output saved to auxiliary2024/output/results/RNN_Lab41-SRI-VOiCES-rm1-none-sp6446-ch078193-sg0011-mc01-stu-clo-dg160.txt
```

Όταν ο χρήστης κλείσει το παράθυρο με το διάγραμμα, εκτυπώνονται τα διαστήματα τα αρχικά, τα διαστήματα που βρήκε ο ταξινομητής και το ποσοστό ακριβείας. Μετά ξεκινάει η αναπαραγωγή των διαστημάτων που εντόπισε ο κάθε ταξινομητής. Στην οθόνη εμφανίζει το διάστημα που παίζει εκείνη την χρονική στιγμή καθώς και τις λέξεις που εντόπισε στη συγκεκριμένη χρονική στιγμή.

Πηγές

- scikit-learn Documentation: https://scikit-learn.org/stable/user_guide.html , accessed by 24/06/2024
- joblib Documentation: https://joblib.readthedocs.io/en/stable/ , accessed by 25/06/2024
- TensorFlow Documentation: https://www.tensorflow.org/ , accessed by 25/06/2024
- keras: https://keras.io/api/, accessed by 27/06/2024
- VOiCES Citation: https://arxiv.org/abs/1804.05053 , accessed by 26/06/2024
    - Dataset: https://registry.opendata.aws/lab41-sri-voices/ , accessed by 26/06/2024
    - README: https://iqtlabs.github.io/voices/Lab41-SRI-VOiCES_README/ , accessed by 26/06/2024
- pydub: https://github.com/jiaaro/pydub/ , accessed by 28/06/2024
- Rabiner, L. R., & Schafer, R. W. (1978). *Ψηφιακή επεξεργασία Φωνής: Θεωρία και Εφαρμογές*. Prentice Hall.