Πανεπιστήμιο Πειραιώς
Τμήμα Πληροφορικής
Έτος: 2023 - 2024

Μάθημα:
«ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΩΝ ΦΩΝΗΣ ΚΑΙ ΗΧΟΥ»
Εργασία:Ατομική Εργασια

Εξάμηνο: 8ο

Στοιχεια:

Ονομα:Αιμιλιανός Κουρπάς-Δανάς

ΑΜ : Π-20100

Ημερομηνια παραδοσης : 4.07.2024

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:

α) 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.

β) τα αρχεία source code σε ένα συμπιεσμένο αρχείο με όνομα source2023.zip (ή .rar ή άλλη σχετική κατάληξη).

γ) 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.

Α) 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.

Β) 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!

Λυση

Αλγοριθμική Περιγραφή

Στην εργασία αυτή, παίρνουμε ένα σήμα και εντοπίζουμε τις περιοχές ομιλίας σε αυτό το σήμα. Δηλαδή, διαχωρίζουμε το σήμα σε background (δεν είναι ομιλία) και foreground (σήμα ομιλίας) χρησιμοποιώντας έναν ταξινομητή.

Αυτό είναι ένα δυαδικό πρόβλημα, δηλαδή πρόκειται για δυαδική ταξινόμηση. Κατά συνέπεια, ταξινομούμε κάθε frame ήχου ως frame background ή foreground. Σαρώνουμε το σήμα μας με την τεχνική κινούμενου παραθύρου. Σε κάθε παράθυρο, εξάγουμε ορισμένες μετρήσεις πριν προχωρήσουμε στην ταξινόμηση. Η συνάρτηση που υλοποιεί την τεχνική παραθύρου και υπολογίζει κάποια φασματικά χαρακτηριστικά είναι το mel spectrogram. Πρόκειται για μια συνάρτηση η οποία παρέχει φασματογράφημα ελαττωμένης διάστασης. Με αυτόν τον τρόπο, λαμβάνουμε μια ακολουθία διανυσμάτων και μια ελαττωμένη αναπαράσταση από block δείγματα που έχουμε επιλέξει.

Μετά το mel spectrogram, έχουμε την αναπαράσταση διανυσμάτων:

$$[\chi_1, \chi_2, \chi_3, \ldots, \chi_{v-1}, \chi_{v}]$$

Πώς θα χρησιμοποιήσουμε τους ταξινομητές προκειμένου να βρούμε τα όρια ομιλίας; Αρχικά, αυτό εξαρτάται από τον ταξινομητή (θα αναφερθούμε συγκεκριμένα στον καθένα παρακάτω). Γενικά, όλοι υιοθετούν μια λογική όπου δέχονται το χ_j το οποίο περνάει από τον ταξινομητή και επιστρέφει το $P(f|\chi_j)$ και το $P(b|\chi_j)$, δηλαδή δίνει μια απόφαση ανά frame. Αυτό μπορεί να το δώσει μόνο ένας εκπαιδευμένος ταξινομητής. Όταν μας δώσει αυτή την ακολουθία, θα έχει τη μορφή:

$$[0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0]$$

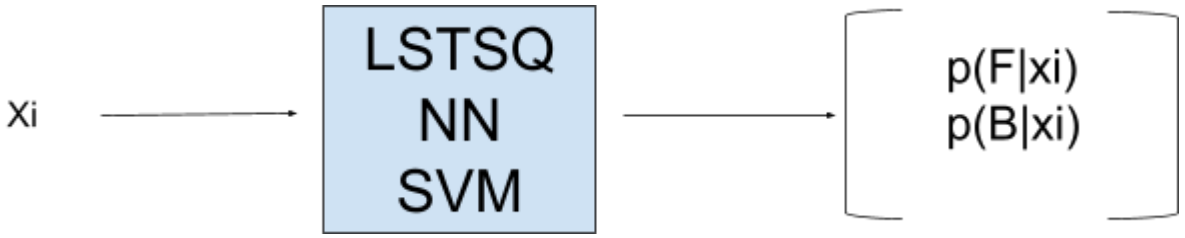
Επίσης, υπάρχουν οι κρίσιμες λέξεις. Σε αυτές τις λέξεις, μπορεί να υπάρχουν τμήματα που είναι ασθενή και έχουν περισσότερα χαρακτηριστικά υποβάθρου παρά ομιλίας. Για να αφαιρεθούν αυτά, χρησιμοποιούμε ένα φίλτρο μεσαίας τιμής (median filter). Αυτό σαρώνει μια ακολουθία αποφάσεων και, ανάλογα με το μήκος του L, μπορεί να αφαιρέσει έως και $(L-1)/2$ διαδοχικά σφάλματα. Άρα, αν το μέγεθος είναι 5, μπορεί να αφαιρέσει έως και 2 σφάλματα. Με αυτόν τον τρόπο, καθαρίζουμε την ακολουθία σφαλμάτων.

Συνοπτικά, τα βήματα που ακολουθούμε είναι:

1) Feature extraction με το mel spectrogram.
2) Classification per frame ή per vector.
3) Post processing με median filter.
4) Scanning - σάρωση της ακολουθίας.

<u>Ταξινομητες</u>

Multilayer Perceptron-Support Vector Machine-Least Squares



Ο τρόπος που λειτουργούν οι LSQ, SVM και MLP είναι χειροκίνητος. Ο MLP (Multilayer Perceptron) δίνει αποτελέσματα 0 και 1, καθώς στον τελευταίο νευρώνα έχουμε προσθέσει sigmoid και στο loss χρησιμοποιούμε binary cross entropy. Μπορούμε να χρησιμοποιήσουμε sigmoid μόνο και μόνο επειδή το πρόβλημα είναι binary.

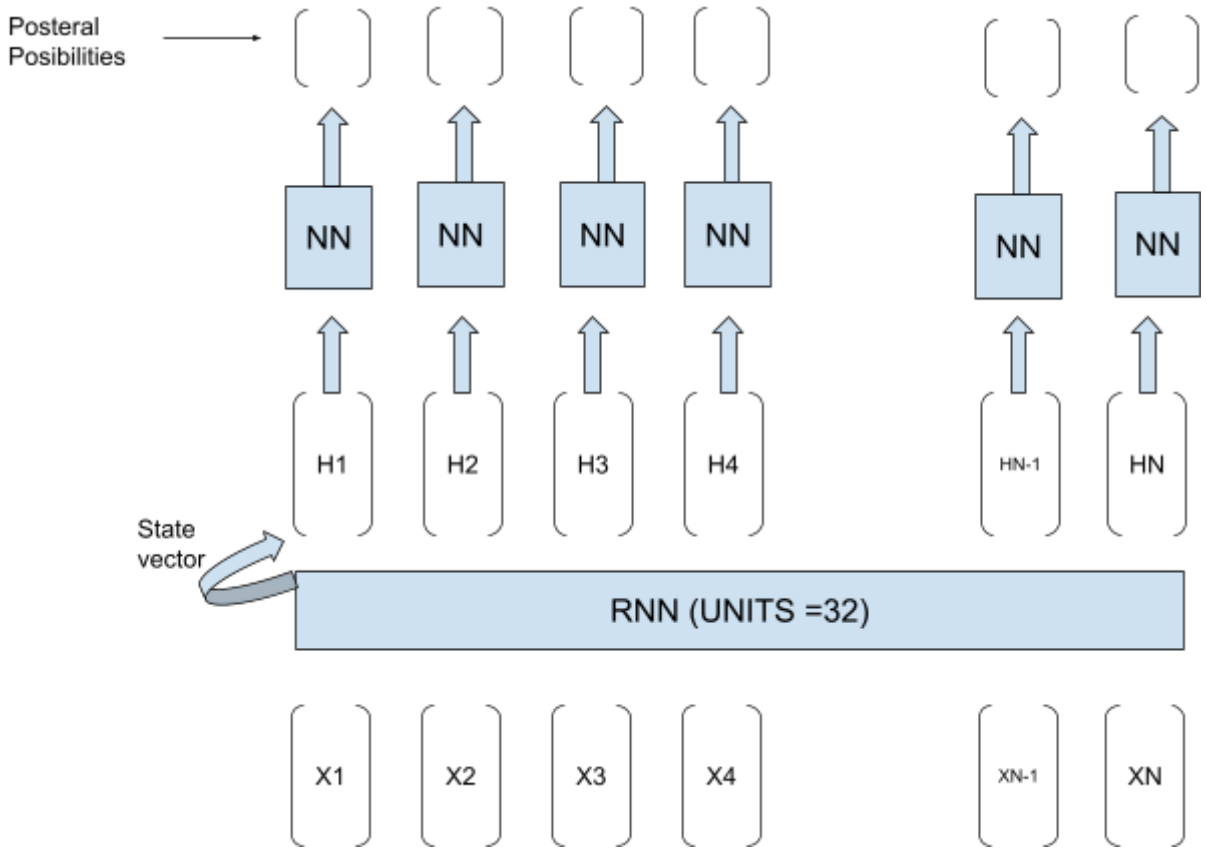
Όσον αφορά τον SVM (Support Vector Machine) και τον LSQ (Least Squares), δεν θα δώσουν ακριβώς 0 και 1 αλλά θα δώσουν θετικούς και αρνητικούς αριθμούς, γι' αυτό χρειαζόμαστε έναν γραμμικό ταξινομητή. Αυτό συμβαίνει γιατί δημιουργούν ένα υπερεπίπεδο σε δύο χώρους, θετικό και αρνητικό.

Το διάγραμμα απεικονίζει τον τρόπο λειτουργίας των μοντέλων:

1. Είσοδος \(\chi_i\) στον ταξινομητή (LSQ, NN, SVM).

2. Ο ταξινομητής δίνει ως έξοδο τις πιθανότητες \(p(F|\chi_i)\) και \(p(B|\chi_i)\).

Recurrent Neural Network



Το διάγραμμα απεικονίζει τη λειτουργία ενός RNN (Recurrent Neural Network) που υλοποιείται με το TensorFlow, χρησιμοποιώντας τον πυρήνα RNN με την επιλογή `return_sequences=True`. Αυτό σημαίνει ότι το δίκτυο θα επιστρέφει μια ακολουθία αποτελεσμάτων ανά frame.

Το RNN δέχεται ως είσοδο τα διανύσματα

\(\chi_1, \chi_2, \chi_3, \ldots, \chi_N\) και παράγει ένα διάνυσμα κατάστασης σε κάθε χρονική στιγμή. Το state vector από κάθε χρονική στιγμή μεταφέρεται στην επόμενη.

Για να μπορέσουμε να πάρουμε αποφάσεις ανά frame, χρειαζόμαστε ένα layer από νευρώνες που θα αποδέχονται την έξοδο του RNN σε κάθε frame και θα τη μετασχηματίζουν σε μια απόφαση. Αυτό γίνεται με τη χρήση neural networks (NN) για κάθε έξοδο του RNN, τα οποία υπολογίζουν τις πιθανολογικές κατανομές \(P(F|\chi_i)\) και \(P(B|\chi_i)\).

Το RNN χτίζει ένα διάνυσμα κατάστασης κάθε χρονική στιγμή, μεταφέροντας πληροφορίες από το προηγούμενο διάνυσμα κατάστασης. Αυτό επιτρέπει στο RNN να δίνει καλύτερα αποτελέσματα, καθώς λαμβάνει υπόψη την ακολουθία των δεδομένων.

Λεπτομέρειες Υλοποίησης

1. TensorFlow RNN Kernel: Η υλοποίηση γίνεται με χρήση του πυρήνα RNN του TensorFlow.

2. Return Sequences: Η επιλογή `return_sequences=True` επιτρέπει την επιστροφή μιας ακολουθίας αποτελεσμάτων ανά frame.

3. Νευρωνικό Δίκτυο για Απόφαση: Ένα layer από νευρώνες χρησιμοποιείται για να αποδέχεται την έξοδο του RNN σε κάθε frame και να τη μετασχηματίζει σε απόφαση.

4. Διάνυσμα Κατάστασης: Το RNN δημιουργεί ένα διάνυσμα κατάστασης σε κάθε χρονική στιγμή, λαμβάνοντας υπόψη την ακολουθία των προηγούμενων διανυσμάτων.

Το διάγραμμα απεικονίζει:

1. Είσοδος διανυσμάτων $\chi_1, \chi_2, \chi_3, \ldots, \chi_N$ στο RNN.

2. Το RNN παράγει ένα state vector σε κάθε χρονική στιγμή.

3. Τα state vectors περνούν από τα neural networks (NN) για να παραχθούν οι πιθανότητες $P(F|\chi_i)$ και $P(B|\chi_i)$.

4. Τα αποτελέσματα παρουσιάζονται ως πιθανότητες posterior.

Εκπαιδευση ταξινομητων - Δεδομενα.

Για την φάση εκπαίδευσης (training phase) των ταξινομητών, ως foreground dataset χρησιμοποιήθηκε το «*Common Voice Corpus Delta Segment 18.0 (6/19/2024)*»
και ως background dataset χρησιμοποιήθηκε το «ESC-50» από την ιστοσελίδα *Harvard Dataverse*
Επειδή τα datasets ήταν πολύ μεγάλα, από το σύνολο των ηχητικών διατήρησα μόνο 150 φακελους απο το καθε ενα.CLOUD_LINK.

Προετοιμασία Δεδομένων
Τα δεδομένα ήχου φορτώνονται από φακέλους με κλιπς ομιλίας (foreground) και μη ομιλίας (background). Για κάθε κλιπ ήχου:

1. Εξάγουμε χαρακτηριστικά χρησιμοποιώντας mel spectrogram.

2. Εξασφαλίζουμε ότι όλα τα χαρακτηριστικά έχουν το ίδιο μέγεθος μέσω padding.

3. Ετοιμάζουμε τα διανύσματα χαρακτηριστικών και τις αντίστοιχες ετικέτες (labels).

Εκπαίδευση MLP (Multilayer Perceptron)

Ο MLP χρησιμοποιεί στρώσεις πλήρους σύνδεσης (Dense layers) με ενεργοποίηση ReLU και sigmoid στην έξοδο για δυαδική ταξινόμηση. Η διαδικασία εκπαίδευσης περιλαμβάνει:

- Δύο στρώσεις πλήρους σύνδεσης με 128 και 64 νευρώνες αντίστοιχα.

- Χρήση Dropout για αποφυγή υπερπροσαρμογής.

- Ενεργοποίηση sigmoid στην έξοδο με binary cross entropy ως συνάρτηση απώλειας.

Εκπαίδευση SVM (Support Vector Machine)

Ο SVM είναι ένας γραμμικός ταξινομητής που χρησιμοποιεί το LinearSVC από το scikit-learn. Η διαδικασία περιλαμβάνει:

- Εκπαίδευση με LinearSVC.

- Αποθήκευση του εκπαιδευμένου μοντέλου.

Εκπαίδευση Least Squares (LSQ)

Ο Least Squares χρησιμοποιεί τη μέθοδο ελαχίστων τετραγώνων για να υπολογίσει τα βάρη της εξίσωσης που χωρίζει τα δεδομένα σε δύο κατηγορίες. Η διαδικασία περιλαμβάνει:

- Υπολογισμό των βάρων.

Ο RNN εκπαιδεύεται χρησιμοποιώντας το TensorFlow και το SimpleRNN layer. Η διαδικασία περιλαμβάνει:

- Είσοδο δεδομένων ως ακολουθία διανυσμάτων.

- Χρήση SimpleRNN με 32 μονάδες και ενεργοποίηση sigmoid στην έξοδο.

- Απαιτεί ίσης διάρκειας ηχητικά

Συνοπτικά, η εκπαίδευση των μοντέλων περιλαμβάνει τη χρήση νευρωνικών δικτύων, υπολογιστικών τεχνικών και γραμμικών ταξινομητών για τη δυαδική ταξινόμηση σημάτων ήχου. Η κάθε μέθοδος έχει τα δικά της πλεονεκτήματα και εφαρμογές, ανάλογα με τα χαρακτηριστικά των δεδομένων και τις απαιτήσεις της εφαρμογής.

Εξεταση δεδομενων

Για την εξέταση δεδομένων έχουμε 3 wav,json,txt files.Τα 3 φωνιτικα ειναι 5-10-20 δευτερολεπτα και στο txt περιεχονται οι λεξεις ενω στο json εχουμε τα timestamps για την καθε λεξη.Αυτο θα μας βοηθησει στην συγκριση των μοντελων.

Δοκιμή MLP (Multilayer Perceptron)

Ο MLP-LSTSQ-SVM-RNN φορτώνεται από το αποθηκευμένο μοντέλο και χρησιμοποιείται για την πρόβλεψη των πιθανοτήτων για κάθε frame. Οι προβλέψεις στη συνέχεια φιλτράρονται χρησιμοποιώντας median filter και ανιχνεύονται τα διαστήματα ομιλίας.

Λεπτομέρειες Δοκιμής

1. Φόρτωση Μοντέλων: Φορτώνονται τα εκπαιδευμένα μοντέλα (MLP, SVM, LSQ, RNN).

2. Επεξεργασία Αρχείων Ήχου: Τα αρχεία ήχου επεξεργάζονται για την εξαγωγή χαρακτηριστικών Mel Spectrogram.

3. Πρόβλεψη: Κάθε ταξινομητής προβλέπει τις πιθανότητες για κάθε frame.

4. Φιλτράρισμα Προβλέψεων: Οι προβλέψεις φιλτράρονται χρησιμοποιώντας median filter.

5. Ανίχνευση Διαστημάτων Ομιλίας: Ανιχνεύονται τα διαστήματα ομιλίας από τις προβλέψεις.

Υλοποίηση Προγράμματος

Γλώσσα Υλοποίησης: Python(3.12.4)

Αρχεια: ειναι 2 αρχεια 1) train.py 2) test.py

Βιβλιοθηκες:

Ο κώδικας χρησιμοποιεί διάφορες βιβλιοθήκες για την επεξεργασία δεδομένων, την εξαγωγή χαρακτηριστικών, την εκπαίδευση και τη δοκιμή μοντέλων μηχανικής μάθησης και βαθιάς μάθησης. Παρακάτω παρατίθενται οι βιβλιοθήκες που χρησιμοποιούνται, καθώς και συνδέσμους προς την επίσημη τεκμηρίωσή τους.

1) **os** : Η βιβλιοθήκη `os` παρέχει έναν τρόπο αλληλεπίδρασης με το λειτουργικό σύστημα, επιτρέποντας την εκτέλεση λειτουργιών όπως η διαχείριση αρχείων και φακέλων.

2) **numpy**:Η numpy είναι μια βιβλιοθήκη για την υποστήριξη μεγάλων πολυδιάστατων πινάκων και μητρών, μαζί με μια μεγάλη συλλογή μαθηματικών συναρτήσεων για να δουλεύουν πάνω σε αυτά τα αντικείμενα.

3) **json** : Η βιβλιοθήκη `json` παρέχει μεθόδους για την ανάλυση και τη δημιουργία δεδομένων σε μορφή JSON (JavaScript Object Notation).

4) **librosa** : Η `librosa` είναι μια βιβλιοθήκη για την ανάλυση ήχου με την Python. Παρέχει λειτουργίες για την εξαγωγή χαρακτηριστικών, τη φόρτωση και αποθήκευση αρχείων ήχου, και άλλες εργασίες επεξεργασίας ήχου.

5) **joblib**: Η joblib είναι μια βιβλιοθήκη για την αποθήκευση και την επαναφόρτωση αντικειμένων Python, χρήσιμη για την αποθήκευση εκπαιδευμένων μοντέλων.

6) **sklearn** : Το `scikit-learn` είναι μια βιβλιοθήκη για μηχανική μάθηση στην Python, που παρέχει εργαλεία για την ταξινόμηση, την

παλινδρόμηση, τη συστάδα (clustering), και την προεπεξεργασία δεδομένων.

7) **tensorflow.keras**: Η `keras` είναι ένα υψηλού επιπέδου API για την ανάπτυξη και την εκπαίδευση μοντέλων βαθιάς μάθησης, ενσωματωμένο στο TensorFlow. Παρέχει ένα απλό και ευέλικτο τρόπο για να δημιουργείτε νευρωνικά δίκτυα.

Συναρτησεις:

TRAIN.py

1) **load_train_audio_clips**(limit=None) Φορτώνει κλιπς ήχου εκπαίδευσης για foreground και background

2) **load_foreground_train_audio_clips**(limit=None) Φορτώνει κλιπς ήχου foreground από το φάκελο δεδομένων εκπαίδευσης.

3) **load_background_train_audio_clips**(limit=None) Φορτώνει clips ήχου background από το φάκελο δεδομένων εκπαίδευσης.

4) **load_train_audio_clip**(audio_name, director) Φορτώνει ένα συγκεκριμένο κλιπ ήχου από τον δεδομένο κατάλογο.

5) **extract_features**(audio_clip) Εξάγει χαρακτηριστικά mel spectrogram από ένα κλιπ ήχου.

6) **pad_features**(features, expected_frames) Προσθέτει padding στα χαρακτηριστικά για να έχουν το αναμενόμενο μέγεθος.

7) **train_mlp_classifier**(train_features, train_labels) Εκπαιδεύει ένα πολυεπίπεδο perceptron (MLP) ταξινομητή με τα δεδομένα εκπαίδευσης.

8) **train_svm_classifier**(train_features, train_labels) Εκπαιδεύει έναν SVM ταξινομητή με τα δεδομένα εκπαίδευσης.

9) **train_lstsq_classifier**(train_features, train_labels) Εκπαιδεύει έναν Least Squares ταξινομητή με τα δεδομένα εκπαίδευσης.

10) **train_rnn_classifier**(train_features, train_labels) Εκπαιδεύει ένα RNN ταξινομητή με τα δεδομένα εκπαίδευσης.

TEST.py

1) **compute_mel_spectrogram**(y, sr, n_fft, hop_length, n_mels, window_type) Υπολογίζει το mel spectrogram ενός σήματος ήχου.

2) **process_audio_file**(audio_path, sr) Επεξεργάζεται ένα αρχείο ήχου και επιστρέφει το mel spectrogram και το σήμα ήχου.

3) **detect_conversations**(sequence, threshold=0.5) Ανιχνεύει συνομιλίες σε μια ακολουθία προβλέψεων με βάση ένα κατώφλι.

4) **extract_actual_intervals**(json_file)Εξάγει τα πραγματικά διαστήματα ομιλίας από ένα αρχείο JSON.

5) **read_words_from_txt**(txt_file)Διαβάζει λέξεις από ένα αρχείο κειμένου.

6) **detect_words**(actual_intervals, predicted_intervals, words, tolerance) Ανιχνεύει λέξεις που βρίσκονται εντός προβλεπόμενων διαστημάτων με ανοχή.

7) **interval_within_tolerance**(act_start, act_end, pred_intervals, tolerance) Ελέγχει αν ένα πραγματικό διάστημα βρίσκεται εντός των προβλεπόμενων διαστημάτων με ανοχή.

8) **pad_spectrogram**(spectrogram, expected_frames) Προσθέτει padding σε ένα spectrogram για να έχει το αναμενόμενο μέγεθος.

9) **predict_rnn**(rnn_model, log_mel_spectrogram, expected_frames, n_mels) Κάνει προβλέψεις με ένα RNN μοντέλο χρησιμοποιώντας mel spectrogram δεδομένα.

10) **median_filter**(sequence, kernel_size=5) Εφαρμόζει φίλτρο μεσαίας τιμής σε μια ακολουθία προβλέψεων.

11) **compute_autocorrelation**(signal, sr, threshold=0.7) Υπολογίζει την αυτοσυσχέτιση ενός σήματος και ανιχνεύει κορυφές.

12) **print_menu**() Εκτυπώνει ένα μενού επιλογών για αρχεία ήχου και ταξινομητές.

13) **process_and_evaluate**(file_choice, classifier_choice)Επεξεργάζεται και αξιολογεί τα αρχεία ήχου χρησιμοποιώντας τον επιλεγμένο ταξινομητή.

14) **normalize_features**(features) Κανονικοποιεί τα χαρακτηριστικά αφαιρώντας τη μέση τιμή και διαιρώντας με την τυπική απόκλιση.

Το τεχνικό κομμάτι:

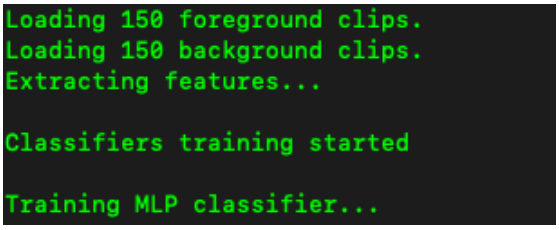
Οδηγίες για εκτέλεση του προγράμματος:

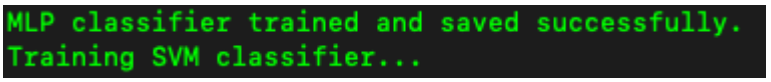
- Χρειάζεται να έχουμε εγκατεστημένη την python
- Χρειαζεται να εχουμε τις βιβλιοθήκες **pip install numpy librosa joblib scikit-learn tensorflow**
- Βρισκουμε το terminal στο path που βρίσκεται το train.py , και γράφετε `python3 train.py` μετα θα αναπαραγουν τα μοντελα μας τους φακελους

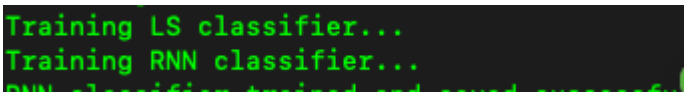
- Βρισκουμε το terminal στο path που βρίσκεται το train.py , και γράφετε `python3 test.py` και θα παρουμε τα αποτελεσματα.

Παράδειγμα:

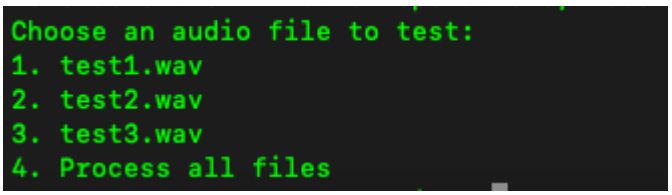
train.py

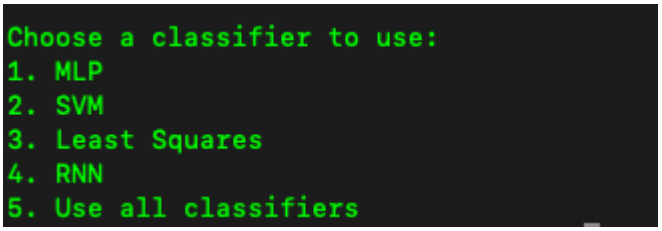






test.py

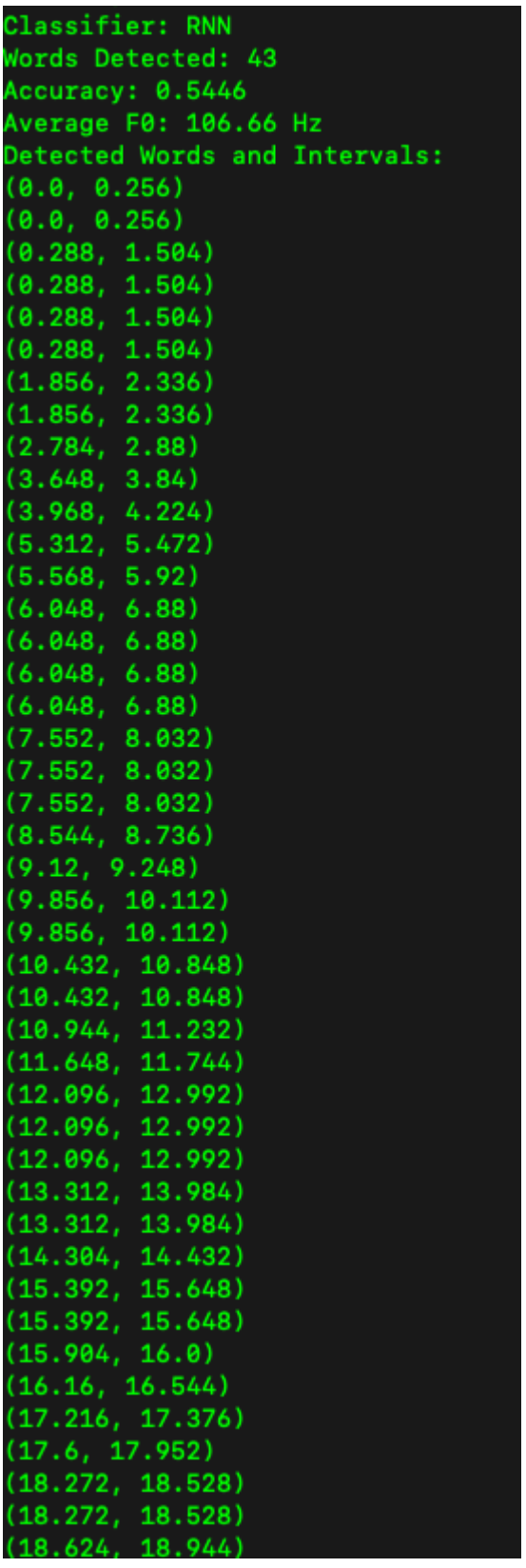



Choose a classifier to use:
1. MLP
2. SVM
3. Least Squares
4. RNN
5. Use all classifiers

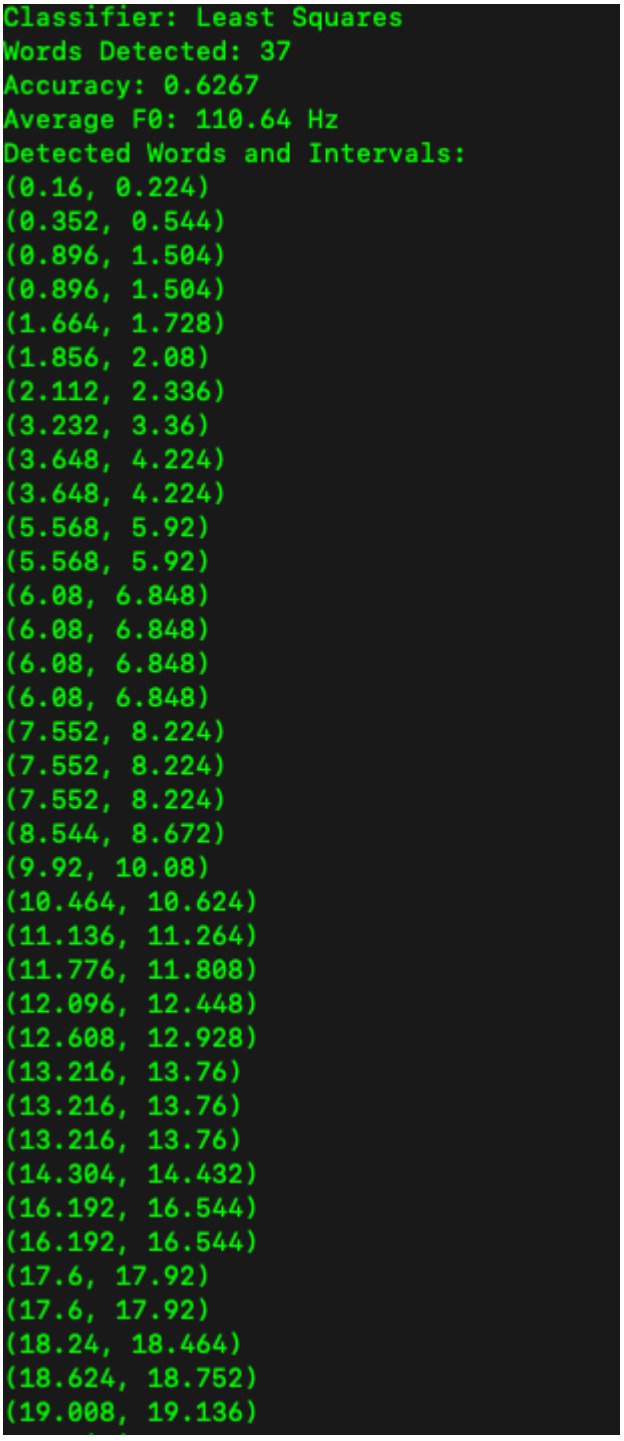
```
Classifier: MLP
Words Detected: 39
Accuracy: 0.6092
Average F0: 107.45 Hz
Detected Words and Interval
(0.16, 0.224)
(0.352, 0.576)
(0.64, 1.504)
(0.64, 1.504)
(0.64, 1.504)
(1.856, 2.4)
(1.856, 2.4)
(1.856, 2.4)
(2.752, 2.912)
(2.752, 2.912)
(3.04, 3.296)
(3.648, 3.776)
(4.128, 4.192)
(5.344, 5.44)
(5.568, 5.92)
(6.08, 6.912)
(6.08, 6.912)
(6.08, 6.912)
(6.08, 6.912)
(7.552, 7.712)
(7.744, 8.16)
(7.744, 8.16)
(9.92, 10.048)
(10.944, 11.168)
(10.944, 11.168)
(12.096, 12.992)
(12.096, 12.992)
(12.096, 12.992)
(13.184, 13.824)
(13.184, 13.824)
(14.304, 14.432)
(15.488, 15.648)
(16.192, 16.512)
(16.192, 16.512)
(17.6, 17.92)
(17.6, 17.92)
(18.272, 18.4)
(18.656, 18.912)
(18.656, 18.912)
```

```
Classifier: SVM
Words Detected: 47
Accuracy: 0.5538
Average F0: 110.66 Hz
Detected Words and Intervals:
(0.0, 0.128)
(0.16, 0.224)
(0.352, 0.544)
(0.896, 1.536)
(0.896, 1.536)
(1.632, 1.792)
(1.888, 2.08)
(2.112, 2.4)
(2.112, 2.4)
(3.232, 3.392)
(3.648, 4.224)
(3.648, 4.224)
(4.416, 4.608)
(4.416, 4.608)
(4.96, 5.12)
(5.568, 5.92)
(5.568, 5.92)
(6.08, 6.848)
(6.08, 6.848)
(6.08, 6.848)
(6.08, 6.848)
(7.52, 8.224)
(7.52, 8.224)
(7.52, 8.224)
(8.512, 8.736)
(8.928, 9.76)
(8.928, 9.76)
(8.928, 9.76)
(9.92, 10.112)
(10.528, 10.592)
(11.136, 11.264)
(11.424, 11.552)
(11.648, 11.936)
(11.648, 11.936)
(12.16, 12.448)
(12.608, 12.928)
(13.152, 13.632)
(13.152, 13.632)
(14.304, 14.432)
(15.936, 16.064)
(16.192, 16.48)
(16.992, 17.344)
(16.992, 17.344)
(17.536, 17.856)
(18.24, 18.528)
(18.24, 18.528)
(18.976, 19.136)
```

```
Classifier: RNN
Words Detected: 43
Accuracy: 0.5446
Average F0: 106.66 Hz
Detected Words and Intervals:
(0.0, 0.256)
(0.0, 0.256)
(0.288, 1.504)
(0.288, 1.504)
(0.288, 1.504)
(0.288, 1.504)
(1.856, 2.336)
(1.856, 2.336)
(2.784, 2.88)
(3.648, 3.84)
(3.968, 4.224)
(5.312, 5.472)
(5.568, 5.92)
(6.048, 6.88)
(6.048, 6.88)
(6.048, 6.88)
(6.048, 6.88)
(7.552, 8.032)
(7.552, 8.032)
(7.552, 8.032)
(8.544, 8.736)
(9.12, 9.248)
(9.856, 10.112)
(9.856, 10.112)
(10.432, 10.848)
(10.432, 10.848)
(10.944, 11.232)
(11.648, 11.744)
(12.096, 12.992)
(12.096, 12.992)
(12.096, 12.992)
(13.312, 13.984)
(13.312, 13.984)
(14.304, 14.432)
(15.392, 15.648)
(15.392, 15.648)
(15.904, 16.0)
(16.16, 16.544)
(17.216, 17.376)
(17.6, 17.952)
(18.272, 18.528)
(18.272, 18.528)
(18.624, 18.944)
```



RNN

```
Classifier: Least Squares
Words Detected: 37
Accuracy: 0.6267
Average F0: 110.64 Hz
Detected Words and Intervals:
(0.16, 0.224)
(0.352, 0.544)
(0.896, 1.504)
(0.896, 1.504)
(1.664, 1.728)
(1.856, 2.08)
(2.112, 2.336)
(3.232, 3.36)
(3.648, 4.224)
(3.648, 4.224)
(5.568, 5.92)
(5.568, 5.92)
(6.08, 6.848)
(6.08, 6.848)
(6.08, 6.848)
(6.08, 6.848)
(7.552, 8.224)
(7.552, 8.224)
(7.552, 8.224)
(8.544, 8.672)
(9.92, 10.08)
(10.464, 10.624)
(11.136, 11.264)
(11.776, 11.808)
(12.096, 12.448)
(12.608, 12.928)
(13.216, 13.76)
(13.216, 13.76)
(13.216, 13.76)
(14.304, 14.432)
(16.192, 16.544)
(16.192, 16.544)
(17.6, 17.92)
(17.6, 17.92)
(18.24, 18.464)
(18.624, 18.752)
(19.008, 19.136)
```

LSTSQ

Συμπεράσματα – Παρατηρήσεις

Βάσει των ποσοστών ακρίβειας που δίνουν οι ταξινομητές από την σύγκριση της ground truth ακολουθίας και της predicted ακολουθίας, φαίνεται ότι οι LS και SVM ταξινομητές δεν είναι τόσο αποτελεσματικοί όσο οι MLP και RNN ταξινομητές.

Βιβλιογραφια

-Σημειωσεις Μαθηματος

-Σημεωσεις GUNET
https://thales.cs.unipi.gr/modules/document/?course=TMD103

-https://docs.python.org/3/library/os.html

- https://numpy.org/doc/stable/

- https://librosa.org/doc/latest/

- https://joblib.readthedocs.io/en/latest/

-https://scikit-learn.org/stable/documentation.html

- https://keras.io/

- https://www.tensorflow.org/api_docs/python/tf/keras

- https://docs.python.org/3/library/json.html