ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

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

Ονοματεπώνυμο	Δημήτρης Στυλιανού
Αριθμός Μητρώου	Π20004
Υπεύθυνος Καθηγητής	Άγγελος Πικράκης

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:

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!

Εισαγωγή

Επισημάνσεις:

Στην παρούσα τεχνική αναφορά θα χρησιμοποιώ την εξής αγγλική ορολογία έναντι της ελληνικής:

- *σύνολο δεδομένων προσκηνίου* → *foreground dataset*
- *σύνολο δεδομένων υποβάθρου* → *background dataset*
- *θόρυβος υποβάθρου/βάθους* → *background noise*
- *ετικέτα* → *label*
- *πλαίσιο* → *frame*

Για την επιτυχής εκτέλεση των προγραμμάτων, πρέπει να έχετε εγκαταστήσει τις βιβλιοθήκες που αναγράφονται στο αρχείο *requirements.txt***. Για να δείτε πώς μπορείτε να δημιουργήσετε ένα virtual environment, πατήστε** *εδώ***.**

Για την επιτυχής αναπαραγωγή των ηχητικών μέσω του προγράμματος *word_detector.py***, πρέπει να έχετε εγκαταστημένο στον υπολογιστή σας τον** *VLC media player***.**

Η εκπόνηση της εργασίας έγινε στην γλώσσα προγραμματισμού Python (version 3.12.4).

Για την φάση εκπαίδευσης (training phase) των ταξινομητών, ως foreground dataset χρησιμοποιήθηκε το «*Common Voice Corpus Delta Segment 17.0 (3/20/2024)*» από την ιστοσελίδα *Common Voice by Mozilla* και ως background dataset χρησιμοποιήθηκε το «ESC-50» από την ιστοσελίδα *Harvard Dataverse*, το οποίο είναι δημοσιευμένο και στο *GitHub*.

Επειδή το foreground dataset ήταν πολύ μεγάλο, από το σύνολο των ηχητικών διατήρησα μόνο όσα ήταν στην κατηγορία validated και είχαν διάρκεια μεγαλύτερη των τεσσάρων δευτερολέπτων. Η επιλογή των τεσσάρων δευτερολέπτων έγινε για σκοπούς απλότητας, καθώς η υλοποίηση του RNN (Recurrent Neural Network) ταξινομητή απαιτεί ίσης διάρκειας ηχητικά. Περεταίρω επεξήγηση θα δοθεί στο κυρίως μέρος της τεχνικής αναφοράς.

Για την φάση ελέγχου (testing phase) των ταξινομητών, τα ηχητικά που χρησιμοποιήθηκαν, δημιουργήθηκαν με την βοήθεια του *ElevenLabs Text To Speech With Timestamps API*. Έγινε χρήση του προαναφερθέντος API διότι μας δίνει την δυνατότητα παραγωγής ομιλίας βάσει ενός δοθέντος κειμένου, αλλά κυρίως γιατί εκτός της παραγόμενης ομιλίας, το API επιστρέφει και τις χρονικές στιγμές εμφάνισης του κάθε χαρακτήρα/γράμματος μέσα στην ομιλία. Επομένως, έχοντας τις χρονικές στιγμές κάθε χαρακτήρα/γράμματος, με μια μικρή επεξεργασία αυτών, δημιούργησα και αποθήκευσα σε μορφή json την χρονική στιγμή εμφάνισης (σε δευτερόλεπτα) κάθε λέξης μέσα στην παραχθείσα ομιλία.

Σημείωση: Θα ακολουθήσει επεξήγηση της θεωρητικής προσέγγισης για την υλοποίηση της εργασίας, με όσο το δυνατόν λιγότερη αναφορά σε κομμάτια του κώδικα, καθώς έχει γίνει πλήρης τεκμηρίωση των βημάτων και συναρτήσεων μέσα στον κώδικα.

Κύριο Μέρος

Μέρος Α

Εκπαίδευση ταξινομητών

Το αρχείο *train_classifiers.py* είναι υπεύθυνο για την εκπαίδευση των ταξινομητών.

Αρχικά, φορτώνω τα ηχητικά αρχεία (foreground και background) μέσω του αρχείου *dataset.py*. Επέλεξα να φορτώνω από το 1.5 δευτερόλεπτο μέχρι και το 3.5 δευτερόλεπτο για να ελαχιστοποιήσω όσο γίνεται το background noise στα ηχητικά ομιλίας (foreground).

Έπειτα, μέσω του αρχείου *feature_extraction.py* βρίσκω τα διανύσματα χαρακτηριστικών (feature vectors) που αντιστοιχούν σε κάθε ηχητικό αρχείο που φόρτωσα. Ο υπολογισμός αυτών γίνεται με τον υπολογισμό του mel-spectrogram φασματογραφήματος, μέσω την συνάρτηση *librosa.feature.melspectrogram* και μετατρέποντας τις τιμές του σε μονάδες decibel (dB). Έτσι, κάθε διάνυσμα χαρακτηριστικών θα είναι μια δισδιάστατή μήτρα, όπου η μια διάσταση θα παραπέμπει στα frames του ηχητικού και η άλλη στα χαρακτηριστικά του (Mel bands). Αξίζει να σημειωθεί ότι η συνάρτηση *librosa.feature.melspectrogram* υλοποιεί και την τεχνική κινούμενου παραθύρου, επομένως δεν χρειάστηκε να την υλοποιήσω.

Στην συνέχεια, αφού ολοκληρωθεί η εξαγωγή των χαρακτηριστικών διανυσμάτων όλων των ηχητικών εκπαίδευσης, σειρά έχει η δημιουργία των labels. Κάθε frame με τα χαρακτηριστικά του (Mels bands) πρέπει να αντιστοιχεί σε ένα label, 0 αν προέρχεται από το background dataset και 1 αν προέρχεται από το foreground dataset.

Μετά την δημιουργία των labels για κάθε ηχητικό, ακολουθεί μια προεπεξεργασία (pre-process) των διανυσμάτων χαρακτηριστικών και των labels όπου είναι απαραίτητο, για να χρησιμοποιηθούν στην συνέχεια για την εκπαίδευση των ταξινομητών.

Εκπαίδευση MLP (Multi-Layer Perceptron) ταξινομητή:

Το αρχείο *mlp_classifier.py* περιλαμβάνει τον κώδικα για την εκπαίδευση του MLP ταξινομητή. Για τον MLP ταξινομητή χρησιμοποιήθηκε η κλάση *MLPClassifier* από το πακέτο *sklearn.neural_network* της βιβλιοθήκης *scikit-learn*. Κατά την δημιουργία του ταξινομητή, όρισα 3 κρυφά επίπεδα με πλήθος νευρώνων 128, 64 και 32 αντίστοιχα. Επίσης, όρισα το 15% από το συνολικό πλήθος ηχητικών προς εκπαίδευση να χρησιμοποιηθεί για validation, ώστε να επιτύχω όσο το δυνατόν καλύτερη εκπαίδευση του μοντέλου.

Εκπαίδευση SVM (Support Vector Machine) ταξινομητή:

Το αρχείο *svm_classifier.py* περιλαμβάνει τον κώδικα για την εκπαίδευση του SVM ταξινομητή. Για τον SVM ταξινομητή χρησιμοποιήθηκε η κλάση *LinearSVC* από το πακέτο *sklearn.svm* της βιβλιοθήκης *scikit-learn*.

Εκπαίδευση LS (Least Squares) ταξινομητή:

Το αρχείο *lstsq_classifier.py* περιλαμβάνει τον κώδικα για την εκπαίδευση του LS ταξινομητή. Για τον LS ταξινομητή υλοποίησα την κλάση *LeastSquaresClassifier* όπου περιέχει την συνάρτηση train που είναι υπεύθυνη για την εκπαίδευση του μοντέλου, υπολογίζοντας τα βέλτιστα βάρη. Η συνάρτηση αυτή προσθέτει στα διανύσματα χαρακτηριστικών μια επιπλέον στήλη από άσσους, η οποία αντιστοιχεί στην τιμή απόκλισης (bias term – intercept) και έπειτα υπολογίζονται τα βάρη του ταξινομητή.

Εκπαίδευση RNN (Recurrent Neural Network) ταξινομητή:

Το αρχείο *rnn_classifier.py* περιλαμβάνει τον κώδικα για την εκπαίδευση του RNN ταξινομητή. Για τον RNN ταξινομητή χρησιμοποιήθηκε η κλάση *SimpleRNN* με 32 νευρώνες, καθώς και ακόμη ένα επίπεδο (layer) *απλού νευρωνικού δικτύου* ενός νευρώνα, με σιγμοειδή activation και binary cross-entropy loss function. Επίσης, όρισα το 15% από το συνολικό πλήθος ηχητικών προς εκπαίδευση να χρησιμοποιηθεί για validation, ώστε να επιτύχω όσο το δυνατόν καλύτερη εκπαίδευση του μοντέλου.

Αφότου τελειώσει εκπαίδευση του κάθε ταξινομητή, γίνεται αποθήκευση του στον φάκελο «classifiers», ώστε να μπορεί να χρησιμοποιηθεί μετέπειτα στην φάση ελέγχου.

Έλεγχος ταξινομητών

Το αρχείο *word_detector.py* είναι υπεύθυνο για τον έλεγχο των ταξινομητών. Στο αρχείο αυτό έχει υλοποιηθεί ένα command-line menu όπου δίνει τις εξής επιλογές στον χρήστη:

1. Επιλογή ηχητικών που θα χρησιμοποιηθούν για τον έλεγχο των ταξινομητών
2. Επιλογή επιθυμητών ταξινομητών προς έλεγχο
3. Αναπαραγωγή λέξεων που εντοπίστηκαν από τον ταξινομητή
4. Καθαρισμός κονσόλας
5. Τερματισμός προγράμματος

Εφόσον ο χρήστης επιλέξει τα ηχητικά και τους ταξινομητές που επιθυμεί, για κάθε ηχητικό εμφανίζεται το ποσοστό ακρίβειας του κάθε ταξινομητή. Το ποσοστό ακρίβειας βγαίνει συγκρίνοντας τα πραγματικά labels (ground truth labels) που αντιστοιχούν στο ηχητικό, με αυτά που έδωσε ως αποτέλεσμα ο ταξινομητής (predicted labels). Για τον υπολογισμό αυτού του ποσοστού χρησιμοποιείται η συνάρτηση *accuracy_score* του πακέτου *sklearn.metrics* της βιβλιοθήκης *scikit-learn*.

Παράδειγμα εκτέλεσης

Επιλογή ηχητικών και ταξινομητών

```
(venv) PS C:\Users\dimit\Desktop\p20004\source2024> py word_detector.py

=========== Word Detector Menu ===========

        1. Detect words in audio clips
        2. Clear console
        3. Exit

==========================================

> Enter your choice: 1

Available audio clips:
        1. test-1.mp3
        2. test-2.mp3
        3. test-3.mp3
        0. Back

> Enter the indices of the audio clips to detect words (separated by comma): 1,2,3

Choose the classifier/s to use:
        1. Least Squares (LS) classifier
        2. Multi-Layer Perceptron (MLP) classifier
        3. Recurrent Neural Network (RNN) classifier
        4. Support Vector Machine (SVM) classifier
        0. Back

> Enter the indices of the classifiers to use (separated by comma): 1,2,3,4
```

Προβολή ποσοστού ακρίβειας LS ταξινομητή και λέξεων που εντοπίστηκαν

```
For the following audio clips:
        + test-1.mp3
        + test-2.mp3
        + test-3.mp3

You have chosen the following classifiers:
        + Least Squares (LS) classifier
        + Multi-Layer Perceptron (MLP) classifier
        + Recurrent Neural Network (RNN) classifier
        + Support Vector Machine (SVM) classifier


========== Detecting words in test-1.mp3 ==========

+ Detecting words using the LS classifier...
Classifier detected 14 word/s.
Accuracy score: 0.20

Detected words' timestamps:
        1) 2.368 - 2.856 seconds
        2) 5.062 - 5.132 seconds
        3) 5.224 - 6.223 seconds
        4) 6.873 - 7.221 seconds
        5) 9.056 - 9.590 seconds
        6) 10.728 - 11.169 seconds
        7) 11.981 - 12.005 seconds
        8) 14.582 - 15.488 seconds
        9) 17.601 - 18.181 seconds
        10) 20.271 - 21.293 seconds
        11) 21.873 - 22.361 seconds
        12) 23.452 - 23.940 seconds
        13) 26.564 - 27.376 seconds
        14) 30.511 - 30.674 seconds

> Enter the index of the word to listen to it or any other key to continue:
```

Προβολή ποσοστού ακρίβειας MLP ταξινομητή και λέξεων που εντοπίστηκαν

```
Continuing execution...

+ Detecting words using the MLP classifier...
Classifier detected 74 word/s.
Accuracy score: 0.54

Detected words' timestamps:
        1) 0.000 - 0.279 seconds
        2) 0.418 - 0.697 seconds
        3) 0.789 - 0.859 seconds
        4) 0.929 - 1.022 seconds
        5) 1.347 - 1.440 seconds
        6) 1.556 - 1.625 seconds
        7) 1.741 - 1.974 seconds
        8) 2.067 - 2.252 seconds
        9) 2.415 - 2.624 seconds
        10) 2.810 - 3.135 seconds
        11) 3.274 - 3.622 seconds
        12) 3.646 - 3.669 seconds
        13) 3.715 - 3.762 seconds
        14) 3.785 - 3.924 seconds
        15) 4.063 - 4.272 seconds
        16) 4.667 - 4.807 seconds
        17) 4.969 - 5.085 seconds
        18) 5.341 - 5.573 seconds
        19) 5.596 - 5.642 seconds
        20) 5.689 - 6.060 seconds
        21) 6.130 - 6.409 seconds
        22) 6.571 - 6.803 seconds
        23) 7.268 - 7.639 seconds
        24) 7.709 - 7.895 seconds
        25) 8.127 - 8.591 seconds
        26) 8.754 - 9.288 seconds
        27) 9.706 - 9.845 seconds
        28) 9.915 - 10.333 seconds
        29) 10.542 - 10.797 seconds
        30) 10.820 - 10.844 seconds
        31) 10.867 - 10.890 seconds
        32) 11.122 - 11.447 seconds
        33) 11.540 - 11.656 seconds
        34) 11.796 - 11.865 seconds
        35) 12.051 - 12.307 seconds
        36) 12.469 - 12.585 seconds
        37) 12.794 - 13.189 seconds
        38) 13.514 - 13.723 seconds
        39) 13.816 - 14.048 seconds
        40) 14.164 - 14.466 seconds
        41) 14.675 - 15.209 seconds
        42) 15.232 - 15.348 seconds
        43) 15.441 - 15.697 seconds
```

Προβολή ποσοστού ακρίβειας RNN ταξινομητή και λέξεων που εντοπίστηκαν

```
Continuing execution...

+ Detecting words using the RNN classifier...
Classifier detected 55 word/s.
Accuracy score: 0.64

Detected words' timestamps:
        1) 0.023 - 0.279 seconds
        2) 0.418 - 0.511 seconds
        3) 0.604 - 0.697 seconds
        4) 0.789 - 1.045 seconds
        5) 1.324 - 1.440 seconds
        6) 1.533 - 1.672 seconds
        7) 1.811 - 1.950 seconds
        8) 2.067 - 2.252 seconds
        9) 2.879 - 3.111 seconds
        10) 3.274 - 3.413 seconds
        11) 3.483 - 3.622 seconds
        12) 3.692 - 3.924 seconds
        13) 3.994 - 4.296 seconds
        14) 4.342 - 4.551 seconds
        15) 4.644 - 4.807 seconds
        16) 4.969 - 5.062 seconds
        17) 6.223 - 6.432 seconds
        18) 6.594 - 6.711 seconds
        19) 7.221 - 7.941 seconds
        20) 8.220 - 8.615 seconds
        21) 8.754 - 9.056 seconds
        22) 9.590 - 9.729 seconds
        23) 9.892 - 10.426 seconds
        24) 10.542 - 10.728 seconds
        25) 11.169 - 11.656 seconds
        26) 11.703 - 11.842 seconds
        27) 12.051 - 12.283 seconds
        28) 12.446 - 12.585 seconds
        29) 12.701 - 13.212 seconds
        30) 13.398 - 13.723 seconds
        31) 13.816 - 14.048 seconds
        32) 14.164 - 14.489 seconds
        33) 15.488 - 15.697 seconds
        34) 15.790 - 16.184 seconds
        35) 16.417 - 16.997 seconds
        36) 17.183 - 17.601 seconds
        37) 18.228 - 18.483 seconds
        38) 18.553 - 19.551 seconds
        39) 19.644 - 19.923 seconds
        40) 20.108 - 20.271 seconds
        41) 21.293 - 21.478 seconds
        42) 21.618 - 21.873 seconds
        43) 22.361 - 22.570 seconds
        44) 22.663 - 22.825 seconds
```

Προβολή ποσοστού ακρίβειας SVM ταξινομητή και λέξεων που εντοπίστηκαν, καθώς και ένδειξη αναπαραγωγής κάποιων λέξεων που εντοπίστηκαν.

```
Continuing execution...

+ Detecting words using the SVM classifier...
Classifier detected 14 word/s.
Accuracy score: 0.20

Detected words' timestamps:
        1) 2.368 - 2.856 seconds
        2) 5.062 - 5.132 seconds
        3) 5.224 - 6.223 seconds
        4) 6.873 - 7.221 seconds
        5) 9.056 - 9.590 seconds
        6) 10.728 - 11.169 seconds
        7) 11.981 - 12.005 seconds
        8) 14.582 - 15.488 seconds
        9) 17.601 - 18.181 seconds
        10) 20.271 - 21.293 seconds
        11) 21.873 - 22.361 seconds
        12) 23.452 - 23.940 seconds
        13) 26.564 - 27.376 seconds
        14) 30.511 - 30.674 seconds

> Enter the index of the word to listen to it or any other key to continue: 1
Listening to the word 1 (0.488 seconds)...

> Enter the index of the word to listen to it or any other key to continue: 10
Listening to the word 10 (1.022 seconds)...

> Enter the index of the word to listen to it or any other key to continue: 13
Listening to the word 13 (0.813 seconds)...

> Enter the index of the word to listen to it or any other key to continue:
```

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. Όμως σε γενικές γραμμές, παρατήρησα γενικά ότι όλοι ταξινομητές δεν είναι τόσο αποτελεσματικοί όσο περίμενα. Αυτό μπορεί να οφείλεται στο μέγεθος των datasets που χρησιμοποίησα για την εκπαίδευση τους, καθώς και στις διάφορες παραμέτρους που δέχεται ο κάθε ταξινομητής τόσο κατά την δημιουργία όσο και κατά την εκπαίδευση του.

Πηγές

- *librosa library*
  Χρησιμοποιήθηκε για την εξαγωγή διανυσμάτων χαρακτηριστικών και την φόρτωση των ηχητικών.

- *scikit-learn library*
  Χρησιμοποιήθηκε για την δημιουργία των MLP και SVL ταξινομητών, καθώς και τον υπολογισμό της ακρίβειας των ταξινομητών.

- *TensorFlow library*
  Χρησιμοποιήθηκε για την δημιουργία, αποθήκευση και φόρτωση του RNN ταξινομητή.

- *Joblib library*
  Χρησιμοποιήθηκε για την αποθήκευση και φόρτωση των MLP, SVM και LS ταξινομητών.

- *vlc module*
  Χρησιμοποιήθηκε για την αναπαραγωγή των ηχητικών.

- *ElevenLabs Text To Speech With Timestamps API*
  Χρησιμοποιήθηκε για την δημιουργία ηχητικών και labels που χρησιμοποιούνται στην φάση ελέγχου των ταξινομητών.

- *Common Voice by Mozilla*
  Χρήση του «*Common Voice Corpus Delta Segment 17.0 (3/20/2024)*» ως foreground dataset για την εκπαίδευση των ταξινομητών.

- *Harvard Dataverse*|*github.com/karolpiczak/ESC-50*
  Χρήση του «ESC-50» ως background dataset για την εκπαίδευση των ταξινομητών.