Réseaux de neurones - Devoir Final

Orane Dufour

29 février 2024

Sorbonne Nouvelle !!!
université des cultures

Table des matières

1 Introduction

L'objectif de ce projet est de créer une API python afin de permettre l'évaluation de la robustesse d'un modèle de deep learning. Ce modèle est entraîné à détecter l'âge d'un locuteur sur une séquence audio de 2 à 10 secondes. Nous l'avons entraîné sur un corpus mais nous n'avons pas de données d'autres corpus qui permettraient de vérifier s'il est capable de généraliser. Ainsi, l'idée de notre API est d'évaluer sa robustesse en le testant sur des segments audios fournis directement par les utilisateurs de l'API.

2 Présentation du corpus d'entraînement

Le corpus ESLO2 [1] a été sélectionné car il présente un nombre conséquent de locuteurs avec une répartition équilibrée sur différentes classes d'âge, ainsi que de nombreuses métadonnées sur les locuteurs, dont l'âge. Le corpus s'accompagne d'une transcription orthographique effectuée sur Transcrier, segmentée en tours de parole et normalisée entre les différents enregistrements. Après pré-traitement, 35h d'enregistrement ont pu être extraites, pour 84 locuteurs répartis comme l'indique le tableau.1.

Age	Nombre de locuteurs	Durée
20-29	20	06 :36 :42
30-39	16	06 :10 :01
40-49	14	06 :19 :17
50-59	12	05 :29 :48
60-69	13	06 :07 :16
70-79	3	01 :15 :44
80-89	6	02 :37 :38

Table 1 – Corpus initial

2.1 Pré-traitements

Les enregistrements récupérés et leurs transcriptions ont subi plusieurs transformations afin de constituer un corpus exploitable pour nos besoins :

- Extraction du canal - Les enregistrements originaux étaient composées de deux pistes, vraisemblablement une première piste pour l'interviewer et une seconde pour la personne interviewée, sans que le choix du canal pour une des deux personnes ne soit figé. Nous avons donc extrait seulement un des deux canaux pour la suite de cette étude, en retenant le plus intense de deux.
- Extraction de la tier du locuteur - Les tours de parole du locuteur, auxquels une tier est dédiée dans les TextGrids, ainsi que les parties audio correspondantes, ont été extraites à l'aide d'un script Praat.
- Extraction des intervalles - Les transcriptions présentaient déjà une segmentation au niveau de la phrase. Ainsi ces intervalles ont été extraits à la fois pour l'audio et les TextGrids.
- Alignement - L'alignement entre l'audio et la transcription a été effectué à l'aide de l'outil MFA (Montreal Force Aligner), qui non seulement aligne les mots à l'audio mais aussi génère une transcription phonétique de ces derniers, également alignée à l'audio.

2.2 Évaluation de l'alignement

L'alignement, qu'il soit automatique ou manuel, est une tâche difficile dans les corpus de parole spontanée. L'extraction de features et de spectrogrammes dépend de l'alignement qu'il est donc crucial d'évaluer. De même, les analyses phonétiques que nous présenterons ci-dessous sont directement dépendantes de l'alignement phonétique. Ainsi, un échantillon de 50 TextGrids a été corrigé manuellement, afin de pouvoir calculer un accord inter-annotateurs. Ce n'est pas le type de phonème qui a été évalué mais la segmentation, en comparant les durées des deux annotations. Pour calculer le Kappa

1. http ://eslo.huma-num.fr/

de Cohen, la comparaison a été réduite à un problème de classification à deux classes : accord ou désaccord. Une tolérance de 10ms de différence entre la fin ou le début du segment a été appliquée. Le Kappa obtenu est de 0,68, ce qui correspond à un "accord fort".

2.3 Réduction du corpus et classification binaire

Après plusieurs essais infructueux de classification avec des CNN (spectrogrammes) et des MLP (vecteurs Wav2Vec), nous avons décidé de nous concentrer pour l'instant sur une classification binaire en -30 et +60, car les âges extrêmes sont perceptivement les plus simples à reconnaitre. Nous avons donc un corpuis réduit de 43 locuteurs au total (Table2).

Age	Sexe	Locuteurs	Séquences
+60	F	14	16198
-30	F	10	11329
+60	H	13	20646
-30	H	6	9490

TABLE 2 – Corpus Réduit

3 Expérimentations avec Wav2Vec

3.1 Récupération des vecteurs

Les modèles Wav2Vec "large" comportent 24 couches différentes, c'est-à-dire 24 niveaux de représentation du signal. Les résultats d'études visant à analyser les types d'informations contenus dans les couches montrent que les premières couches encodent des informations acoustiques et phonétiques, tandis que les dernières couches encodent des informations plutôt lexicales et sémantiques (citer article). Afin de sélectionner la meilleure couche pour notre tâche de classification, nous avons extrait les 24 représentations que propose Wav2Vec, puis nous avons testé pour chaque couche 2 modèles (séparation hommes et femmes), sur une tâche de classification binaire de l'âge. Nous avons effectué ces tests avec un corpus de 24 locuteurs, 12 pour les femmes et 12 pour les hommes ainsi qu'une répartition égale entre les âges (6 vieux 6 jeunes). Pour chaque couche, les modèles sont testés sur deux locuteurs, la configuration des ensembles d'entraînement et de test reste la même pour chaque couche. Comme l'illustre la Figure 1, la couche qui offre les meilleurs résultats est la première pour les femmes, avec un score de 99% de précision, et la sixième pour les hommes avec une précision de 97%

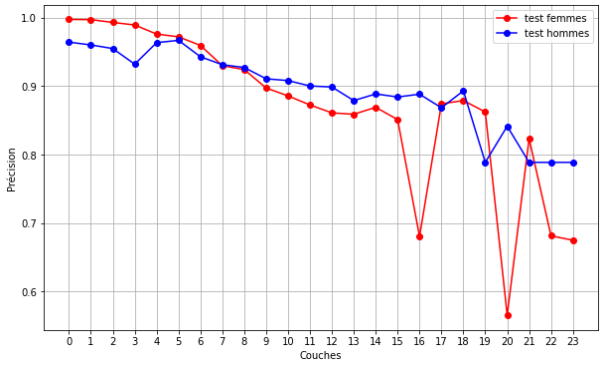

FIGURE 1 – Précision des 24 couches sur les corpus de test

Ces résultats montrent que les informations acoustiques et phonétiques, encodées dans les premières couches, sont plus utiles au modèle pour discriminer l'âge des locuteurs, alors que les informations lexicales et sémantiques des dernières couches sont moins utiles. L'analyse des erreurs sur les moins bonnes couches (20 et 21) corrobore cette hypothèse : les séquences mal classées contiennent enmoyenne

3 mots de moins que les séquences bien classées. De plus, 52% des séquences mal classées contiennent 5 mots ou moins contre 27% pour les séquences bien classées. Nous avons donc choisi de poursuivre nos expériences en utilisant la première couche pour les locuteurs de sexe féminin, et la sixième couche pour ceux de sexe masculin.

3.2 Entraînement des modèles

Nous avons donc entraîné deux modèles (Figure2), un pour les hommes et un pour les femmes, en utilisant les vecteurs correspondants. Afin de tester la robustesse des deux modèles et leur généralisation à dess locuteurs inconnus, nous avons pour chacun d'eux entraînés plusieurs modèles sur l'ensemble des locuteurs -1, que nous avons testés à chaque fois sur un seul locuteur inconnu. Nous obtenons ainsi un taux de classification par locuteur. Sur tous les modèles confondus (hommes et femmes), le taux de classification moyen par locuteur est de 94%. Seuls 2 locuteurs obtiennent un taux de bonne classification inférieur à 20%, une jeune femme et un jeune homme. Après vérification, nous pouvons imputer cela à la présence d'un bruit de fond persistant sur l'ensemble de l'enregistrement.

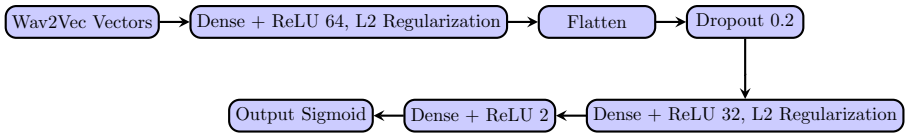

FIGURE 2 – Architecture des deux modèles perceptrons multicouches
Les modèles sont très simples mais nous avons dû veiller à les régulariser pour éviter le surapprentissage. L'accuracy était au début très haute sur le set de train (tâche trop facile car locuteurs connus), et stagnait sur le set de validation (1 seul locuteur innconnu)La diminution du nombre de neurones, les régularisations L2 sur les couches denses et la couche de dropout a permis de supprimer le problème de surapprentissage.

L'objectif de notre travail étant de pouvoir classer des séquences audios en dehors du corpus ESLO2, nous avons finalement enregistré un modèle pour chaque sexe entraîné sur l'ensemble du corpus. Comme nous n'avons pas de données hors ESLO2 pour le test, nous avons scindé le corpus pour le set de validation et de test grâce au module sickit-learn. Nous obtenons de très bons scores (99% pour les deux sexes). Cependant, la séparation aléatoire du corpus opérée avec sickit learn ne garantit pas qu'il y a exclusivement des locuteurs inconnus par le modèle dans le set de test et de validation. Ainsi ces deux modèles testés sur des locuteurs connus sont très susceptibles d'avoir reconnu le locuteur plutôt que l'âge. Seul un test avec des locuteurs hors ESLO2 permettrait de tester vraiment la robustesse de ces modèles. C'est l'ambition qui a motivé la mise en oeuvre de notre API.

4 API

4.1 FastAPI

Pour construire une interface utilisateur et la mettre en ligne, nous avons utilisé le module python FastAPI [2] qui permet de définir des fonctions asynchrones. Dans notre cas, les fonctions asynchrones transforment le fichier audio transmis par l'utilisateur en vecteur Wav2Vec, chargent nos modèles pré-entraînés en fonction du sexe du locuteur et renvoient une catégorie d'âge (-30 ou +60).

4.2 Mise en ligne

L'API fonctionne localement, cependant, le but était de le mettre en ligne afin de permettre l'accès distant à d'autres utilisateurs. Pour ce faire, nous avions besoin d'un serveur. Nous nous sommes donc tournés vers l'offre gratuit d'Amazon Web Service (AWS) [3]. Nous avons crée une instance Elastic Compute Cloud (EC2) qui permet de louer des machines virtuelles (instances) sur lesquelles nous pouvons exécuter nos propres applications. Malheureusement, le service gratuit d'AWS n'offre qu'un seul type d'instance EC2 (nano), qui n'est pas adaptée au deep learning, et qui a un espace de stockage

2. https ://fastapi.tiangolo.com/
3. https ://aws.amazon.com/fr/

limité (30GO maximum). Nous avons tout de même importer notre projet sur notre serveur, en clonant le dépôt GitHub de ce dernier, et nous avons installé tous les modules nécessaires au projet. Nous avons rencontré des problèmes pour installer tensorflow et pytorch sur le serveur, qui manquait de mémoire vive, mais nous avons réussi en n'installant que les versions cpu des deux modules. La commande *uvicorn main :app –host 0.0.0.0 –port 8000* permet de lancer notre application FastAPI en indiquant à Uvicorn d'accepter les connexions sur toutes les interfaces réseaux disponibles, sur le port 8000. En ajoutant l'adresse publique du serveur, nous obtenons l'URL du site où est déployée notre API : `http://13.53.175.163:8000/`. Tout fonctionne bel et bien cependant le temps de calcul est très long, le serveur envoie sa réponse en environ 5 minutes alors que c'est presque instantané en utilisation locale. Nous ne pouvons malhheureusement pas modifier notre instance EC2 avec l'offre gratuite d'AWS. Le site reste accessible (nous utilisons tmux) et nous le fermerons après correction du projet.

5 Discussion

Nous voulions créer une API en ligne afin de tester la robustesse de nos modèles sans avoir à télécharger un nouveau corpus. Même si nous avons réussi à déployer notre API en ligne, nos ressources ne nous permettent pas d'utiliser un serveur assez optimisé pour notre projet. Les résultats du modèle sont affichés sur la page d'accueil du site en temps réel. Il semble que la généralisation à des données hors ESLO se fasse assez bien, cependant nous doutons de pouvoir récolter assez de tests pour rendre cette vérification stable. D'une part car peu de gens utiliserons l'API finalement et d'autre part car le serveur et très lent et une utilisation simultanée de l'API n'est sûrement pas permise.

Références

[PCL22] Ankita Pasad, Ju-Chieh Chou, and Karen Livescu. Layer-wise analysis of a self-supervised speech representation model, 2022.

[TMCK21] Anvarjon Tursunov, Mustaqeem, Joon Yeon Choeh, and Soonil Kwon. Age and gender recognition using a convolutional neural network with a specially designed multi-attention module through speech spectrograms. *Sensors*, 21(17), 2021.

Le dépôt GitHub pour ce travail est disponible à cette adresse : `https://github.com/OraneD/AgePredict` Le corpus et les vecteurs générés pour l'entraînement des modèles n'ont pas pu être exportés sur le Git en raison de leur trop grande taille.