Rapport de projet : Environnement de télémédecine

Alice GUILLOTEAU

Notice utilisateur (version longue)

Les fichiers correspondant à la première partie du projet, c'est-à-dire au pré-traitement et à l'envoi des données sont disponibles dans le dossier Partie 1.

L'appel de l'ensemble des fonctions correspondant à la partie 1 pour mes fichiers est disponible dans le fichier « part1.py » (en commentaire car une partie doit être exécutée sur Matlab).

Les fichiers txt bruts sont à déposer dans le dossier data_txt. Le nom du fichier doit être composé de l'identifiant étudiant de l'élève ainsi que de la séquence, suivant ce format : « 95476D_S1 »

La première étape de pré-traitement consiste à convertir le fichier txt brut provenant de biosignalplus en un fichier csv exploitable. Pour cela, il faut appeler la fonction txt2csv, présente dans le fichier du même nom. Celle-ci prend en paramètre le nom d'un fichier txt présent dans le dossier data_txt et et crée un fichier csv convertit dans le dossier data_csv.

La seconde étape consiste à effectuer un découpage par séquence ainsi qu'un filtrage des signaux. Il faut appeler la fonction dataPreProcessing sur Matlab en indiquant en paramètre le nom du fichier brut (sans l'extension), la date, et la liste contenant les points de découpage de la séquence. Ex : dataPreProcessing(« 95476D_S1 », »2023-09-22_10-39-22 »,[2655, 5019, 7384])

La troisième étape consiste à convertir les fichiers csv obtenus après traitement en fichiers json. Il faut pour cela appeler la fonction dataSequence2jsonFormat dans le fichier data2json en précisant le nom du fichier à convertir ainsi que les paramètres nécessaires. (exemples disponibles sur part_1.py)

Enfin, la dernière étape consiste à envoyer le fichier json sur le serveur grâce à la requête POST. Il faut pour cela faire appel à la fonction postJson du fichier getLib.py. La fonction postDossier est également disponible pour envoyer tous les fichiers json d'un dossier.

Les fichiers correspondant à la seconde partie, c'est-à-dire à la récupération, au post-traitement et à l'affichage des données sont disponibles dans le dossier Partie 2.

Les fonctions liées à la partie 2 sont appelées par l'interface utilisateur. Pour lancer l'interface, il faut se positionner dans le dossier « Partie 2 » et appeler la commande « flask run » dans un terminal python. Il faut ensuite coller le lien affiché par le terminal dans un navigateur.

Méthodologie

Partie 1 :

J'ai choisi de procéder au pré-traitement de mes données de manière partiellement manuelle, au vu du faible nombre de séquence liée aux types de protocoles que j'ai effectué durant l'acquisition. Ainsi, j'ai choisi mes points de découpage du signal en différentes séquences en les observant sur biosignalplus.

Le découpage ainsi que le filtrage des signaux sont réalisés par deux fonctions Matlab (dataPreProcessing, qui appelle la fonction filterData), conçues en partie avec l'aide de chatGPT.

Pour le filtrage des signaux respiratoire, j'ai choisi d'appliquer un filtre passe bas de type butterworth afin d'éliminer en partie le bruit. En effet, celui-ci est présent dans des fréquences plus hautes que celles correspondant à la respiration. J'ai choisi, après divers essais (les graphiques sont en commentaire dans le code), un ordre de 4 et une fréquence de coupure de 2, qui donnaient les résultats les plus concluants.

J'ai également appliqué une moyenne glissante aux signaux d'accéléromètre, selon les indications que nous avions reçu en classe.

Les différentes fonctions de ces parties appliquent des transformations successives aux fichiers, qui sont enregistrés dans les dossiers correspondants (data_csv_raw, data_csv_processed etc). Ainsi, il est possible d'observer l'état des fichiers après les différentes étapes ce qui m'a été utile durant ma phase de test.

Le fichier part_1.py contient l'ensemble de l'appel des fonctions que j'ai dû réaliser pour traiter et envoyer mes données. Le fichier ne peut cependant pas être exécuté en une fois (il faut décommenter les parties), car une partie du traitement a lieu sur Matlab. J'ai tenté d'installer la Matlab Engine API afin de pouvoir appeler une fonction Matlab depuis mon script python, mais je n'ai pas réussi à faire fonctionner le package et ai donc laissé ce code en commentaire.

Partie 2 :

J'ai choisi, pour l'interface utilisateur, de créer une application web en utilisant le framework Flask. Celle-ci fonctionne grâce à mon fichier python app.py, qui appelle les fonctions nécessaires, et utilise comme template mon fichier index.html présent dans le dossier templates.

Les fonctions de récupération de données et de post-traitement sont disponibles dans les fichiers getLib et post_processing. J'ai utilisé ici chatGpt pour écrire les codes de construction des graphiques et également pour extraire les informations du json récupérées depuis le serveur. Le fichier index.html a également été écrit en partie avec chatGPT.

L'interface web est composée d'un formulaire à remplir pour récupérer une séquence, ainsi que de 4 onglets affichant des informations sur la séquence et un onglet à propos. Le fichier app.py fait appel aux fonctions créant les dataframes contenant les statistiques ainsi que des graphiques qui sont stockés dans le dossier « static ».

Le premier onglet « overview » affiche les 4 signaux de la séquence ayant subis une fonction de transfert ainsi que les informations générales sur la séquence.

Les 2ème et 3ème onglet correspondent à la respiration thoracique et abdominale. J'ai choisi, pour traiter ces signaux et générer des graphiques d'utiliser la librairie neurokit2 : https://neuropsychology.github.io/NeuroKit/

Cette librairie comporte de nombreuses fonctions utiles au traitement de signaux physiologiques. J'ai utilisé ici la fonction rsp, plus précisément la fonction rsp_process() pour les signaux respiratoires. Cette fonction applique un nettoyage supplémentaire au signal et détecte les cycles respiratoires. Elle génère également un graphique représentant la fréquence respiratoire, l'amplitude, le volume respiratoire et la symétrie du cycle en fonction du temps. (se référer à la documentation neurokit2 pour plus d'information sur son fonctionnement)

Le 4ème onglet contient les signaux d'accéléromètre convertis en m.s2 ainsi que la moyenne et l'écart type des signaux. J'ai tenté d'effectuer d'autres traitements, cependant j'obtenais des résultats incohérents sur mes signaux ainsi que ceux des autres élèves, peut être dus à une mauvaise calibration du dispositif de mesure.

Contexte applicatif

Cette application a pour but d'étudier la fonction respiratoire et ses changements dans diverses conditions (montée ou descente d'escalier, repos, marche).

Elle apporte à l'utilisateur diverses informations sur une séquence respiratoire (fréquence respiratoire, amplitude, volume, symétrie du cycle) ainsi que son contexte (type d'activité réalisé et accéléromètre). Ces informations peuvent avoir un intérêt à titre médical, car elles peuvent être indicateurs d'anomalies ou de particularité dans la fonction respiratoire de l'individu.

Cette application peut également avoir un intérêt à titre de recherche, notamment en comparant les signaux de divers étudiants (à condition d'avoir recueilli leur consentement au préalable.)

Bilan critique

Lorsque j'ai commencé ce projet, je n'avais pas encore une vision d'ensemble de celui-ci. J'ai réalisé une partie des pré-traitements sur Matlab ce qui a compliqué l'intégration de cette partie avec le reste de mon code en python.

L'ensemble de mon code a été encapsulé dans des fonctions adaptées ce qui permet de séparer les différents traitements et étapes du projet. Les différents fichiers créés par les fonctions (images, csv, json etc) sont également stockés dans des dossiers adaptés.

L'utilisation de la librairie NeuroKit 2 m'a permis de gagner du temps lors du post-traitement des données respiratoires et également d'afficher des indicateurs supplémentaires que je n'aurais pas forcément été capable de calculer moi-même par manque d'expertise sur le sujet.

L'authentification pour récupérer les données depuis le serveur a lieu à l'intérieur de mon code et utilise donc par défaut mes identifiants. Dans le cas d'un déploiement de l'application il serait nécessaire de demander une authentification dans l'interface utilisateur.

Au niveau fonctionnel, la première partie du projet est en majorité automatique mais nécessite tout de même des actions de la part de l'utilisateur : donner un nom adapté au fichier brut, trouver les points de coupure de la séquence et exécuter les fonctions python et Matlab avec les paramètres corrects.

La seconde partie du projet est totalement automatisée grâce à l'interface utilisateur et ne nécessite pas l'appel de fonction dans un terminal ou dans un fichier. Elle nécessite en revanche de démarrer l'application flask depuis un terminal.

Celle-ci présente cependant des limites concernant la récupération de séquences d'autres étudiants. J'ai en effet principalement basé mon application sur mes propres données (identifiant E9), et celle-ci peut ne pas fonctionner correctement lors de la récupération d'autres données. Cela peut être dû à des données manquantes, des séquences trop courtes ne permettant pas une analyse du cycle respiratoire, certains post-traitements réalisés avant l'envoi etc. Certaines erreurs sont affichées par l'application, mais celle-ci ne gère pas toutes les exceptions possibles et il se peut donc qu'elles provoquent une internal serveur error (il suffit pour cela de revenir en arrière).

Enfin, le nombre de fréquences respiratoire thoracique et abdominale peuvent parfois être différents pour une même séquence. Cela peut être liée à la qualité des signaux dues à l'acquisition (ceinture plus ou moins serrée) ou au bruit qui n'a pas pu être totalement éliminé.