Rapport PAN 4 module Test et Intégration

Clément Bernard et Thiziri Nait Saada : PACT 3.5

May 16, 2019

Module Base de données

La base de données nous envoie, grâce à un lien spécifique, une page web contenant du texte. C'est ensuite à nous de récupérer le texte pour en déduire les données utiles.
De ce fait, tout comme le module Android a fait, nous récupérons toutes les données sur les livres et les données sur les auteurs. Ensuite, nous examinons chaque donnée pour tester si elle est valide ou non.
Pour ce faire, nous avons considéré qu'une donnée est fausse pour les livres si la date ne contenait pas 5 chiffres au minimum, ou si l'auteur/titre/lien epub/lien texte était nul.
Nous avons testé pour chaque livre, en les regroupant par première lettre. Nous avons tracé les graphes (cf fin du document) suivant si les livres étaient totalement correctes (aucune donnée fausse) et aussi en fonction des données (graphe suivant si le nom est erroné, l'auteur, etc).
Les codes associés à ces tests se trouvent sur le dépôt git :
pact35/modules/TestIntegration/src/PythonTest/test.py
Ensuite, nous avons testé l'envoie par la base de données des textes après traitement. Nous avons récupéré le nombre d'ambiances par texte pour en faire une moyenne. Ce code a été fait à partir du travail fait en Android. De ce fait, nous avons utilisé Java. Les codes se trouvent sur le dépôt git :
pact35/modules/TestIntegration/src/JavaTest/src/Main.java
Pour l'exemple de tous les livres commençant par la lettre "a", nous avons récupérer toutes les ambiances pour en faire une moyenne : nous en avons compté **43201** au total pour **5435** livres, soit un total de **7.94** ambiances par livre.
Au vu de la durée d'acquisition de ces ambiances (cela a mis près de 6 heures afin de récupérer ces ambiances pour les livres commençant par la lettre "a"), nous n'avons pas pris la peine de regarder pour les autres lettres.

Module Communication Client Serveur

Pour ces tests, nous avons récupérer le fichier python de notre collègue du module communication client serveur. Ce serveur simule une connection à la Raspberry Pi.
Nous avons ensuite créé une boucle "for" où nous simulons la connection à la Raspberry Pi et la diffusion d'une odeur (de manière aléatoire). Nous avons récolté les temps moyens. Voici le lien vers le code :
pact35/modules/TestIntegration/src/JavaTest/src/Tests.java
Les résultats obtenus sont :
Temps moyen de connexion au serveur : 0.61 secondes
Temps moyen de diffusion d'une odeur : 0.27 secondes
Bien évidemment le résultat ne sera pas aussi optimale car la diffusion d'une odeur est plus longue avec le vrai dispositif : l'action du moteur sur les diffuseurs prend un temps non négligeable, pas pris en compte ici.

Module Android

Pour ces tests, nous avons implémenter cela directement dans les codes du module.
Pour ce faire, nous avons testé le temps de récupération et d'affichage des données de la base de données. Dans la page "Library" qui charge les données des livres et des auteurs, nous avons fait une boucle "for" qui recharge les données. Nous récoltons ensuite le temps effectué pour en déduire la moyenne de chargement des livres.
Ensuite, nous avons télécharger le même livre 40 fois pour estimer de même le temps moyen de téléchargement d'un livre aléatoire.
Voici les résultats : **Temps de rechargement moyen des livres (pour 40 essais) : 1,145 secondes**
(le code est disponible dans : pact35/modules/Android/src/ReadingMood/app/src/main/java/com/example/clement/readingmood/Fragments/FragmentLibrary/FragmentLibraryLetter.java)
Temps de rechargement moyen des auteurs : 40/10000 secondes (la précision de l'horloge Android était trop faible pour indiquer le temps exacte de récupération des données) (le code est disponible dans : **pact35/modules/Android/src/ReadingMood/app/src/main/java/com/example/clement/readingmood/Fragments/FragmentLibrary/AuthorFragmentList.java**).
Temps de téléchargement moyen d'un livre aléatoire : 6 secondes
(lien code : **pact35/modules/Android/src/ReadingMood/app/src/main/java/com/example/clement/readingmood/Pages/Library**

Figure 1: Exemple de test de données des livres commençant par la lettre "e"

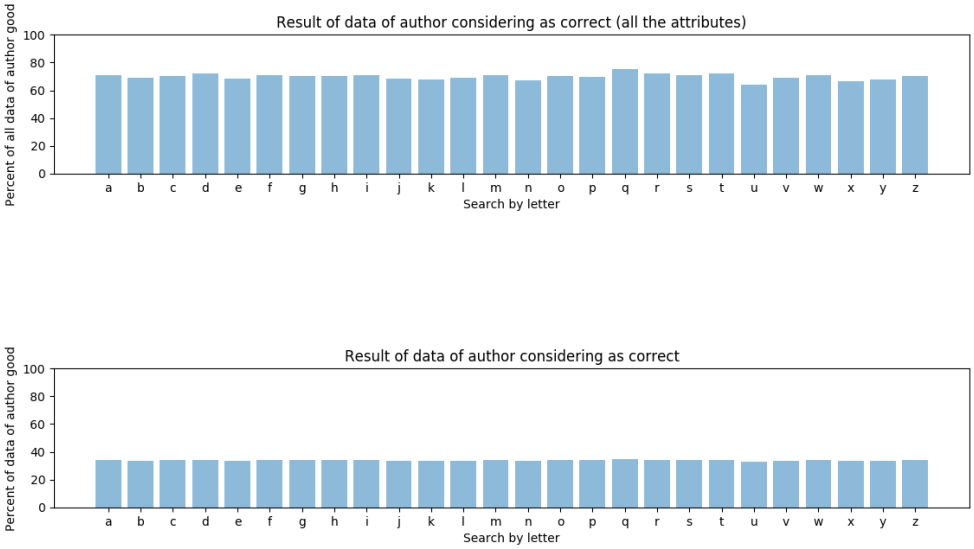
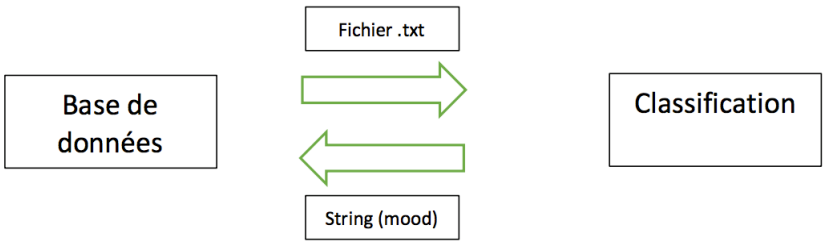


Figure 2: Test des données des auteurs par début de lettre

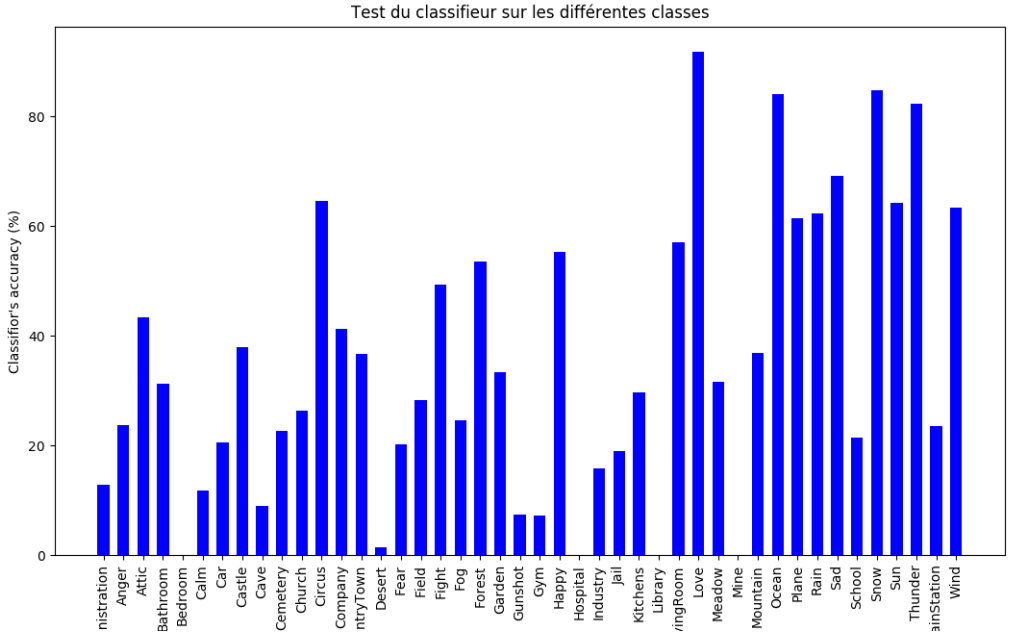
Figure 3: Test des données des livres par début de lettre

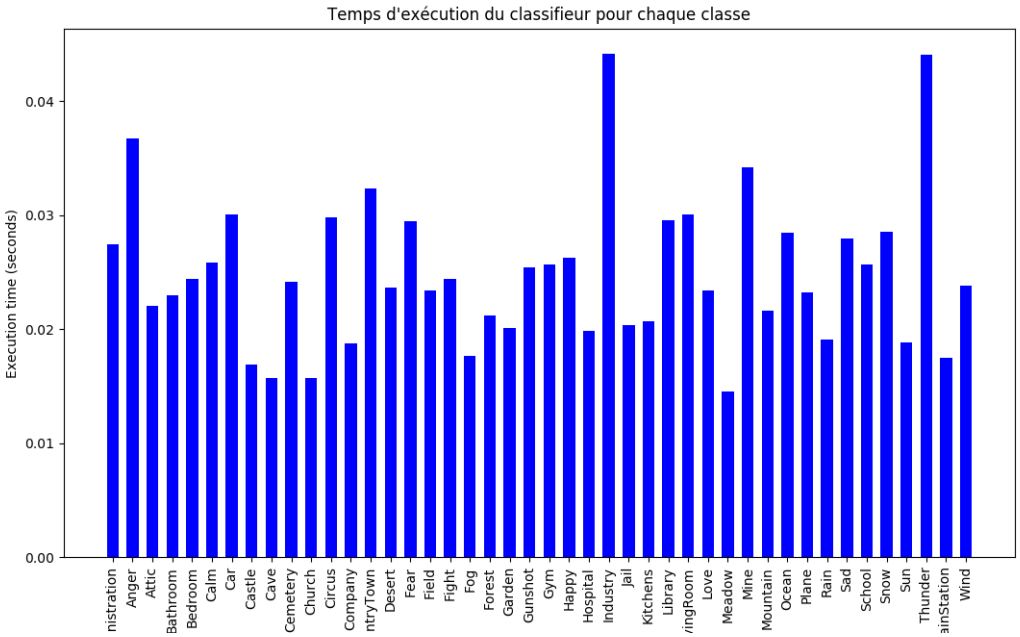
Classification

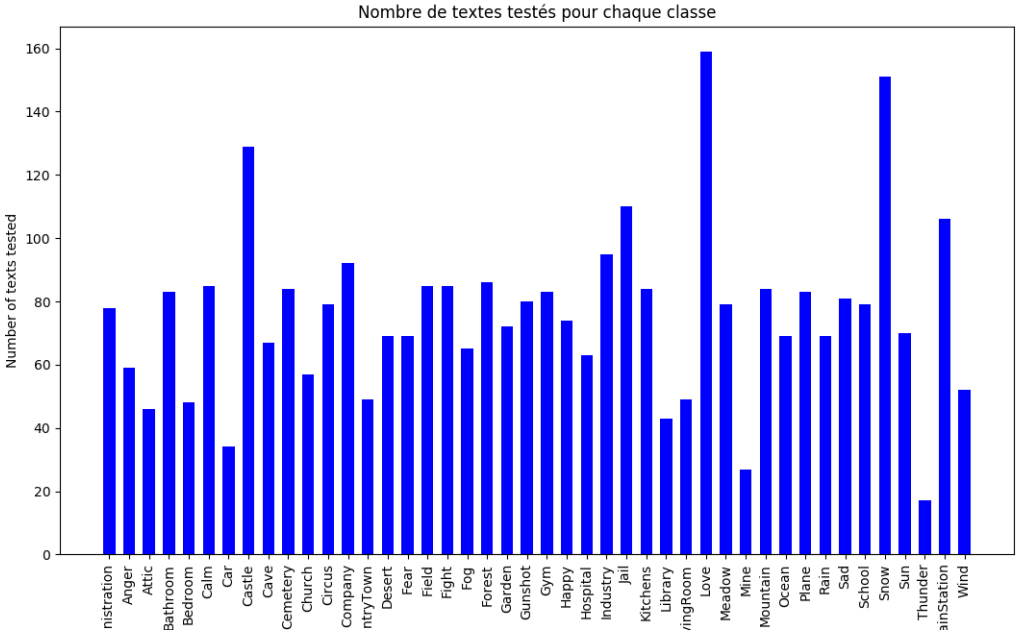
Le module Base de données utilise la fonction de classification développée dans le module Classification. Autrement dit, la base de données donne en entrée un fichier .txt à traiter, issu du découpage préalable du livre téléchargé par l'utilisateur, au classifieur. Celui-ci renvoie en sortie un string qui désigne l'une des 48 classes (ou ambiances) sur lesquelles il a été entrainé.
On peut par ailleurs noter que le découpage préalable en fichiers .txt se fait selon les paragraphes des pages HTML du fichier epub associé, afin de faciliter l'affichage de ces fichiers epub sur l'interface graphique développée par le module Android.
Enfin, pour que la base de données puisse utiliser la fonction, il a fallu sauvegarder et lire les données d'entrainement du classifieur, ainsi que toutes les fonctions permettant le traitement du fichier .txt dans des fichiers binaires.



Nous avons effectué une batterie de tests concernant la qualité du classifieur développé par le module Classification, dont vous pourrez retrouver les fonctions sur GitLab à l'adresse Pact35/modules/TestIntegration/src/PythonTest/TestMoodFinal.py.
Pour cela, nous avons testé la fonction sur près de 4000 fichiers .txt dont nous connaissions leur "mood" associé et avons incrémenté notre score d'une unité à chaque fois que le classifieur renvoyait un string correspondant exactement à son " mood " préalablement défini. Nous avons également enregistré les temps d'exécution du classifieur ainsi que le nombre de textes testés, et ce pour chaque classe. En voici les résultats. En moyenne, sur toutes les classes, nous avons une classification qui avoisine les **35,505** pour cent d'accuracy (avec un critère de réussite très strict à savoir l'ambiance retournée correspond-elle oui ou non à celle pensée au préalable).


Test du classifieur sur les différentes classes
Classifior's accuracy (%)
0
20
40
60
80
nistration
Anger
Attic
Bathroom
Bedroom
Calm
Car
Castle
Cave
Cemetery
Church
Circus
Company
ntryTown
Desert
Fear
Field
Fight
Fog
Forest
Garden
Gunshot
Gym
Happy
Hospital
Industry
Jail
Kitchens
Library
ingRoom
Love
Meadow
Mine
Mountain
Ocean
Plane
Rain
Sad
School
Snow
Sun
Thunder
ainStation
Wind


Temps d'exécution du classifieur pour chaque classe
Execution time (seconds)
0.00
0.01
0.02
0.03
0.04
nistration
Anger
Attic
Bathroom
Bedroom
Calm
Car
Castle
Cave
Cemetery
Church
Circus
Company
ntryTown
Desert
Fear
Field
Fight
Fog
Forest
Garden
Gunshot
Gym
Happy
Hospital
Industry
Jail
Kitchens
Library
ingRoom
Love
Meadow
Mine
Mountain
Ocean
Plane
Rain
Sad
School
Snow
Sun
Thunder
ainStation
Wind


Nombre de textes testés pour chaque classe
Number of texts tested
0
20
40
60
80
100
120
140
160
nistration
Anger
Attic
Bathroom
Bedroom
Calm
Car
Castle
Cave
Cemetery
Church
Circus
Company
ntryTown
Desert
Fear
Field
Fight
Fog
Forest
Garden
Gunshot
Gym
Happy
Hospital
Industry
Jail
Kitchens
Library
ingRoom
Love
Meadow
Mine
Mountain
Ocean
Plane
Rain
Sad
School
Snow
Sun
Thunder
ainStation
Wind