Guide d'utilisation de l'implémentation du serveur NISQA pour Docker

Auteurs :
Séverin CHEVALIER
Liam COURSODON
Jules DECAESTECKER
Zaccarie KANIT
Clémence MARINIER
Léo STENGEL
Elias TRANCHANT

Décembre 2022

1 Présentation

Ce projet a été réalisé par 7 étudiants d'IMT Atlantique Brest avec Vincent BARRIAC, Yvain JORIGNY et Nicolas PENNANEACH de Orange (Innovations/Networks). NiSQA est un algorithme évaluant la qualité d'une communication audio développé par des chercheurs de l'Université Technique de Berlin. Ce projet visait à remplacer l'algorithme PESQ anciennement utilisé. Ce guide d'utilisation détaille l'architecture et le fonctionnement du service docker et complète le readme associé à ce projet.

2 Contenu de l'archive

Dans cette archive se trouve plusieurs fichier :

- README.md : description du répertoire et consignes d'utilisation
- Dockerfile : dockerfile utilisé pour la construction de l'image docker
- .dockerignore : fichier permettant au Dockerfile de ne pas copier tout le répertoire dans l'image
- requirements.txt : fichier contenant les dépendances python à installer sur le docker
- nisqa.deb : logiciel Debian faisant tourner un serveur et l'algorithme NiSQA. Ce logiciel est installé dans le docker
- launch.sh : script bash allumant le serveur NiSQA dans le docker et le faisant tourner en continu
- dg105.wav : fichier audio servant aux tests
- test.sh : script bash effectuant un test du bon fonctionnement du logiciel debian en utilisant l'audio dg105.wav

3 Installation

En se plaçant dans le répertoire du projet :

- Pour crer une image dockerdu nom de nisqa:

```
sudo docker build . -t=nisqa
```

- Pour lancer un conteneur docker du nom de nisqa en utilisant l'image nisqa avec une redirection des ports 8080 :

```
sudo docker run -d -p 8080:8080 --name nisqa nisqa
```

4 Utilisation

4.1 Lancer / Arrêter le conteneur nisqa

```
sudo docker start nisqa
sudo docker stop nisqa
```

4.2 Lancement d'une mesure

- Pour utiliser avec un fichier .wav

```
curl -H "Content-type: audio/vnd.wave" --data-binary
@- http://127.0.0.1:8080/cgi-bin/nisqa.cgi < audio.wav
```

- Pour utiliser avec .a8k échantillonné à 8khz
- Test avec test.sh

```
./test.sh 127.0.0.1
```

Ce script fait le test avec le fichier dg105.wav et compare le résultat obtenu avec le résultat qu'on obtient normalement si on applique directement le modèle NiSQA au fichier audio (écrit à la main dans le script). Comme l'algorithme NiSQA est du machine learning, les décimales des résultats peuvent différer selon les machines. C'est pourquoi on ne compare que les 3 premiers chiffres après la virgule.

4.3 Résultats attendus

Le résultat de la commande curl précedente est de la forme :

```
[MOS, Noisiness, Coloration, Discontinuity, Loudness]
```

Chaque note est comprise entre 1 et 5. On peut avoir par exemple :

```
[1.495162,2.407247,2.001792,2.026033,2.895231]
```

5 Fonctionnement et Maintenance

5.1 Cadre d'utilisation du serveur NiSQA pour Debian

Ce conteneur docker vise à être déployé sur des robots constitué de plusieurs images docker. Le schéma ci-dessous explique l'interaction des robots sipcan (robots qui envoient les audio sur le réseau et simulent une communication VoIP) avec ce conteneur contenant NiSQA.

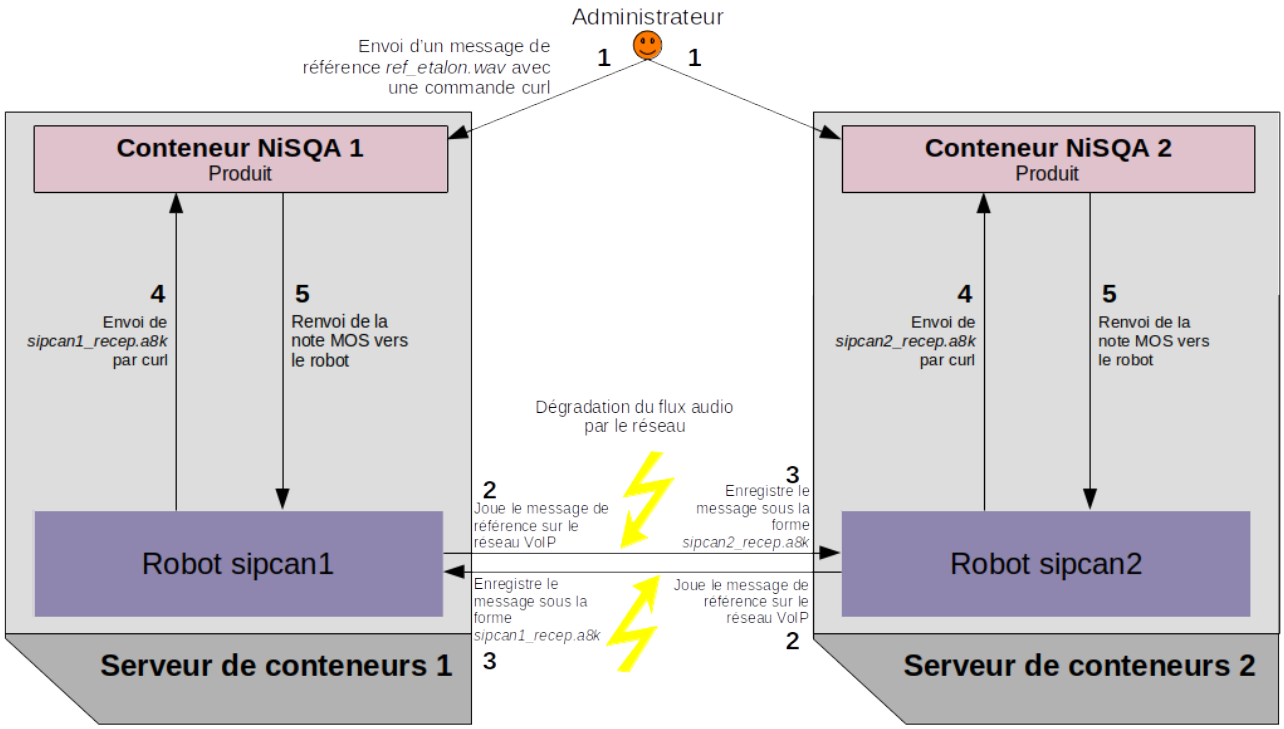

Figure 1: Périmètre de notre produit dans le cas d'un serveur décentralisé

5.2 Algorithme Nisqa

Le modèle d'évaluation de la qualité audio que nous utilisons est nommé NiSQA pour Non-intrusive Speech Quality Assessment. C'est un réseau de neurones convolutif basé donc sur des méthodes de deep-learning codé en python. Il a été entraîné sur différents ensembles de données et est utilisable tel quel.
Pour l'instant, l'algorithme fonctionne sans fichier de référence. Cela signifie que le fichier audio dégradé après transmission n'est pas comparé avec celui enregistré avant la transmission. Les sources sont hébergées sur le dépôt Github de G. Mittag : https://github.com/gabrielmittag/NISQA

5.3 Architecture du Docker

Cette partie décrit les différents composants de ce service pour que par la suite, si une nouvelle version du logiciel NiSQA est publié ou si Orange décide de changer d'algorithme d'évaluation, la modification puisse se faire sans problème.

5.3.1 Construction de l'image docker

Le construction de l'image se fait en 2 étapes :

- installation des dépendances python (environ 1Go à cause de la taille de pytorch) dans un environnement virtuel
- Copie de l'environnement virtuel puis installation du logiciel debian (et des dépendances debian d'environ 50Mo)

5.3.2 Le logiciel debian

La création de nisqa.deb (logiciel debian) n'est pas développé ici. C'est ce logiciel qu'il faudra modifier si l'algorithme vient à changer. Un autre projet (et un autre user guide) a été créé à cet effet.