

Rapport 7 Mai

Objectifs

- Inférence avec FastSpeech2+MultiSinger
- Etude de speechbrain et explorer la possibilité de train fastspeech2 sur nos anciennes datas
- Incorporer un modèle G2p (anglais) à la fonction alignement de MLPSinger
- Etudier les G2p en général

Mes études

Etude 1: Prise en main de speechbrain

SpeechBrain: Open-Source Conversational AI for Everyone

Open, simple, flexible, well-documented, and with competitive performance.

 <https://speechbrain.github.io/>




SpeechBrain

SpeechBrain est un toolkit de traitement de la parole open-source et tout-en-un qui s'appuie sur PyTorch.

L'objectif est de créer un toolkit unique, flexible et convivial qui peut être utilisé pour développer facilement des technologies vocales de pointe, y compris des systèmes de reconnaissance vocale (**end-to-end et HMM-DNN**), de reconnaissance du locuteur, de séparation vocale, de traitement de signal multi-microphone (**par exemple, la formation de faisceaux**), d'apprentissage supervisé et non supervisé, de contamination / augmentation vocale, et beaucoup plus.

Speech Synthese

Google Colab

 <https://colab.research.google.com/drive/1QLKtArv0PhJ8g4cfJGpoaGqDHxzup7KB#scrollTo=61faed93>



On voit qu'avec Speechbrain on accède à une panoplie de modèles acoustiques et de vocodeurs comme FastSpeech2, Tacotron

speechbrain (SpeechBrain)
Deep Learning, Speech Technologies

🤖 <https://huggingface.co/speechbrain>


speechbrain

🤖 huggingface.co/speechbrain

1. Ce qu'on peut faire avec

J'ai exploré à travers ce notebook les divers tâches réalisable avec Speechbrain et ainsi qu'un aperçu des différents langues prises en compte anglais, français, le mandarin aussi

- ASR

```
[4] from speechbrain.inference.ASR import EncoderDecoderASR

asr_model = EncoderDecoderASR.from_hparams(source="speechbrain/asr-crdnn-commonv
asr_model.transcribe_file("/content/example-fr.wav")

hyperparams.yaml: 100% 3.90k/3.90k [00:00<00:00, 151kB/
normalizer.ckpt: 100% 1.78k/1.78k [00:00<00:00, 15.6kB/
asr.ckpt: 100% 593M/593M [00:04<00:00, 83.8MB/
tokenizer.ckpt: 100% 245k/245k [00:00<00:00, 3.80MB/
```

- Speaker separation

Speech Separation

We here show a mixture with 2 speakers, but we have a state-of-the-art system for separating mixture with 3 speakers as well. We also have models that deals with noise and reverberation. [See your HuggingFace repository](#)

```
from speechbrain.inference.separation import SepformerSeparation as separator

model = separator.from_hparams(source="speechbrain/sepformer-wsj02mix", savedir=
est_sources = model.separate_file(path="/content/test_mixture.wav")

hyperparams.yaml: 100% 1.51k/1.51k [00:00<00:00, 83.6kB/
masknet.ckpt: 100% 113M/113M [00:00<00:00, 141MB/
encoder.ckpt: 100% 17.3k/17.3k [00:00<00:00, 417kB/
decoder.ckpt: 100% 17.2k/17.2k [00:00<00:00, 503kB/

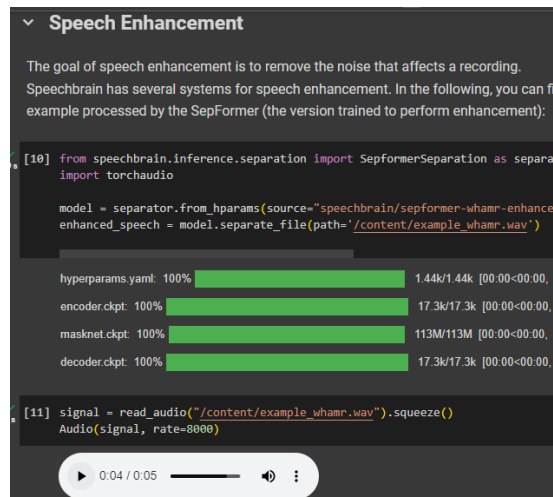
[7] signal = read_audio("/content/test_mixture.wav").squeeze()
Audio(signal, rate=8000)
```



On mets en entrée un audio où trois personnes parlent à la fois et on arrive à extraire un audio pour chacune des voix

- Speech enhancement

On prend un audio avec des bruits des effets, et le modèle enlève ses bruits et le son est net



- TTS

Nous sommes allés à code source nous avons déterminé les modèles TTS qui sont pris en compte

- FastSpeech2
- FastSpeech2Alignment
- Tacotron
- MsTacotron

2. Le Brain, et mise en place d'une boucle d'apprentissage

Logique

La méthode principale de la classe Brain est `fit()` responsable de l'itération dans l'ensemble de données, de la mise à jour du modèle et de la gestion de la boucle d'entraînement. Pour exploiter `fit()`, au moins deux méthodes doivent être définies dans la sous-classe : `compute_forward()` et `compute_objectives()`. Ces méthodes gèrent le calcul du modèle pour générer des prédictions et le calcul des termes de perte requis pour le calcul du gradient.

```
import torch
import speechbrain as sb

class SimpleBrain(sb.Brain):
    def compute_forward(self, batch, stage):
        return self.modules.model(batch["input"])

    def compute_objectives(self, predictions, batch, stage):
```

```

return torch.nn.functional.l1_loss(predictions, batch["target"])

model = torch.nn.Linear(in_features=10, out_features=10)
brain = SimpleBrain({"model": model}, opt_class=lambda x: torch.optim.S
data = [{"input": torch.rand(10, 10), "target": torch.rand(10, 10)}]
brain.fit(range(10), data)

```



Avec seulement une dizaine de lignes de code, nous pouvons entraîner avec succès un modèle neuronal. Cette efficacité est due au fait que la classe Brain gère les détails complexes de la formation, tels que la gestion des états train() et eval() ou le calcul et l'application des gradients. En outre, la flexibilité de la classe permet de modifier chaque étape du processus en ajoutant des méthodes à la sous-classe. Cela signifie que même les procédures de formation complexes, telles que celles impliquées dans les réseaux adversariels génératifs (GAN), peuvent être intégrées de manière transparente dans la classe Brain.



Speechbrain nous mets à disposition un wrapper de train, ce qui nous permettra d'éviter bon nombre d'erreurs et de bugs et de lancer des trains plus facilement

3. Comment ajuster les hyperparamètre

Dans SpeechBrain, les files de train sont réparties en `train.py` (algo) et `train.yaml` (params)

Google Colab

https://colab.research.google.com/drive/1Pg9by4b6-8QD2iC0U7Ic3Vxq4GEwEdDz?usp=sharing#scrollTo=OYul_vzLbhJz



4. Comment Speech Brain nous aide à charger nos propre donnés (LibriSpeech ou LJSpeech)

Exploiter le script de training

speechbrain/recipes/LJSpeech/TTS at develop · speechbrain/speechbrain

A PyTorch-based Speech Toolkit. Contribute to speechbrain/speechbrain development by creating an account on GitHub.

<https://github.com/speechbrain/speechbrain/tree/develop/recipes/LJSpeech/TT>
S

speechbrain/
speechbrain

A PyTorch-based Speech Toolkit

147 Contributors 2k Used by 253 Discussions 8k Stars 11 Forks

<https://www.youtube.com/watch?v=R4HWbZmLGoU>

Etude des G2p

La conversion graphème-phonème (G2P), pierre angulaire des technologies modernes de traitement du langage naturel (NLP).

Ressources :

<https://www.youtube.com/watch?v=jf09yo3NfoQ>

La conversion graphème en phonème (G2P) constitue une technologie essentielle dans le domaine du traitement du langage naturel, reliant de manière transparente les points entre le texte écrit et les mots parlés. Cette technologie sous-tend plusieurs applications essentielles :

- **Synthèse de synthèse vocale (TTS)**
- **Reconnaissance vocale automatique (ASR)**
- **Aides à l'apprentissage des langues**



C'est quoi Graphème et Phonème?

- **Les graphèmes** représentent les plus petites unités du langage écrit. Ceux-ci incluent des lettres, des caractères et tout autre symbole contribuant à la représentation des mots écrits.
- **Les phonèmes**, quant à eux, sont les plus petites unités sonores d'une langue capables de distinguer un mot d'un autre. Ce sont les éléments constitutifs auditifs des langues parlées.

Test Effectués

<https://github.com/xinjli/transphone>

Models - Hugging Face

We're on a journey to advance and democratize artificial intelligence through open source and open science.

🤖 <https://huggingface.co/models?other=G2P>



<https://github.com/uiuc-sst/g2ps/tree/master>

Conclusion

Dans le cadre de l'exploration des technologies de traitement de la parole, ce projet a démontré l'efficacité et la polyvalence de SpeechBrain pour diverses applications, y compris la synthèse vocale avec FastSpeech2, l'amélioration de la parole, et la séparation des voix. L'intégration réussie d'un modèle G2p en anglais et l'analyse approfondie des mécanismes G2p montrent des avancées prometteuses dans la conversion graphème-phonème, essentielle pour la synthèse et la reconnaissance vocale. La facilité d'utilisation de SpeechBrain, couplée à sa capacité d'adaptation aux ensembles de données spécifiques, ainsi que son architecture permettant un apprentissage rapide et efficace grâce à la classe "Brain", révèlent son potentiel pour révolutionner la création et l'amélioration de technologies vocales sophistiquées.