Mise en place de l'autotranscription

POD V2

Utilisation de l'autotranscription dans Pod:

Remarque : à partir de la version 2.6.2 de Pod, c'est la version 0.8.2 de DeepSpeech qui est utilisée. Il faut adapter les modèles et les paramètres de configuration de Pod.

Pour découper le fichier audio de pod et faire sa transcription, nous avons besoin de Sox, il faut donc installer les deux lib suivantes :

- (django_pod) pod@podv2:/usr/local/django_projects/podv2$ sudo aptitude install sox
- (django_pod) pod@podv2:/usr/local/django_projects/podv2$ sudo apt-get install libsox-fmt-mp3

Il faut également installer le module python ffmpeg-normalize

- (django_pod) pod@podv2:/usr/local/django_projects/podv2$ pip install ffmpeg-normalize

L'ensemble des fichiers créés (modèle, alphabet, lm.binary et trie pour version 0.6.1 de deepspeech ou kenlm.scorer et output_graph.pbmm pour la version 0.8.2 de deepspeech) peuvent être stockés dans /usr/local/django_projects/transcription

il convient de faire un sous-dossier par langue (I.E: fr, en etc.)

```
``` pour version de pod < 2.6.2

pod@podv2:/usr/local/django_projects/transcription/model_fr$ ll
total 2716020
-rw-r--r-- 1 pod pod        157 nov.  14 21:14 alphabet.txt
-rw-r--r-- 1 pod pod 1395207692 oct.  13 10:08 lm.binary
-rw-r--r-- 1 pod pod 1125941232 nov.  15 09:40 model_tensorflow_fr.tar.xz
-rw-r--r-- 1 pod pod  189186587 nov.  15 08:42 output_graph.pbmm
-rw-r--r-- 1 pod pod   70846624 nov.  14 11:56 trie

```
```

le fichier "model_tensorflow_fr.tar.xz" peut-etre téléchargé sur le site du projet (attention à prendre la version adéquat : https://github.com/Common-Voice/commonvoice-fr/releases

par exemple pour la version : v0.6.0-fr-0.4

wget https://github.com/Common-Voice/commonvoice-fr/releases/download/v0.6.0-fr-0.4/model_tensorflow_fr.tar.xz

pour la version 0.8.2 (le premier contient les fichiers pour FR, les 2 autres sont pour EN):

wget https://github.com/Common-Voice/commonvoice-fr/releases/download/fr-v0.5.2/model_tensorflow_fr.tar.xz

wget https://github.com/mozilla/DeepSpeech/releases/download/v0.8.2/deepspeech-0.8.2-models.pbmm

wget https://github.com/mozilla/DeepSpeech/releases/download/v0.8.2/deepspeech-0.8.2-models.scorer

puis décompresser le/les fichiers :

tar -xJvf model_tensorflow_fr.tar.xz

Dans le fichier custom/settings-local.py, il suffit d'ajouter les paramètres suivant (une entrée par langue) :

```
Pour Pod à partir de la 2.6.2 avec fr et en :
```

....

USE_TRANSCRIPTION = True

DS_PARAM = {

'fr': { # le modèle deepspeech 0.5.2 FR

'model': "/usr/local/django_projects/transcription/model_fr/output_graph.pbmm",

'scorer': "/usr/local/django_projects/transcription/model_fr/kenlm.scorer",

},

'en': { # le modèle deepspeech 0.8.2 EN

'model': "/usr/local/django_projects/transcription/model_en/deepspeech-0.8.2-models.pbmm",

'scorer': "/usr/local/django_projects/transcription/model_en/deepspeech-0.8.2-models.scorer",

}

}

```
Pour Pod < 2.6.2
"""

USE_TRANSCRIPTION = True

DS_PARAM = {
    'fr': {

        # alphabet;txt contient tous les caractères de la langue lang

        'alphabet': "/usr/local/django_projects/transcription/model_fr/alphabet.txt",

        # le modèle deepspeech

        'model': "/usr/local/django_projects/transcription/model_fr/output_graph.pbmm",

        # le fichier de probabilités construit avec kenlm

        'lm': "/usr/local/django_projects/transcription/model_fr/lm.binary",

        # le fichier trie créé avec generate_trie à partir de lm.binary et

        # alphabet.txt

        'trie': "/usr/local/django_projects/transcription/model_fr/trie",

        'n_features': 26,            # Number of MFCC features to use

        # Size of the context window used for producing timesteps in the input

        # vector

        'n_context': 9,

        # Beam width used in the CTC decoder when building candidate

        # transcriptions

        'beam_width': 500,

        # The alpha hyperparameter of the CTC decoder / Language Model weight

        'lm_alpha': 0.65,

        'lm_beta': 1.4          # The beta hyperparameter of the CTC decoder / Word insertion bonus

    }

}

"""
```

Pour tester l'encodage en ligne de commande :

$ python manage shell (pour lancer le shell)

$> from pod.video import transcript (dans le shell, je charge le module transcript de pod)

$>transcript.main_threaded_transcript(<video_id>) (j'appelle la fonction main_threaded_transcript du module transcript pour transcrire la video dont l'identifiant est <video_id>)

Pour construire un modèle DeepSpeech :

Les DB doivent être de la forme :

```
main_folder
|—- dev folder
|—- test folder
|—- train folder
|—- dev.csv
|—- test.csv
|—- train.csv
```

les dossiers dev test et train contiennent des fichiers wave en 16bit 16kHz Mono de 5 à 20 secondes répartis selon environ dev 10% , test 10% , train 80% du nombre total de wave
les csv sont répartis en 3 colonnes dont les noms sont à respecter : "wav_filename", "wav_filesize", "transcript"

Pour importer l'ensemble des wav pour réaliser le modèle - trouver les wav en francais (exemple vorforge voir lien bas de page)

Pour installer deepspeech afin de construire le modèle, il faut se rendre sur le github de deepspesch : https://github.com/mozilla/DeepSpeech (Attention ne pas prendre la branche master)

Pour information,TensorFlow : IA opensource développé par Google (reconnaissance Image, audio, texte etc.) et Baidu : societe chinoise qui implémente le concept de reconnaissance vocale DeepSpeech

Mozilla : ils ont codé la transcription en utilisant tensorFlow et en suivant le concept de Baidu

Pour construire le modèle on lance la commande :

```
$ ./DeepSpeech.py
--train_files              # chemin du fichier train.csv
--dev_files                # chemin du fichier dev.csv
--test_files               # chemin du fichier test.csv
--export_dir               # chemin du dossier où sera exporté le fichier output_graph.pb
--validation_step 1
--train_batch_size         # nombre de fichiers wave propagés à travers le réseau dans la catégorie train , typiquement en **2 , souvent 16, 32, 64, 128 ou 256 (nb : plus le batch size est grand, plus la mémoire utilisée sera importante)
--dev_batch_size           # nombre de fichiers wave propagés à travers le réseau dans la catégorie dev
--test_batch_size          # nombre de fichiers wave propagés à travers le réseau dans la catégorie test
--checkpoint_step 1
-—n_hidden             # « nombre » de réseaux de neurones , de préférence en **2 , souvent 512, 1024, 2048 , doit être le même à chaque entrainement
--learning_rate        # taux d'apprentissage , généralement en puissance de 10 négative , souvent compris entre 1e-1 et 1e-6 (nb : à écrire en forme décimale : ex : 0.001)
--dropout_rate             # pourcentage de neurones tuées aléatoirement entre chaque epoch , de la forme 0.xxx; souvent max 12e-2
—-epoch                # OPTIONNEL , positif : nombre d'epoch après lequel stopper si il n'y a pas eu d'arrêt anticipé  ,  négatif : nombre d'epoch à réaliser en plus après un arrêt anticipé
--checkpoint_dir           # dossier pour les checkpoints
```

Pour obtenir le output_graph.pbmm :

```
$ cd 'export_dir'
$ wget https://index.taskcluster.net/v1/task/project.deepspeech.tensorflow.pip.master.cpu/artifacts/public/convert_graphdef_memmapped_format
$ chmod +x convert_graphdef_memmapped_format
$ convert_graphdef_memmapped_format --in_graph=output_graph.pb --out_graph=output_graph.pbmm
```

Si 4 epoch successives avec un taux de loss identiques, alors early stop (arret anticipé)
par défaut, le paramétre epoch est mis à 80

Pour réaliser tous les cycles, 2 heures par époch avec un n_hidden à 2048. il faut conmpter une dizaine de cycle

Pour améliorer le model, on peut lancer plusieurs fois la commande deepseech.py qui pointe vers le même output graph mais avec les sources différnetes.
Attention, il ne faut pas changer le n_hidden ni l'alphabet
Il faut jouer avec le learning rate et le dropout_rate pour chaque bdd utilisée pour améliorer le rendu
On peut créer notre propre source de donnée avec des vidéos sous-titrées (voir script de valentin pour le site caito.de)

sauvegarder les checkpoint et le graph avant de relancer la construction du modèle

une fois le modèle créer il faut le transformer en pbm (pour mise en memoire) voir doc

lm.binary et trie :

Pour créer un lm.binary :

Dans un fichier txt (ici vocabulary.txt) , peupler les lignes avec des phrases ou des mots (1 phrase ou 1 mot par ligne de préférence)
NB : tous les caractères dans ce fichier doivent être des caractères présents dans le fichier alphabet.txt

compiler le binaire kenlm en suivant le site : https://kheafield.com/code/kenlm/

en supposant être à la racine du dossier kenlm après compilation :
$ /bin/bin/./lmplz --text vocabulary.txt --arpa words.arpa --o K
en remplaçant K par un nombre entre 2 et 6 environs , K représente le nombre de mots dans les sections de phrases qui vont être réalisés durant le processus,
c'est le n-gram, plus la valeur du n-gram sera grande et plus le lm.binary sera bon, mais c'est un processus très consommateur de mémoire RAM et ROM
une valeur entre 3 et 5 est préférable selon la longueur des phrases présentes dans le fichier vocabulary.txt
NB : le fichier intermédiaire words.arpa peut avoir une taille conséquente selon le vocabulary.txt de base (plusieurs Go)

Une fois le words.arpa créé :
$ /bin/bin/./build_binary -T -s 'path_to_words.arpa' 'path_to_export_lm.binary'

Pour créer le trie :

suivre les instructions à :
https://github.com/mozilla/DeepSpeech/blob/master/native_client/README.md#compile-libdeepspeechso--generate_trie
pour générer le fichier generate_trie

lorsque generate_trie est build :
$ chmod +x generate_trie
$ ./generate_trie 'path_to_alphabet.txt' 'path_to_lm.binary' 'path_to_export_trie'

Liens DB :

COMMON VOICE :
https://voice.mozilla.org/fr/datasets
Telecharger et extraire l'archive puis executer l'importer bin/import_cv2.py du depot deepspeech

VOXFORGE :
http://www.repository.voxforge1.org/downloads/fr/Trunk/Audio/Main/16kHz_16bit/
Dans bin/import_voxforge.py remplacer l'url ligne 123 par celle ci-dessus et executer le fichier

CAITO :
https://www.caito.de/2019/01/the-m-ailabs-speech-dataset/#more-242
Telecharger et extraire l'archive, puis executer le fichier aux_db_fr.py avec 2 paramètres, le chemin du dossier d'extraction et un nom de genre (female, male ou mix)
Attention, ce fichier est brut, à utiliser avec précaution.
NB : pour le dossier Mix il faut lui créer un dossier parent de manière a avoir l'arborescence ./mix/mix/… au lieu de ./mix/…

LINGUA LIBRE :
executer la commande en supposant être dans le dossier racine du projet deepspeech
$ ./bin/import_lingua_libre 'path_to_download' --qId 21 —-iso639-3 fra -—english-name French —-normalize (optionnel)

Liens utiles :

https://discourse.mozilla.org/c/deep-speech
https://discourse.mozilla.org/t/tutorial-how-i-trained-a-specific-french-model-to-control-my-robot/22830
https://discourse.mozilla.org/t/un-premier-modele-francais/41100

https://github.com/Common-Voice/commonvoice-fr/releases
https://github.com/Common-Voice/commonvoice-fr
https://github.com/mozilla/voice-corpus-tool