Installation de l'autotranscription en Pod V3

- Si vous souhaitez déporter la transcription sur le serveur d'encodage, les commandes suivantes sont à effectuer sur le serveur d'encodage (voir la page Déporter l'encodage sur un ou plusieurs serveurs en Pod V3)
- Utilisation de l'auto-transcription dans Pod
- STT (Coqui Ai)
- Pour tester la transcription en ligne de commande
- Vosk
    - [OPTIONNEL] Activer l'enrichissement du modèle vosk dans Pod
- Whisper (v3.5.0)

Si vous souhaitez déporter la transcription sur le serveur d'encodage, les commandes suivantes sont à effectuer sur le serveur d'encodage (voir la page Déporter l'encodage sur un ou plusieurs serveurs en Pod V3)

Utilisation de l'auto-transcription dans Pod

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

```
(django_pod) pod@:/$ sudo apt-get install sox
(django_pod) pod@:/$ sudo apt-get install libsox-fmt-mp3
```

> **CentOS**
>
> Si vous n'avez pas de chance et utilisez encore CentOS 6, la doc suivante peut aider : https://www.linuxhelp.com/how-to-install-sox-on-centos-6

Il faut également installer le module python `ffmpeg-normalize`

```
(django_pod) pod@:/path/to/project/django_projects/pod$ pip install ffmpeg-normalize
```

L'ensemble des modèles peuvent être stockés dans `/path/to/project/django_projects/transcription`.
Il convient de faire un sous-dossier par langue (I.E: fr, en etc.), et un sous-dossier par type de modèle (I.E: stt, vosk, etc.)

Par exemple, pour un modèle vosk français:

`/path/to/project/django_projects/transcription/fr/vosk/vosk-model-fr-0.6-linto-2.2.0/`

> À présent, vous pouvez choisir d'installer un des 2 modèles STT ou Vosk. Il est toutefois conseillé d'utiliser **Vosk**.

STT (Coqui Ai)

Il faut installer l'application dans l'environnement virtuel de Pod (stt==1.4.0)

installation de STT
`(django_pod3)pod@podv3:/usr/local/django_projects/podv3$ pip3 install stt`

Les fichiers peuvent être téléchargés sur le site du projet : https://github.com/coqui-ai/STT-models

```
pod@podv3:/usr/local/django_projects/transcription/model_fr/stt$ ll
-rw-r--r-- 1 pod pod        248 nov.  14 21:14 alphabet.txt
-rw-r--r-- 1 pod pod  189372825 nov.  15 09:12 model.pbmm
-rw-r--r-- 1 pod pod  1007576678 nov.  15 08:42 fr-cvfr-2-prune-kenlm.scorer
-rw-r--r-- 1 pod pod  47500492 nov.  15 08:54 model.tflite
```

Dans le fichier custom/settings-local.py, il suffit d'ajouter les paramètres suivant:

Pour Pod à partir de la version 3 avec fr et en :

```
# Transcription
USE_TRANSCRIPTION = True

## Transcription use
# * STT
# * VOSK
TRANSCRIPTION_TYPE = "STT"

# Paramétrage des modèles
# * Pour télécharger les Modèles STT : https://coqui.ai/models

TRANSCRIPTION_MODEL_PARAM = {
        # les modèles Stt
        'STT': {
                'fr': {
                        'model': "/usr/local/django_projects/transcription/model_fr/stt/model.pbmm",
                        'scorer': "/usr/local/django_projects/transcription/model_fr/stt/fr-cvfr-2-prune-kenlm.
scorer",
                }
        }
}
```

Pour ajouter un modèle d'un autre langage, ajouter une entrée comme l'exemple ci-dessous, fait pour le langage Anglais "en" :

```
# Paramétrage des modèles
# * Pour télécharger les Modèles STT : https://coqui.ai/models
TRANSCRIPTION_MODEL_PARAM = {
        # les modèles Stt
        'STT': {
                'fr': {
                        'model': "/usr/local/django_projects/transcription/model_fr/stt/model.pbmm",
                        'scorer': "/usr/local/django_projects/transcription/model_fr/stt/fr-cvfr-2-prune-kenlm.
scorer",
                },
                'en': {
                        'model': "/usr/local/django_projects/transcription/model_en/stt/model.pbmm",
                        'scorer': "/usr/local/django_projects/transcription/model_en/stt/kenlm.scorer",
                }
        }
}
```

Pour tester la transcription en ligne de commande

```
(django_pod)$> python manage.py shell
Python 3.7.3
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> video_id = 1234
>>> from pod.video import transcript
>>> transcript.main_threaded_transcript(video_id)
```

Vosk

I faut installer l'application dans l'environnement virtuel de Pod (vosk==0.3.45)

installation de VOSK

```
(django_pod3)pod@podv3:/usr/local/django_projects/podv3$ pip3 install vosk
```

Les fichiers pour les modèles peuvent être téléchargés sur cette page : https://alphacephei.com/vosk/models.

Par exemple pour le modèle français :

```
(django_pod3) pod@:/path/to/project/django_projects/transcription/fr/vosk/$ wget https://alphacephei.com/vosk
/models/vosk-model-fr-0.6-linto-2.2.0.zip
```

Après avoir téléchargé le modèle, il faut le décompresser. Pour cela vous pouvez utiliser la librairie unzip :

```
(django_pod3) pod@:/path/to/project/django_projects/transcription/fr/vosk/$ sudo apt-get install unzip
(django_pod3) pod@:/path/to/project/django_projects/transcription/fr/vosk/$ unzip vosk-model-fr-0.6-linto-2.2.0.
zip
```

Dans le fichier `custom/settings-local.py`, il suffit d'ajouter les paramètres suivants (a préciser egalement sur les frontaux pour qu'il puisse lister les langue/model de dispo depuis la 3.3.1) :

Pour Pod avec Vosk en fr :

```
# Transcription
USE_TRANSCRIPTION = True

## Transcription use
# * STT
# * VOSK
TRANSCRIPTION_TYPE = "VOSK"

# Paramétrage des modèles
# * Pour télécharger les Modèles Vosk : https://alphacephei.com/vosk/models
TRANSCRIPTION_MODEL_PARAM = {
  # les modèles Vosk
  'VOSK':{
    'fr': {
      'model': "/path/of/project/django_projects/transcription/fr/vosk/vosk-model-fr-0.6-linto-2.2.0",
    }
  }
}
```

Pour ajouter un modèle d'un autre langage, ajouter une entrée comme l'exemple ci-dessous, fait pour le langage Anglais "en" :

```
# Paramétrage des modèles
# * Pour télécharger les Modèles Vosk : https://alphacephei.com/vosk/models
TRANSCRIPTION_MODEL_PARAM = {
  # le modèle vosk
  'VOSK':{
    'fr': {
      'model': "/path/of/project/django_projects/transcription/fr/vosk/vosk-model-fr-0.6-linto-2.2.0",
    },
         'en': {
      'model': "/path/of/project/django_projects/transcription/en/vosk/vosk-model-en-us-0.22",
    }
  }
}
```

Maintenant lors de l'upload d'une vidéo avec l'auto-transcription activée le modèle Vosk sera utilisé pour effectuer la transcription.

[OPTIONNEL] Activer l'enrichissement du modèle vosk dans Pod

En installant les modèles de compilation vous pourrez contribuer à l'enrichissement des modèles.

Les modèles utilisés pour l'enrichissement du modèle peuvent être stockés dans `/path/to/project/django_projects/compile-model`

Il faut télécharger le modèle de compilation correspondant sur ce lien : https://alphacephei.com/vosk/lm#update-process.

Par exemple pour le modèle français :

```
(django_pod3) pod@:/path/to/project/django_projects/compile-model/fr$ wget https://alphacephei.com/vosk/models
/vosk-model-fr-0.6-linto-2.2.0-compile.zip
```

Il faut après avoir téléchargé le modèle de compilation, le décompresser. Pour cela vous pouvez utiliser la librairie unzip :

```
(django_pod3) pod@:/path/to/project/django_projects/compile-model/fr$ sudo apt-get install unzip
(django_pod3) pod@:/path/to/project/django_projects/compile-model/fr$ unzip vosk-model-fr-0.6-linto-2.2.0-
compile.zip
```

Il faut seulement que la structure du dossier compile-model ressemble à cela :

```
(django_pod3) pod@:/path/to/project/django_projects/compile-model
```

```
compile-model
|--fr
|   |--conf
|   |   |...
|   |
|   |--data
|   |   |...
|   |
|   |--db
|   |   |...
|   |
|   |--exp
|   |   |...
|   |
|   |--local
|   |   |...
|   |
|   |--mfcc
|   |   |...
|   |
|   |--steps
|   |   |...
|   |
|   |--utils
|   |   |...
|   |
|   |--cmd.sh
|   |--compile-grapg.sh
|   |--decode.sh
|   |--dict.py
|   |--path.sh
|
|--en
```

Maintenant il faut installer docker sur votre machine. (voir https://docs.docker.com/engine/install/debian/ si besoin)

Après que docker soit installé, créer un fichier entrypoint.sh et DockerFile dans un même dossier.

Copier le script suivant dans le fichier `entrypoint.sh`;

entrypoint.sh

```
#!/bin/bash
modelPath="$KALDI_ROOT/compile-model/$1"
cat "$KALDI_ROOT/tools/env.sh" > "$modelPath/path.sh"
cd $modelPath
/bin/bash -c "./compile-graph.sh"
/bin/bash -c "utils/build_const_arpa_lm.sh lm.gz data/lang_test data/lang_test_rescore"
```

Puis copier le code ci-dessous, fait sur mesure afin d'enrichir un modèle dans le Fichier `DockerFile`, cela créera un container avec tout ce qu'il faut installer :

DockerFile

```
## Build the DockerFile
# docker build --tag kaldi -f DockerFile .
##
## Example of manual execution of the Docker file
# sudo docker run -v ${PWD}/compile-model:/kaldi/compile-model -it kaldi
##
FROM debian:10
RUN apt-get update && apt-get install -y ca-certificates \
 && apt-get install -y \
 python3-pip \
 git \
 && apt-get install -y zlib1g-dev automake autoconf unzip wget sox gfortran libtool subversion python2.7 nano
libfst-tools \
 && apt-get clean
RUN python3 --version
ENV KALDI_ROOT="/kaldi"
RUN git clone https://github.com/kaldi-asr/kaldi.git $KALDI_ROOT
WORKDIR $KALDI_ROOT"/tools"
RUN bash $KALDI_ROOT"/tools/extras/check_dependencies.sh"
RUN touch $KALDI_ROOT"/tools/python/.use_default_python"
RUN bash $KALDI_ROOT"/tools/extras/install_mkl.sh"
RUN apt-get install gfortran sox
RUN make -j $(nproc)
RUN pip3 install phonetisaurus
RUN bash $KALDI_ROOT"/tools/extras/install_opengrm.sh"
RUN make
RUN bash $KALDI_ROOT"/tools/extras/install_irstlm.sh"
RUN apt-get install gawk
RUN bash $KALDI_ROOT"/tools/extras/install_srilm.sh" "unkown" "unkown" "unkown"
RUN cd $KALDI_ROOT"/src" && ./configure --shared
RUN cd $KALDI_ROOT"/src" && make depend -j $(nproc)
RUN cd $KALDI_ROOT"/src" && make -j $(nproc)
RUN cd $KALDI_ROOT"/src/fstbin" && make
RUN echo "export PATH="$KALDI_ROOT"/src/fstbin:\$PATH" >> $KALDI_ROOT"/tools/env.sh"
RUN cd $KALDI_ROOT"/src/lmbin" && make
RUN echo "export PATH="$KALDI_ROOT"/src/lmbin:\$PATH" >> $KALDI_ROOT"/tools/env.sh"
RUN cd $KALDI_ROOT"/src/tree" && make
RUN echo "export PATH="$KALDI_ROOT"/src/tree:\$PATH" >> $KALDI_ROOT"/tools/env.sh"
RUN cd $KALDI_ROOT"/src/bin" && make
RUN echo "export PATH="$KALDI_ROOT"/src/bin:\$PATH" >> $KALDI_ROOT"/tools/env.sh"
COPY entrypoint.sh /entrypoint.sh
WORKDIR $KALDI_ROOT
ENTRYPOINT ["/entrypoint.sh"]
```

Après avoir copié et créé les deux fichiers `Dockerfile` et `entrypoint.sh` il suffit de lancer la commande ci-dessous en étant dans la même dossier que les fichiers précédemment mentionnés.

```
docker build --tag kaldi -f DockerFile .
```

Pour finir, il faut activer l'enrichissement du modèle vosk dans une application pod, pour cela il suffit d'ajouter dans le fichier custom/settings-local.py les paramètres suivants :

```
ACTIVE_ENRICH = True
MODEL_COMPILE_DIR = "/path/to/project/django_projects/compile-model"
```

Whisper (v3.5.0)

Sur les encodeurs :

```
pip install openai-whisper
```

ou si vous souhaitez bénéficier des derniers commits

```
pip install git+https://github.com/openai/whisper.git
```

Exemple de configuration du custom/settings_local :

```
TRANSCRIPTION_TYPE = "WHISPER"

TRANSCRIPTION_MODEL_PARAM = {
        'WHISPER':{
                'fr': {
                        'model': "small",
                        'download_root': "/pod-transcription/transcription/whisper/",
                },
                'en': {
                        'model': "small",
                        'download_root': "/pod-transcription/transcription/whisper/",
                }
        }
}
```

Voir détails ici pour le choix du modèle
Le small n'est pas plus gourmand que vosk et est déjà performant