

De l'IA en Local

- [Découpeur Vidéo Intelligent avec IA local](#)
- [DES pistes d'évolutions](#)

Découpeur Vidéo Intelligent avec IA local

Objectifs :

- Automatiser le découpage de vidéos d'interviews par thématiques
- Exploiter l'IA locale (Ollama) pour l'analyse sémantique
- Transcription audio précise avec Whisper
- Génération automatique de segments vidéo pertinents

Contexte du projet

Application desktop Windows permettant de traiter automatiquement des vidéos d'interviews pour identifier et extraire des segments thématiques cohérents, sans nécessiter de découpe manuelle chronophage.

Environnement :

- Windows 10/11 (64 bits)
- Python 3.12 avec PyQt5
- FFmpeg pour le traitement vidéo
- Ollama pour l'analyse IA en local

Architecture du système

Pipeline de traitement

1. Vidéos sources (MTS/MP4/AVI)
 - ↓
 2. Concaténation automatique (si plusieurs fichiers)

↓
3. Extraction audio → WAV
↓
4. Transcription → Whisper (OpenAI)
↓
5. Analyse thématique → Ollama (local)
↓
6. Génération des segments → FFmpeg
↓
7. Export final → MP4 optimisés

Stack technique

Backend:

- `whisper` : Transcription audio multilingue
- `ollama` : Modèles LLM locaux pour l'analyse
- `ffmpeg-python` : Manipulation vidéo/audio
- `torch` : Framework ML pour Whisper

Frontend:

- `PyQt5` : Interface graphique native
- `librosa` : Traitement audio
- Système de logs intégré

Étape 1 : Installation des prérequis

FFmpeg (traitement vidéo)

Installation :

```
# Télécharger depuis https://www.gyan.dev/ffmpeg/builds/  
# Extraire dans C:\ffmpeg  
# Ajouter au PATH : C:\ffmpeg\bin
```

Vérification :

```
ffmpeg -version
```

Explication :

- FFmpeg = Outil de référence pour manipuler vidéos/audio
- Utilisé pour concaténer, extraire audio, découper les segments
- Version complète nécessaire (incluant libx264, aac)

Ollama (modèles IA locaux)

Installation :

```
# Télécharger depuis https://ollama.ai/download
# Installer l'application
# Télécharger un modèle :
ollama pull qwen2.5:3b
```

Modèles disponibles :

Modèle	Taille	RAM	Forces	Cas d'usage
qwen2.5:3b	2 GB	4 GB	Léger, rapide, bon raisonnement	Recommandé - Analyse de texte, thématisation
gemma2:2b	1.6 GB	3 GB	Très léger, rapide	PC limités en RAM, tests
mistral:7b	4 GB	8 GB	Excellent raisonnement, précis	Analyse complexe, nuances fines
llama3.1:8b	4.7 GB	10 GB	Multilingue, polyvalent	Multilingue, transcriptions mixtes
phi3:3.8b	2.3 GB	6 GB	Optimisé Microsoft, efficace	Alternative à Qwen

Explication :

- Ollama = Plateforme pour exécuter des LLM en local (sans cloud)
- Analyse le contenu textuel pour détecter les changements de thème
- Pas de coût API, confidentialité des données garantie
- Modèles optimisés pour CPU (pas de GPU requis)

Configuration dans le projet :

Fichier `src/config.py:23`

```
OLLAMA_MODEL = "qwen2.5:3b" # Modifier ici pour changer de modèle
```

Python et dépendances

Installation automatique :

```
# Double-cliquer sur install.bat
# OU manuellement :
python -m venv venv_312
venv_312\Scripts\activate
pip install -r requirements.txt
```

Dépendances principales :

- `openai-whisper` : Modèle de transcription
- `torch` : Backend pour Whisper
- `ollama` : Client Python pour Ollama
- `PyQt5` : Interface graphique
- `ffmpeg-python` : Wrapper Python pour FFmpeg

Étape 2 : Configuration des modèles

Whisper (transcription)

Fichier `src/config.py:18`

```
WHISPER_MODEL = "medium" # Options : tiny, base, small, medium, large
WHISPER_LANGUAGE = "fr"
```

Comparaif des modèles Whisper :

Modèle	Taille	RAM	Vitesse	Qualité	Usage
tiny	75 MB	1 GB	32x	Moyenne	Tests rapides

Modèle	Taille	RAM	Vitesse	Qualité	Usage
base	145 MB	1 GB	16x	Correcte	Vidéos courtes
small	488 MB	2 GB	6x	Bonne	Bon compromis
medium	1.5 GB	5 GB	2x	Excellente	Recommandé
large	3 GB	10 GB	1x	Maximale	Production finale

Explication :

- Whisper = Modèle ML d'OpenAI pour speech-to-text
- Exécution locale (pas de cloud)
- Supporte 99 langues avec timestamps précis
- Vitesse relative : tiny = 32× plus rapide que large

Ollama (analyse IA)

Fichier src/config.py:23-24

```
OLLAMA_MODEL = "qwen2.5:3b"
MAX_TOKENS_PER_ANALYSIS = 4000
```

Choisir son modèle selon les besoins :

PC limité (4-8 GB RAM) :

```
OLLAMA_MODEL = "gemma2:2b" # Léger et rapide
```

PC standard (8-16 GB RAM) :

```
OLLAMA_MODEL = "qwen2.5:3b" # Recommandé (défaut)
```

PC puissant (16+ GB RAM) :

```
OLLAMA_MODEL = "mistral:7b" # Meilleure précision
```

Multilingue :

```
OLLAMA_MODEL = "llama3.1:8b" # Multilangue
```

Explication du rôle d'Ollama :

1. Reçoit la transcription découpée en chunks (~4000 tokens)
2. Identifie les thèmes principaux et transitions
3. Génère des titres descriptifs pour chaque segment
4. Propose des timestamps de découpe cohérents

Prompt système utilisé :

```
Analyse cette transcription d'interview et identifie les différents
thèmes abordés. Pour chaque thème, fournis :
- Un titre court et descriptif
- Le timestamp de début/fin
- Une justification du découpage
```

Étape 3 : Utilisation de l'application

Lancement

```
# Double-cliquer sur launch.bat
# OU :
venv_312\Scripts\python src\main.py
```

Workflow type

1. Sélection des fichiers

- Cliquer sur "Sélectionner les fichiers vidéo"
- Choisir un ou plusieurs fichiers (MTS, MP4, AVI, MOV, MKV)
- Les MTS multiples sont automatiquement concaténés

2. Traitement automatique

- Cliquer sur "Démarrer l'analyse"
- Suivre la progression dans l'interface :
 - ☐ Concaténation des fichiers (si nécessaire)
 - ☐ Extraction audio → WAV
 - ☐ Transcription Whisper (la plus longue étape)

- ☐ Export transcription → TXT + JSON
- ☐ Analyse IA Ollama par chunks
- ☐ Génération des segments proposés

3. Validation des segments

- Consulter la liste des segments détectés
- Double-cliquer pour éditer :
 - Modifier le titre
 - Ajuster les timestamps
 - Fusionner/diviser des segments
- Prévisualiser avec le lecteur intégré

4. Export final

- Choisir le dossier de destination
- Cliquer sur "Exporter tous les segments"
- Les fichiers MP4 sont générés avec nommage séquentiel

Étape 4 : Performances et optimisations

Temps de traitement (vidéo 1h)

Configuration standard (Whisper medium + Qwen 3B) :

Étape	Durée	Facteurs
Concaténation	1-2 min	Nombre de fichiers sources
Extraction audio	10 sec	Taille de la vidéo
Transcription	35-42 min	Modèle Whisper, CPU
Analyse IA	8-12 min	Modèle Ollama, longueur texte
Export segments	10-15 min	Nombre de segments, qualité
TOTAL	60-90 min	

Optimisations possibles

Pour accélérer (sacrifie qualité) :

```
WHISPER_MODEL = "small"           # Divise temps par 3
OLLAMA_MODEL = "gemma2:2b"        # Divise RAM par 2
VIDEO_QUALITY = 28                 # Fichiers plus légers
```

Pour qualité max (plus lent) :

```
WHISPER_MODEL = "large"           # Transcription parfaite
OLLAMA_MODEL = "mistral:7b"       # Analyse fine
VIDEO_QUALITY = 18                 # Qualité visuelle maximale
```

Étape 5 : Structure des fichiers générés

```
OPCI-VIDEO-TEST-N2/
|
├─ output/
|   └─ transcriptions/
|       ├── interview_transcription.txt    # Texte lisible avec timestamps
|       └─ interview_transcription.json    # Données structurées (words + segments)
|
├─ [dossier_choisi]/
|   ├── 01_Introduction_et_parcours.mp4
|   ├── 02_Defis_techniques_rencontres.mp4
|   ├── 03_Vision_future_du_projet.mp4
|   └─ ...
|
├─ logs/
|   └─ video_cutter.log                    # Logs détaillés du traitement
|
└─ temp/
    └─ [fichiers_temporaires]              # Nettoyés après export
```

Explication des formats :

TXT :

```
[00:00:05] Bonjour, je suis ravi de...  
[00:00:42] Donc le projet a commencé en...
```

JSON :

```
{  
  "text": "Transcription complète...",  
  "segments": [  
    {  
      "start": 5.2,  
      "end": 42.8,  
      "text": "Bonjour, je suis ravi de...",  
      "words": [...]  
    }  
  ]  
}
```

Étape 6 : Dépannage

Ollama ne répond pas

```
# Vérifier qu'Ollama est lancé  
ollama list  
  
# Relancer Ollama  
taskkill /F /IM ollama.exe  
ollama serve  
  
# Vérifier le modèle  
ollama pull qwen2.5:3b
```

Explication :

- Ollama doit tourner en arrière-plan (service)
- Icône visible dans la barre des tâches Windows
- Port par défaut : `http://localhost:11434`

Whisper trop lent

Solution 1 : Modèle plus léger

```
WHISPER_MODEL = "small" # Au lieu de "medium"
```

Solution 2 : GPU (si disponible)

- Installer `torch` avec support CUDA
- Whisper utilisera automatiquement le GPU NVIDIA
- Gain de vitesse : 5-10x

Solution 3 : Découper la vidéo

- Traiter par tranches de 30 min
- Concaténer les transcriptions manuellement

Erreur de mémoire (OOM)

Symptômes :

```
RuntimeError: CUDA out of memory  
MemoryError: Unable to allocate...
```

Solutions :

```
# 1. Réduire le modèle Whisper  
WHISPER_MODEL = "base"  
  
# 2. Réduire le modèle Ollama  
OLLAMA_MODEL = "gemma2:2b"  
  
# 3. Réduire chunk size  
MAX_TOKENS_PER_ANALYSIS = 2000
```

Segments mal découpés

Causes possibles :

- 1. Transcription imprécise → Tester `medium` ou `large`
- 2. Modèle Ollama trop léger → Essayer `mistral:7b`
- 3. Transitions abruptes dans l'interview → Ajustement manuel

Améliorer la détection :

```
# Tester un modèle plus puissant
OLLAMA_MODEL = "mistral:7b"

# Ou un modèle spécialisé français
ollama pull vigogne:7b # Modèle français
```

Avantages de l'approche locale (Ollama)

vs API Cloud (GPT-4, Claude, etc.)

Critère	Ollama (local)	API Cloud
Coût	Gratuit, illimité	0.01-0.10€ par analyse
Confidentialité	Données restent en local	Envoyées au cloud
Vitesse	Dépend du CPU local	Dépend du réseau
Disponibilité	Offline OK	Nécessite Internet
Qualité	Bonne (modèles 3-7B)	Excellente (70B+)

Cas d'usage idéal pour Ollama :

- Vidéos confidentielles (entreprise, médical)
- Traitement par lots sans limite
- Pas de connexion Internet fiable
- Coût récurrent inacceptable

Évolutions futures

Fonctionnalités prévues :

- Export multi-formats (MOV, AVI, etc.)
- Support multilingue simultané
- Mode batch (dossier complet)

Modèles IA à tester :

- `nous-hermes2:10.7b` : Excellent suiveur d'instructions
 - `solar:10.7b` : Performance/taille optimale
-

Ressources et références

Documentation officielle :

- [Ollama Models](#)
- [OpenAI Whisper](#)
- [FFmpeg Documentation](#)

Comparaison modèles :

- [Ollama Performance Benchmarks](#)
 - [Whisper Model Card](#)
-

Configuration matérielle recommandée

Minimum (workflow ralenti) :

- CPU : Intel i5 / AMD Ryzen 5 (4 cœurs)
- RAM : 8 GB
- Stockage : 50 GB SSD

- Modèles : `small` + `gemma2:2b`

Recommandé (workflow fluide) :

- CPU : Intel i7 / AMD Ryzen 7 (8 cœurs)
- RAM : 16 GB
- Stockage : 100 GB SSD
- Modèles : `medium` + `qwen2.5:3b`

Optimal (vitesse maximale) :

- CPU : Intel i9 / AMD Ryzen 9 (16 cœurs)
- RAM : 32 GB
- GPU : NVIDIA RTX 3060+ (support CUDA)
- Stockage : 256 GB NVMe SSD
- Modèles : `large` + `mistral:7b`

Version : 1.0 **Dernière mise à jour** : Novembre 2024 **Licence** : MIT **Auteur** : OPCI Productions

DES pistes d'évolutions