VidChapters-7M: Video Chapters at Scale

本研究では、ユーザがキャプション付けした動画の大規模データセットである VidChapters-7M を紹介する。このデータセットを用いて3つのタスクを研究し、VidChapters-7M で学習した動画チャプター生成モデルが、高密度の動画キャプションにうまく移行することを示す。

このリポジトリは、私たちの論文のコードを提供する.

- 環境設定
- VidChapters-7M のデータ収集パイプライン（独自のチャプター付きビデオセットを収集したい場合）
- データのダウンロード方法と処理済みデータファイル
- データ処理および分析スクリプト（前処理を再現したい場合）
- VidChapters-7M における真実境界なし/ありのビデオチャプター生成とビデオチャプターのグラウンディング、YouCook2 と ViTT における高密度ビデオキャプションのタスクのためのトレーニングと評価スクリプト
- 事前学習済みモデルのチェックポイント
- 事前に学習されたVid2Seqモデルを使用して、選択したビデオにチャプターまたは密なキャプションを付けるデモ

このコードベースには Vid2Seq のPyTorch実装も含まれている（特に `model/vid2seq.py` ）。オリジナルの Jax の実装とは以下のような違いがある.

- t5-v1_1-base の代わりにt5-base を使用することで、いくつかのアーキテクチャ上の違いが生じる（ `is_gated_act=True` の代わりに `False` ）。
- 最適化ステップごとに、時間トークンに関連する重みの正規化を追加する。
- トレーニング中のランダムな時間的トリミングはない
- グーグルASRに代わるウィスパーASR

パスと条件

`args.py` ファイル(PDVC / Moment-DETRを使用する場合は `PDVC/cfgs` / `moment_detr/moment_detr/scripts/`)のスクリプトの空パスを埋める。

METEORキャプションメトリックで評価スクリプトを使用するには、Javaも必要です。

要件をインストールするには（本来はPython 3.7で行われる）、以下を実行する.

```
pip install -r requirements.txt
```

注意：

- Whisper ASR の抽出は、Python 3.10 と PyTorch 2.0 を使用し、WhisperX で指定されたとおりに作成された別の conda 環境で行われる。
- PDVC の実験では、変形可能な注目レイヤーをコンパイルするために、PDVC が提案したように、別の conda 環境で実行されている。

データ収集パイプライン

手始めに、YouTube のビデオ ID（必ずしもビデオチャプターが含まれているとは限らない）をたくさん取得し、 yt-dlp を使って YouTube から説明をダウンロードする。例：

```
yt-dlp https://www.youtube.com/watch?v=<VIDEO_ID> --write-description --skip-download
```

そして、 `SSD_DIR/chapters_descriptions` に `.txt` ファイルとして説明がダウンロードされていると仮定して、説明からチャプターを抽出するために `python collection/desc2chapters.py` を実行することができます。出力ファイルは、ユーザがチャプターを付けた動画の ID を、チャプターのタイトルとタイムスタンプにマッピングします。 `yt-dlp https://www.youtube.com/watch?v=<VIDEO_ID>` などで、チャプターを持つ動画の YouTube 動画コンテンツをダウンロードできます。

データのダウンロード

VidChapters-7M: データセットのアノテーションとASRはこちらのリンクからダウンロードできます。アノテーションは `DATA_DIR/AllChapters` からダウンロードしてください。また、ここで加工したアノテーションも提供しています。

HowTo100M **：** センテンス化されたデータセットを使用しています。 `DATA_DIR/howto100m` からダウンロードしてください。

ViTT **：** データプロバイダーからダウンロードする。また、YouTube-8Mの4文字IDとYouTube動画IDのマッピングもダウンロードする必要があります。 `DATA_DIR/ViTT` からダウンロードしてください。また、処理済みのアノテーション、ASR、ビジュアルフィーチャーもこちらで提供しています。

YouCook2 **：** データプロバイダーからダウンロードしてください。 `YouCook2` でダウンロードしてください。また、処理済みのアノテーション、ASR、ビジュアルフィーチャーもこちらで提供しています。

データ処理

視覚的特徴抽出

FrozenBiLM に従って、全動画の CLIP ViT-L/14 @ 224 ピクセルの特徴を 1FPS で抽出する。VidChapters-7M / HowTo100M では `SSD_DIR/chapters_clipvitl14_features` / `SSD_DIR/howto100m_clip_features` に動画ごとに1ファイルずつ保存し、YouCook2 / ViTT では全動画を1つの `.pth` ファイルにまとめる。

ASR抽出

ASR を抽出するために、視覚的特徴抽出と同様に用意された `csv` ファイルと、抽出された ASR を格納する `output_path` が与えられ、単一のGPU上で実行される.

```
conda activate whisperX_env
python asr_extract/whisper_inference.py --csv=<csv> --output_path=<output_path> --faster
```

多くのジョブで並列化することもできる。このためには、 `<MODEL_DIR>` にある Whisper Large-V2 モデルの重みをダウンロードしておく必要がある。

次に、抽出された ASR を `asr` ファイルとしてまとめる.

```
python asr_extract/merge_asr_whisper.py <output_path> DATA_DIR/AllChapters/whisper.pkl
```

単語レベルのタイムスタンプを抽出し、ASR をセンテンスにセグメント化するために、シングル GPU で実行する.

```
conda activate whisperX_env
python asr_extract/whisper_align.py --csv=<csv> --asr=DATA_DIR/AllChapters/whisper.pkl --output_
```

複数のジョブで並列化することもできる。このためには、 `<MODEL_DIR>` にある WhisperX から全言語のアライメントモデル重みをダウンロードしておく必要があることに注意してください。

最後に、アラインメントされた ASR を1つのファイルにマージします：

```
python asr_extract/merge_asr_whisper_align.py <align_output_path> DATA_DIR/AllChapters/asr.pkl [
```

注釈ファイル

注釈ファイルを前処理するには

```
python preproc/chapters_to_dvc.py
python preproc/chapters_to_vmr.py
python preproc/vitt.py
python preproc/youcook.py
```

分析

ASR やチャプターから言語を検出するために、シングルGPUで実行する.

```
python analysis/language.py
```

多くのジョブで並列化することもできる。

ジェンダーの統計を取るために、CPUで実行する.

```
python analysis/gender.py
```

NSFW フレームや有害なチャプタータイトルや ASR を含むビデオを検出するために、シングル GPU で実行します（このためには、pip でインストールできる detoxify=0.5.1 も必要です）.

```
python analysis/nsfw.py
```

多くのジョブで並列化することができます。NSFW 分類器とDetoxify 言語モデルをダウンロードする必要があることに注意してください。

また、論文のプロットのコードはノートブックの `analyze/plots.ipynb` に、論文で紹介した手動評価の詳細は `analyze/manual_assessment.xlsx` にあります。

モデルのチェックポイント

HowTo100M 事前トレーニング、完全なビデオチャプター生成タスク、および高密度ビデオキャプションタスクについて、以下の Vid2Seq チェックポイントをリリースし、対応する SODA パフォーマンスを報告する。

学習データ	VidChapters-7M (テスト)	YouCook2 (バリデーション)	ViTT (テスト)	URL	大きさ
HouTo100M				Drive	1.1GB
VidChapters-7M	10.6			Drive	1.1GB
HowTo100M + VidChapters-7M	11.4			Drive	1.1GB
HowTo100M + VidChapters-7M + YouCook2		10.3		Drive	1.1GB
HowTo100M + VidChapters-7M + ViTT			15.0	Drive	1.1GB

グラウンド・トゥルースの境界を持つビデオ章生成のタスクについて、以下の Vid2Seq チェックポイントをリリースし、それに対応する CIDEr のパフォーマンスを報告する。

学習データ	VidChapters-7M(テスト)	URL	大きさ
HowTo100M + VidChapters-7M	120.5	Drive	1.1GB

ビデオチャプターのグラウンディングのタスクについて、以下の Moment-DETR チェックポイントをリリースし、それに対応する R@10s のパフォーマンスを報告する。

学習データ	VidChapters-7M(テスト)	URL	大きさ
VidChapters-7M	21.8	Drive	0.9GB

訓練と評価

特に断りのない限り、以下のスクリプトで事前訓練されたチェックポイントをロードするには `--load=<CHECKPOINT>` を使用し、評価は以下と同じスクリプトで `--eval` を指定して行うことができます。

トレーニング実行のほとんどは、80GB のメモリを搭載した A100 GPU を使用していることに注意してください。低メモリ GPU を使用している場合は、バッチサイズを調整する必要があるかもしれません。

また、BLIP-2 ベースのスクリプトを使用するには、対応するデータセットから生の動画をダウンロードし、動画 ID と動画パスを対応付けた `video_paths.json` ファイルを用意する必要がある。

HowTo100M で事前学習された Vid2Seq

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env dvc.py --epochs=5 --fraction_wa
```

動画チャプター生成

Vid2Seq を実行するには,

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env dvc.py --epochs=10 --lr=3e-4 --
```

論文で報告された複数のベースラインも `args.py` で見つけることができます。例えば、`--no_speech` や `--no_video` で視覚入力や音声入力のみを使用したり、`--gen_asr` で ASR のみを使用してトレーニングしたりすることができます。

PDVC を実行するには,

```
cd PDVC
conda activate PDVC_env
python train.py --cfg_path cfgs/chapters_clip_pdvc.yml --gpu_id=0 --epoch=5 --no_self_iou --lr=1
```

PDVC によるテスト推論は、同じスクリプトを使って、config にテストデータへの評価パスを設定し、`--load=<CHECKPOINT>` と `--epoch=0` というパラメーターを設定することで行うことができる。

テキスト・タイリング＋LLaMA ゼロショット・ベースラインの場合、

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env zs_speechvcg.py --combine_datase
```

前のコマンドに `--random` を渡すと、ランダム・ベースラインが実行される。

ショット検出＋BLIP-2 ゼロショットベースラインの場合、

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env zs_visualvcg.py --combine_datase
```

グラウンドトゥルースバウンダリーによるビデオチャプターの生成

Vid2Seqの場合，

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env vc.py --epochs=20 --lr=3e-4 --s
```

LLaMAのゼロショットベースラインの場合、

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env vc.py --model_name=<MODEL_DIR>/
```

前のコマンドに `--random` を渡すと、ランダム・ベースラインが実行される。

BLIP-2ゼロショットベースラインの場合、

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env vc.py --model_name=Salesforce/b
```

Video Chapter Generation Grounding

Moment-DETR の場合，

```
cd moment_detr
bash moment_detr/scripts/chapters.sh --max_v_l=1200 --downsample --clip_length=3 --lr=3e-4 --n_e
```

Moment-DETRによる推論は、スクリプト `moment_detr/scripts/chapters_inference.sh` 、同じパラメーター、パラメーター `--resume=<CHECKPOINT>` で実行できる。

CLIPゼロショットベースラインの場合，

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env zs_vcgr.py --save_dir=chapters_v
```

BERTゼロショットベースラインの場合，

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env zs_vcgr.py --save_dir=chapters_v
```

ランダムゼロショットベースラインの場合，

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env zs_vcgr.py --save_dir=chapters_v
```

高密度動画キャプショニング

YouCook/ViTT による Vid2Seq の場合,

```
python -m torch.distributed.launch --nproc_per_node 8 --use_env dvc.py --epochs=40 --lr=3e-4 --
python -m torch.distributed.launch --nproc_per_node 8 --use_env dvc.py --epochs=20 --lr=3e-4 --
```

ゼロショット評価は、 `--load=<CHECKPOINT> --eval` の引数を使用して、評価のためにVidChapters-7Mで事前に訓練されたチェックポイントをロードすることで簡単に行うことができます。

YouCook/ViTT による PDVC の場合,

```
cd PDVC
conda activate PDVC_env
python train.py --cfg_path=cfgs/yc2_clip_pdvc.yml --gpu_id=0
python train.py --cfg_path=cfgs/vitt_clip_pdvc.yml --gpu_id=0
```

事前学習されたPDVCチェックポイントをロードするには、 `--load=<CHECKPOINT>` と `--load_vocab data/vocabulary_allchapters.json` というパラメータを設定します。
PDVC によるテスト推論は、同じスクリプトを使って、config にテストデータへの評価パスを設定し、 `--load=<CHECKPOINT>` と `--epoch=0` というパラメーターを設定することで行うことができる。

デモ

事前に訓練された Vid2Seq モデル（ビデオチャプター生成または高密度ビデオキャプション用）を任意のビデオで実行するには、まず、次のコマンドで ASR を抽出する必要があります：

```
conda activate whisperX_env
python demo_asr.py --video_example=<VIDEO_PATH> --asr_example <OUTPUT_ASR_PATH> --combine_datase
```

そしてモデル推論を実行することができる：

```
python demo_vid2seq.py --load=<CHECKPOINT> --video_example=<VIDEO_PATH> --asr_example <OUTPUT_AS
```

ライセンス

このコードは MIT ライセンスで公開されている。論文で使用したデータセットのライセンスは以下のリンクから入手できる： VidChapters-7M、HowTo100M、YouCook2、ViTT。

引用

もしこの仕事が役に立ったと思われるなら、このリポジトリに星をつけ、私たちの論文を以下のように引用してください。

```
@inproceedings{yang2023vidchapters,
title={VidChapters-7M: Video Chapters at Scale},
author={Antoine Yang and Arsha Nagrani and Ivan Laptev and Josef Sivic and Cordelia Schmid},
booktitle={NeurIPS},
year={2023}}
```