

Porównanie modeli przetwarzających zdjęcia na tekst

Grzyb Gerard
Jamróz Mateusz

Abstrakt

Projekt polega na porównaniu modeli przekształcających obrazy na tekst z wykorzystaniem transformatorów dostępnych na platformie Hugging Face. Rezultatem pracy jest narzędzie generujące trzy krótkie opisy filmu na podstawie analizy wybranych klatek. Każdy opis tworzony jest przez inny model, co pozwala na porównanie ich skuteczności i stylu działania.

Wstęp

Celem projektu jest porównanie trzech modeli typu "image-to-text". W ramach projektu opracowano proces podziału filmu na klatki, zastosowano transformatory do generowania opisów tych klatek, oraz opracowano metodę podsumowania filmu w formie streszczenia wygenerowanych opisów. Narzędzie zostało wdrożone na platformę Hugging Face przy użyciu funkcji "Spaces", umożliwiając łatwy dostęp do wyników analizy.

Do realizacji projektu wykorzystano technologie takie jak Docker, Python oraz narzędzia dostępne na platformie Hugging Face(Gradio).

Opis

Głównym tematem opisywanego tematu jest wykorzystanie modeli dostępnych na platformie the Hugging Face, oraz użycie spaces jako wdrożenie do środowiska produkcyjnego.

Wykorzystane zostały trzy modele dostępne na platformie:

- nlpconnect/vit-gpt2-image-captioning
- noamrot/FuseCap
- Salesforce/blip-image-captioning-large

```
# Model 1: ViT-GPT2
model1 = VisionEncoderDecoderModel.from_pretrained("nlpconnect/vit-gpt2-image-captioning").to(device)
feature_extractor1 = ViTImageProcessor.from_pretrained("nlpconnect/vit-gpt2-image-captioning")
tokenizer1 = AutoTokenizer.from_pretrained("nlpconnect/vit-gpt2-image-captioning")

# Model 2: FuseCap
processor2 = BlipProcessor.from_pretrained("noamrot/FuseCap")
model2 = BlipForConditionalGeneration.from_pretrained("noamrot/FuseCap").to(device)

# Model 3: BLIP Large
processor3 = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-large")
model3 = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large").to(device)
```

Skomponowane wyżej trzy modele z naszą aplikacją umożliwią generowanie opisów dla klatek z przesłanych filmów wideo. Rozwiązanie zostało Interfejs oparty jest na frameworku Gradio, co umożliwia łatwe przesyłanie plików wideo oraz przeglądanie wyników.

Funkcjonalności

Generowanie opisów z klatki:

ViT-GPT2

Model ViT-GPT2 jest połączeniem dwóch zaawansowanych komponentów: Vision Transformer (ViT) jako enkodera obrazu i GPT-2 jako dekodera tekstowego. Wykorzystuje architekturę VisionEncoderDecoder, która umożliwia generowanie opisów tekstowych na podstawie obrazów.

Przetwarza obrazy, dzieląc je na mniejsze fragmenty (patche), które są analizowane jako sekwencje cech. Generuje płynne i zrozumiałe opisy na podstawie zakodowanych cech obrazu. Model jest dobrze dostosowany do ogólnych zadań opisu obrazów, ale jego skuteczność może być ograniczona przez brak specjalizacji w danych treningowych

```
import requests
from transformers import VisionEncoderDecoderModel, ViTImageProcessor, AutoTokenizer
import torch
from PIL import Image

model1 = VisionEncoderDecoderModel.from_pretrained("nlpconnect/vit-gpt2-image-captioning")
feature_extractor1 = ViTImageProcessor.from_pretrained("nlpconnect/vit-gpt2-image-captioning")
tokenizer1 = AutoTokenizer.from_pretrained("nlpconnect/vit-gpt2-image-captioning")

device1 = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model1.to(device1)

max_length = 16
num_beams = 4
gen_kwargs = {"max_length": max_length, "num_beams": num_beams}

def image_to_text_model_1(image_url):
    raw_image = Image.open(requests.get(image_url, stream=True).raw).convert('RGB')

    pixel_values = feature_extractor1(images=[raw_image], return_tensors="pt").pixel_values
    pixel_values = pixel_values.to(device1)

    output_ids = model1.generate(pixel_values, **gen_kwargs)

    preds = tokenizer1.batch_decode(output_ids, skip_special_tokens=True)
    preds = [pred.strip() for pred in preds]
    return preds

def bytes_to_text_model_1(bts):
    pixel_values = feature_extractor1(images=[bts], return_tensors="pt").pixel_values
    pixel_values = pixel_values.to(device1)

    output_ids = model1.generate(pixel_values, **gen_kwargs)

    preds = tokenizer1.batch_decode(output_ids, skip_special_tokens=True)
    preds = [pred.strip() for pred in preds]
    return preds[0]
```

FuseCap

FuseCap to model oparty na architekturze BLIP (Bootstrapped Language-Image Pretraining), który został dodatkowo ulepszony w celu lepszego rozumienia kontekstowego między obrazami a tekstami. Wprowadza lepsze sposoby łączenia informacji wizualnych i tekstowych, co pozwala na bardziej precyzyjne opisy. Lepsza adaptacja do danych w kontekście specyficznych zastosowań, np. opisów dla obrazów medycznych czy technicznych.

```
import requests
from PIL import Image
from transformers import BlipProcessor, BlipForConditionalGeneration
import torch

device2 = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
processor2 = BlipProcessor.from_pretrained("noamrot/FuseCap")
model2 = BlipForConditionalGeneration.from_pretrained("noamrot/FuseCap").to(device2)

def image_to_text_model_2(img_url):
    raw_image = Image.open(requests.get(img_url, stream=True).raw).convert('RGB')
    text = "a picture of "
    inputs = processor2(raw_image, text, return_tensors="pt").to(device2)

    out = model2.generate(**inputs, num_beams = 3)
    return processor2.decode(out[0], skip_special_tokens=True)

def bytes_to_text_model_2(byts):
    text = "a picture of "
    inputs = processor2(byts, text, return_tensors="pt").to(device2)

    out = model2.generate(**inputs, num_beams = 3)
    return processor2.decode(out[0], skip_special_tokens=True)
```

BLIP Large

BLIP Large to powiększona wersja modelu BLIP, zaprojektowana specjalnie z myślą o generowaniu precyzyjnych i bogatych opisów obrazów. Model wykorzystuje większą liczbę parametrów, co zwiększa jego możliwości.

```

import requests
from PIL import Image
from transformers import BlipProcessor, BlipForConditionalGeneration

processor3 = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-large")
model3 = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large")

def image_to_text_model_3(img_url):
    raw_image = Image.open(requests.get(img_url, stream=True).raw).convert('RGB')
    text = "a picture of"
    inputs = processor3(raw_image, text, return_tensors="pt")
    inputs = processor3(raw_image, return_tensors="pt")

    out = model3.generate(**inputs)
    return processor3.decode(out[0], skip_special_tokens=True)

def bytes_to_text_model_3(byts):
    text = "a picture of"
    inputs = processor3(byts, text, return_tensors="pt")
    inputs = processor3(byts, return_tensors="pt")

    out = model3.generate(**inputs)
    return processor3.decode(out[0], skip_special_tokens=True)

```

Prezentacja wyników:

Wyniki są prezentowane w formie listy opisów, pogrupowanych według modeli przy użyciu summarizatora.

```

def summarize_articles(article1_content, article2_content, article3_content):
    article1_content = list(set([x for x in article1_content if x]))
    content1 = " ".join(article1_content)
    summary1 = summarizer(content1, max_length=130, min_length=30, do_sample=False)
    print("Summary for Article 1:")
    print(summary1)

    article2_content = list(set([x for x in article2_content if x]))
    print(article2_content)
    content2 = " ".join(article2_content)
    summary2 = summarizer(content2, max_length=130, min_length=30, do_sample=False)
    print("Summary for Article 2:")
    print(summary2)

    article3_content = list(set([x for x in article3_content if x]))
    content3 = " ".join(article3_content)
    summary3 = summarizer(content3, max_length=130, min_length=30, do_sample=False)
    print("Summary for Article 3:")
    print(summary3)

```

Dzielenie na klatki

Aplikacja wykorzystuje bibliotekę OpenCV, aby podzielić video co określoną ilość klatek, a następnie poddać wybraną klatkę analizie. Po wykonanym procesie następuje analiza wyniku, a następnie dodanie do sumy analiz, które później będą wykorzystywane do analizy.

```

def FrameCapture(path):
    vidObj = cv2.VideoCapture(path)
    count = 0
    success = True

    article1_content = []
    article2_content = []
    article3_content = []

    while success:
        success, image = vidObj.read()
        if not success:
            break

        if count % 60 == 0:
            model_1_result = bytes_to_text_model_1(image)
            article1_content.append(model_1_result)
            model_2_result = bytes_to_text_model_2(image)
            article2_content.append(model_2_result)
            model_3_result = bytes_to_text_model_3(image)
            article3_content.append(model_3_result)

        count += 1

    return article1_content, article2_content, article3_content

```

Summaryzacja tekstu

Do podsumowania tekstu wykorzystaliśmy model facebook/bart-large-cnn. Jest to model opracowany przez Facebook AI, który posiada architekturę kodera-dekodera. Koder → Przetwarza wejściowy tekst w sposób dwukierunkowy, analizując kontekst zarówno przed, jak i po każdej pozycji w sekwencji. Dekoder → Generuje wynikowy tekst w sposób autoregresywny (token po tokenie), co jest szczególnie przydatne przy generacji tekstu.

Danymi wejściowymi są przeanalizowane klatki, które następnie są łączone w tekst, a następnie poddane analizie przez model.

```

import cv2
from transformers import pipeline

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

> def FrameCapture(path): ...

def summarize_articles(article1_content, article2_content, article3_content):
    article1_content = list(set([x for x in article1_content if x]))
    content1 = " ".join(article1_content)
    summary1 = summarizer(content1, max_length=130, min_length=30, do_sample=False)
    print("Summary for Article 1:")
    print(summary1)

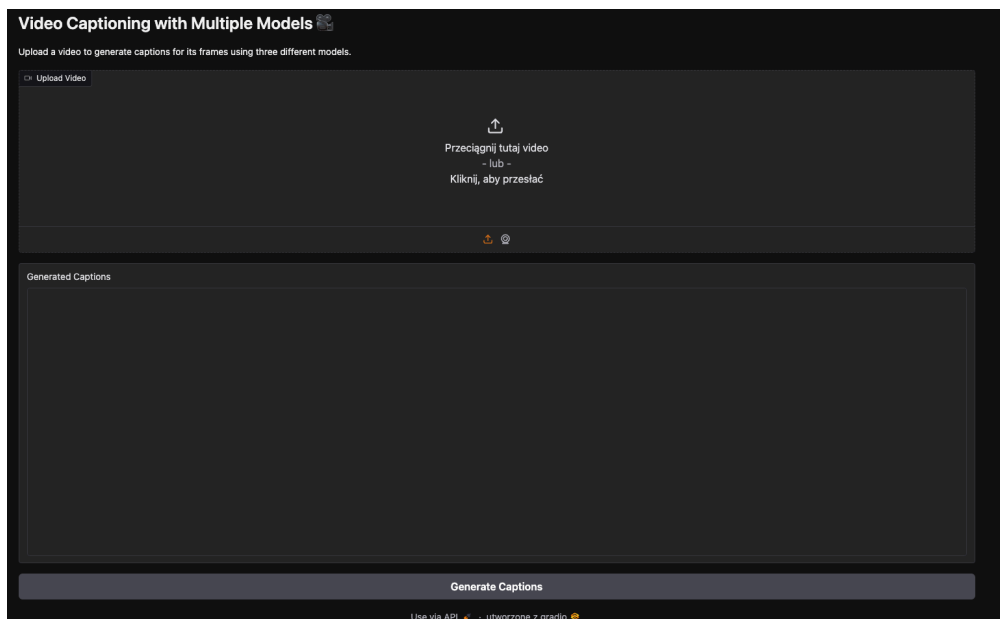
    article2_content = list(set([x for x in article2_content if x]))
    print(article2_content)
    content2 = " ".join(article2_content)
    summary2 = summarizer(content2, max_length=130, min_length=30, do_sample=False)
    print("Summary for Article 2:")
    print(summary2)

    article3_content = list(set([x for x in article3_content if x]))
    content3 = " ".join(article3_content)
    summary3 = summarizer(content3, max_length=130, min_length=30, do_sample=False)
    print("Summary for Article 3:")
    print(summary3)

```

Przesyłanie wideo

Jako środowisko wdrożeniowe wykorzystaliśmy spaces na the hugging face, dzięki któremu użytkownik jest w stanie sam przesłać wideo, a następnie klatki z filmu są analizowane, a opisy generowane dla wybranych klatek (co 20 lub 60 klatka). Ograniczeniem jest darmowy zakres wdrożonego modelu spaces, który działa bardzo wolno. Dostępny adres pod <https://huggingface.co/spaces/matjarm/model-comparison>.



Porównanie i analiza

Do porównania wykorzystaliśmy trzy materiały wideo o różnej długości i stopniu skomplikowania.

Pierwszym jest prosty piętnastosekundowy wyścig dwóch samochodów.



Model 1

Picture of a man stands on a concrete sidewalk next to a blue car and a yellow car, with a brown wall in the background and a white sign nearby', 'a picture of two cars, one blue and one yellow, are parked in a lot surrounded by green grass and a brown wall a white sign is visible in the background', 'a picture of a yellow sports car is parked in front of a brown wall with various signs, including a purple sign, a white sign, and a white and blue sign the car is surrounded', 'a picture of a yellow sports car is parked in front of a brown wall, surrounded by green grass and a blue and purple sign a blue car is also visible in the background', 'a picture of a yellow sports car parked in a lot surrounded by green grass and a brown wall, with a blue and purple sign in the background.

Model 2

A picture of two cars, one blue and one yellow, are parked in a lot surrounded by green grass and a brown wall. A picture of a man stands next to a yellow sports car with a white sign in the background and a white smoke billowing from the car. A man stands on the side of the road next to two cars with a blue and purple sign and white sign. A woman stands in front of a car with a white sign and a blue and purple sign in the backyard.

Model 3

There are two cars that are racing on the track with smoke coming out of them. There are many signs on the side of the road that indicate time on the highway. People are walking up and down the stairs of a stadium, people are watching a baseball game.

Drugim filmem jest fragment filmu "Shrek" o długości dwóch minut.



Model 1

A man standing next to a bunch of trees. a man standing around a fire hydrant a bird that is standing in the water a statue of a man in the middle of a forest. a person with a cigarette in their mouth a blue bottle sitting on top of a blue wall. A picture of a group of people, one wearing a black shirt and another wearing a white hat, stand around a circular object with the word love written on it one person 's arm is. A picture of a black bird perches on a wooden fence, with a black hole in the background

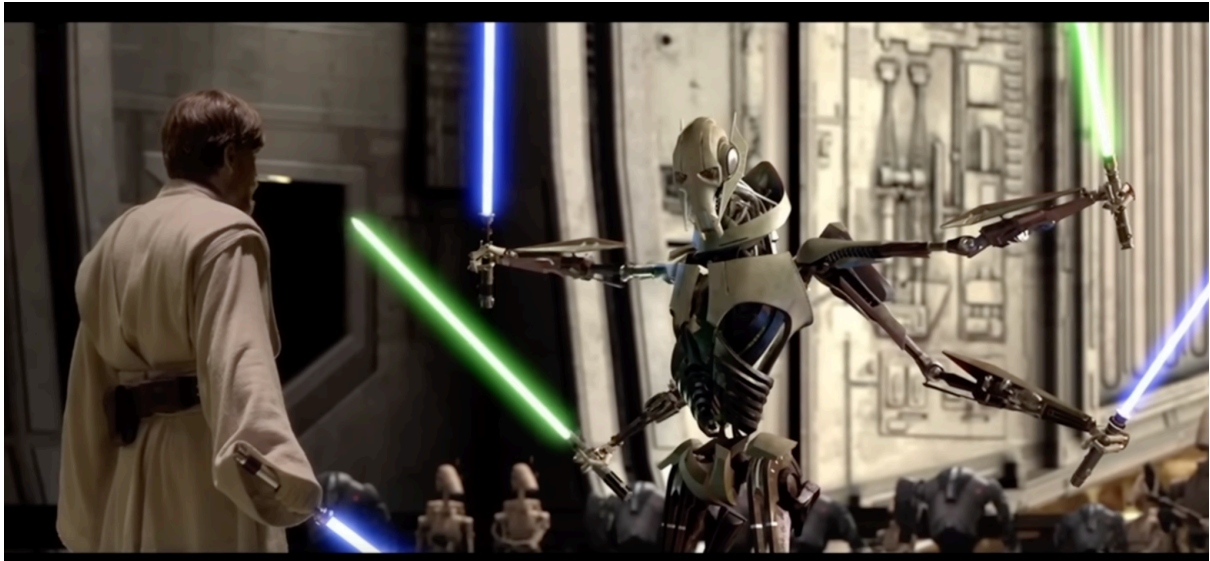
Model 2

Picture of a serene forest scene featuring a large tree and a wooden door, with a fountain adding a touch of tranquility to the tranquil setting. A picture of a man stands in the water surrounded by a variety of fish, including a small white fish, a blue fish, and a green fish, while a white flower adds a touch. a picture of a group of people, one wearing a black shirt and another wearing a white hat, stand around a circular object with the word love written on it one person 's arm is visible in the foreground.

Model 3

Arafed image of a group of umbrellas that are in the grass there is a bear that is sitting in the water with lily pads. arafed scare with a scare head and a scare face on his head. several people in a jungle with a light on their head with torches in their hands. a close up of a cartoon character holding a wooden pole someone is drawing a smiley face on the sidewalk.

Trzecią analizowaną częścią jest urywek filmu “Star Wars” o długości pięciu minut.



Model 1

A man standing in front of a wall with a camera a close up view of a motorcycle on display. A man in a suit is playing a video game with a bat a man is performing a trick on a fire hydrant. A woman is standing on a bench holding a candle on a black and white photo of a set of stairs. A man standing in a room next to a wall a statue of a man riding on the back of an elephant. a picture of a man with short hair wearing a gray shirt stands in front of a black wall, with a blurry face visible in the background', 'a picture of a person wearing a white helmet and black belt stands in front of a white building, holding a green light in the background, there is a black and white wall and a black', "a picture of a person 's legs, one white and one black and white, are seen in close - up on a black floor, with a white sock visible in the foreground", 'a picture of a woman with long hair stands in front of a blue wall, holding a white hand up to her face. a picture of a person wearing a gray and white shirt stands in front of a concrete and gray wall, with a black strap visible in the foreground', 'a picture of a white wall serves as the backdrop for a scene featuring a pair of white scissors and a sharp blade, possibly belonging to a character in a video game'. A picture of a blue light shines in the background as a person wearing a gas mask stands in front of it

Model 2

A picture of a man with a white and blue and black tape, a picture of person who has never played a video game, and a woman who has never played a game. A picture of man with a video game that has never been played, a man with a black and white board, and a woman who's always played a video game with her hands on her knees. A man with a white and black

Model 3

A close up of a person holding a light saber in a dark room is a close up of a person riding a dinosaur with a sword. A man in a star wars suit is a man riding a horse in a cave. A close up is of a man standing in front of a car with a large head and a large vehicle. A close up is of a large animal that is walking on a rock with a view of the earth. A person is sitting on the ground with a pink light in their hand with a batman arks the dark knight.

Analiza

Analizując powyższe summaryzacje można dostrzec, że pierwszy model posiada tendencję do fantazjowania, drugi jest w stanie minimalnie rozpoznać zaprezentowany obraz, aczkolwiek trzeci radzi sobie z tym już stosunkowo dobrze.

Podsumowanie

.Projekt opisuje proces porównania trzech modeli transformatorych do generowania opisów obrazów na podstawie klatek z filmów. Realizacja obejmowała analizę i implementację trzech modeli: **ViT-GPT2**, **FuseCap**, oraz **BLIP Large**, dostępnych na platformie Hugging Face. Utworzone narzędzie pozwala na automatyczne tworzenie opisów dla klatek wideo, ułatwiając ocenę różnic w stylu oraz skuteczności działania modeli.

Aplikacja wykorzystuje bibliotekę OpenCV do podziału filmów na klatki, a następnie stosuje transformatort do analizy wizualnej i generacji opisów. Wyniki są porządkowane w oparciu o zastosowany model, a podsumowania generowane są przy użyciu modelu **facebook/bart-large-cnn**. Narzędzie udostępniono za pośrednictwem platformy **Hugging Face Spaces**, umożliwiając użytkownikom łatwe przesyłanie filmów i przeglądanie wyników.

Kluczowym efektem projektu jest intuicyjne narzędzie z interfejsem opartym na **Gradio**, które pozwala na analizę wizualno-tekstową. Jednak ograniczenia, takie jak wydajność darmowego wdrożenia, wpływają na szybkość przetwarzania. Projekt dostarcza cennych informacji na temat zastosowań modeli typu "image-to-text", oferując platformę do dalszych badań i ulepszeń.

Bibliografia

<https://opencv.org/>

https://huggingface.co/models?pipeline_tag=text-to-speech

<https://www.youtube.com/>

<https://www.imagetotext.info/>