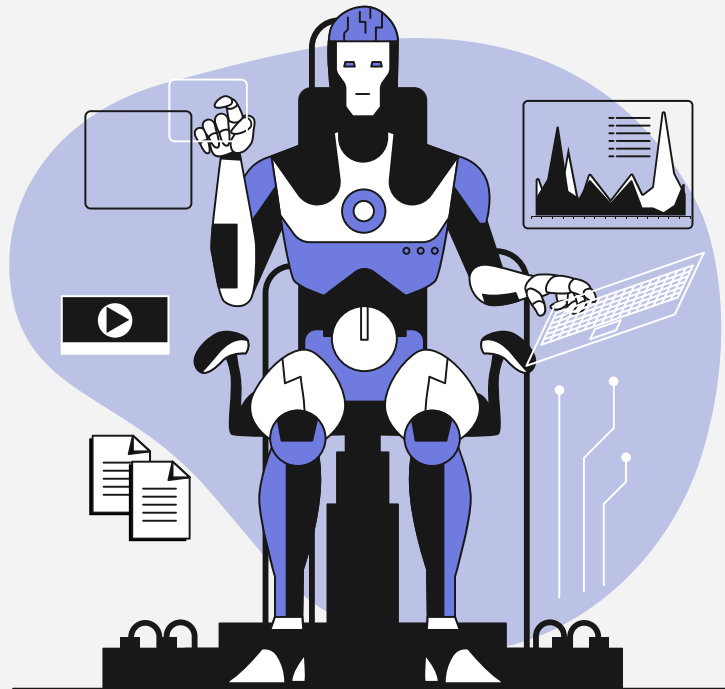


Vision Language Models



subtelne wprowadzenie
w świat modeli multimodalnych





Piotr Tynecki

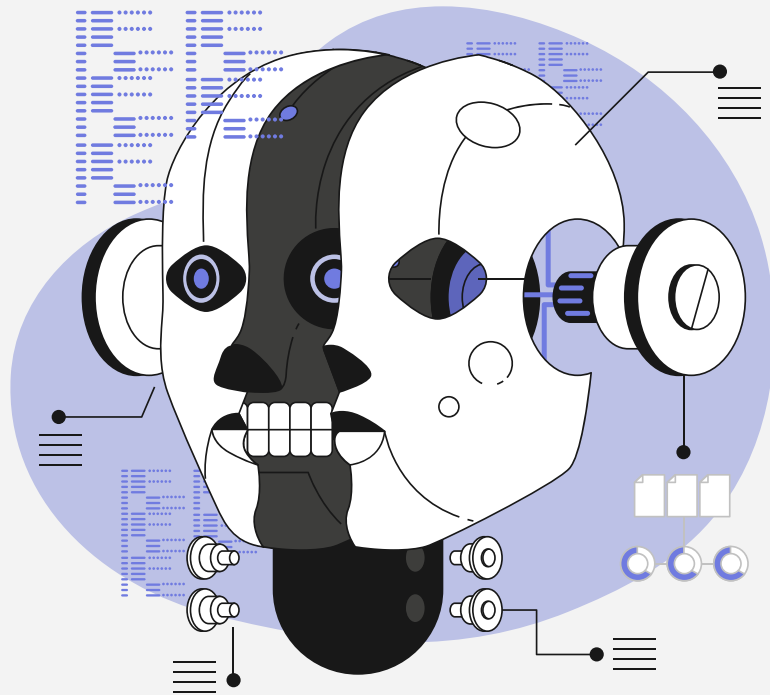
Inżynier SI dla sektora “Life Science”
(biotech, wildlife conservation & healthcare)

- Klasyczne Uczenie Maszynowe
- Przetwarzanie Języka Naturalnego
- Komputerowa Wizja
- ... a od niedawna także LLM/RAG



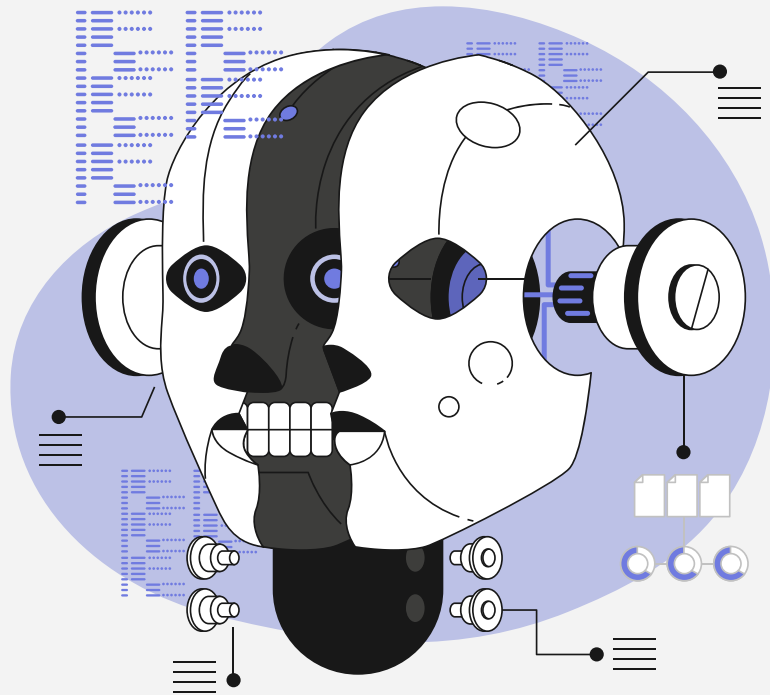


- Modele językowe i wizyjne
- Florence-2
- Phi-3.5-vision
- Fine-tuning
- Wnioski końcowe





- **Modele językowe i wizyjne**
- Florence-2
- Phi-3.5-vision
- Fine-tuning
- Wnioski końcowe



**Konkurs ImageNet z 2012 roku,
zwycięstwo sieci konwolucyjnych AlexNet**
silny sygnał nadejścia ery głębokich sieci neuronowych

**Publikacja "Attention Is All You Need" z 2017 roku,
nowa architektura Transformer**
efektywny mechanizm uwagi (atencji) i wielowarstwowy perceptron
uwzględniające relacje między słowami i kontekst znaczeniowy

"Małe" i Duże Modele Językowe
przewidywanie następnego *tokenu* na podstawie kontekstu
(sekwencji); modele reprezentacyjne i modele generatywne



Modele językowe

Dwie główne klasy modeli językowych:



Modele reprezentacyjne (np. BERT, RoBERT, DeBERT) z reguły mają do jednego mld parametrów.

Oparte na architekturze **encoder-only** (embeddingach). Przeznaczone do rozpoznawania nazw własnych (NER) czy klasyfikacji dokumentów.



Modele generatywne (np. T5, GPT, Llama) cechuje wiele mld parametrów (wag).

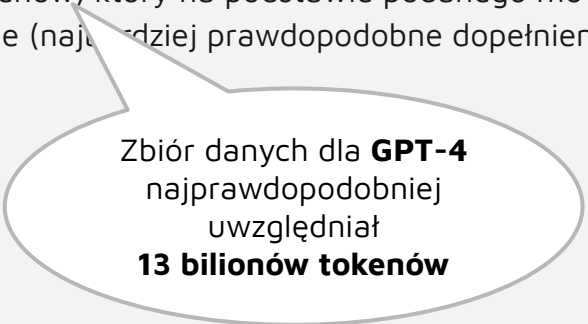
Oparte najczęściej na architekturze **encoder-decoder** lub **decoder-only**. Przeznaczone do generacji tekstu tj. streszczenia, tłumaczenia czy odpowiadania na pytania.



Duży Model Językowy (Large Language Model, LLM) to model językowy, wytrenowany na ogromnych zbiorach danych tekstowych liczących setki miliardów tokenów, który na podstawie podanego mu wejściowego tekstu, potrafi generować tekst wyjściowy słowo po słowie (najbardziej prawdopodobne dopełnienie).



Duży Model Językowy (Large Language Model, LLM) to model językowy, wytrenowany na ogromnych zbiorach danych tekstowych liczących setki miliardów tokenów, który na podstawie podanego mu wejściowego tekstu, potrafi generować tekst wyjściowy słowo po słowie (najbardziej prawdopodobne dopełnienie).



Zbiór danych dla **GPT-4**
najprawdopodobniej
uwzględnił
13 bilionów tokenów



Duży Model Językowy (Large Language Model, LLM) to model językowy, wytrenowany na ogromnych zbiorach danych tekstowych liczących setki miliardów tokenów, który na podstawie podanego mu wejściowego tekstu, potrafi generować tekst wyjściowy słowo po słowie (najbardziej prawdopodobne dopełnienie).

“Mały” Model Językowy (Small Language Model, SLM) to wersje modeli NLP o mniejszej liczbie parametrów, zazwyczaj liczących się w milionach zamiast miliardach, co przekłada się na mniejsze zapotrzebowanie na moc obliczeniową. Są bardziej zoptymalizowane pod kątem wydajności i kosztów.



Duży Model Językowy (Large Language Model, LLM) to model językowy, wytrenowany na ogromnych zbiorach danych tekstowych liczących setki miliardów tokenów, który na podstawie podanego mu wejściowego tekstu, potrafi generować tekst wyjściowy słowo po słowie (najbardziej prawdopodobne dopełnienie).

“Mały” Model Językowy (Small Language Model, SLM) to wersje modeli NLP o mniejszej liczbie parametrów, zazwyczaj liczących się w milionach zamiast miliardach, co przekłada się na mniejsze zapotrzebowanie na moc obliczeniową. Są bardziej zoptymalizowane pod kątem wydajności i kosztów.

SLM szkoli się na bardziej ukierunkowanych zestawach danych dostosowanych do indywidualnych potrzeb (określonego problemu), co sprawia, że są rekomendowane dla mniejszych firm i przedsiębiorstw o ograniczonych zasobach IT. Kluczowe dla optymalizacji SLM są techniki kompresji modelu, destylacji wiedzy oraz uczenia transferowego.



Duży Model Językowy (Large Language Model, LLM) to model językowy, wytrenowany na ogromnych zbiorach danych tekstowych liczących setki miliardów tokenów, który na podstawie podanego mu wejściowego tekstu, potrafi generować tekst wyjściowy słowo po słowie (najbardziej prawdopodobne dopełnienie).

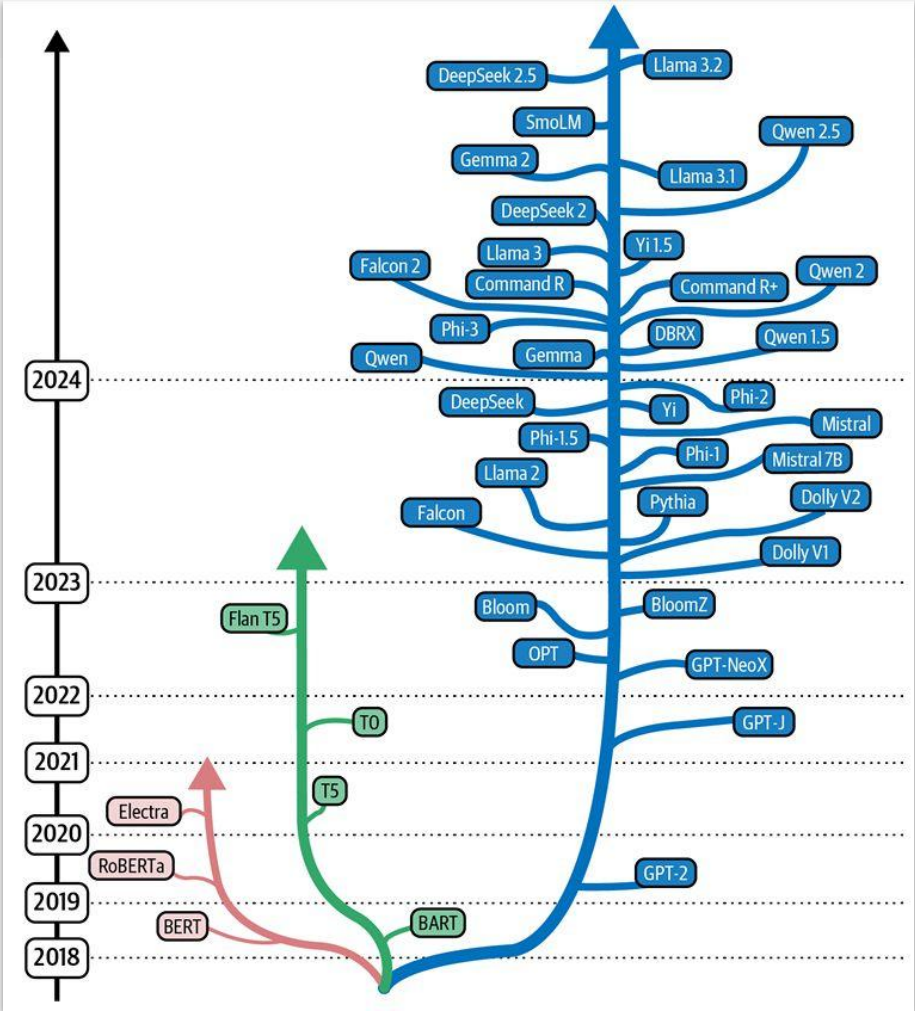
“Mały” Model Językowy (Small Language Model, SLM) to wersje modeli NLP o mniejszej liczbie parametrów, zazwyczaj liczących się w milionach zamiast miliardach, co przekłada się na mniejsze zapotrzebowanie na moc obliczeniową. Są bardziej zoptymalizowane pod kątem wydajności i kosztów.

SLM szkoli się na bardziej ukierunkowanych zestawach danych dostosowanych do indywidualnych potrzeb (określonego problemu), co sprawia, że są rekomendowane dla mniejszych firm i przedsiębiorstw o ograniczonych zasobach IT. Kluczowe dla optymalizacji SLM są techniki kompresji modelu, destylacji wiedzy oraz uczenia transferowego.

Gdy uczenie modeli odbywa się z wykorzystaniem korpusów instrukcji (poleceń, zadań z rozwiązaniami) oraz korpusów preferencji, w efekcie otrzymujemy **modele konwersacyjne** (tj. ChatGPT, BIELIK, PLLuM).



Modele językowe



Rys 1. Rozwój otwartych modeli językowych. **Czerwony** - encode-only, **Zielony** - encoder-decoder, **Niebieski** - decoder-only.

Źródło: Julien Chaumond, CTO Hugging Face



"Małe" i duże modele językowe

Model	Data wydania	Deweloper	Parametry (B)	Długość kontekstu (k)	Licencja
<u>Llama 3.2</u>	09.2024	Meta AI	1, 3, 11, 90	128	Llama 3.2 Community License
<u>Falcon 2</u> (model SSLM)	08.2024	Tech. II	*7, 11, 40, 180	8	TII Falcon License
<u>BIELIK v2</u>	08.2024	SpeakLeash	11	32	Apache 2.0
<u>Phi-3.5</u>	07.2024	Microsoft	3.8, 4.2, 6.6	128	MIT
<u>Gemma 2</u>	06.2024	Google	2, 9, 27	8	Gemma License
<u>Nemotron-4</u>	06.2024	NVIDIA	4, 8, 15, 22, 340	4	NVIDIA Open Model License
<u>Mixtral</u>	04.2024	Mistral AI	7, 22	32	Apache 2.0

Tab 1. Wybrane otwarte modele językowe wydane w roku 2024.

Model	Aktualizacja	Deweloper
Claude 3.5	10.2024	Anthropic
Gemini Pro 1.5	09.2024	Google DeepMind
OpenAI o1	09.2024	Open AI
Grok 2 (beta)	08.2024	xAI
GPT-4o	05.2024	Open AI

Tab 2. Wybrane komercyjne modele językowe dostępne poprzez API.

Modele multimodalne

Rozwój Dużych Modeli Językowych pociągnął za sobą rozwój **Modeli Multimodalnych (MLLM)**, czyli takich które mogą przyjąć na wejściu więcej niż jeden typ danych. W tym przypadku, omówimy możliwość analizy obrazu i tekstu jednocześnie.



Modele multimodalne

Rozwój Dużych Modeli Językowych pociągnął za sobą rozwój **Modeli Multimodalnych (MLLM)**, czyli takich które mogą przyjąć na wejściu więcej niż jeden typ danych. W tym przypadku, omówimy możliwość analizy obrazu i tekstu jednocześnie.

Architektura MLLM bazuje na trzech komponentach:

1. **Enkoder wizyjny** (enkoder obrazu)
2. **Adapter wizja-tekst**
3. Duży Model Językowy (LLM)



Modele multimodalne

Rozwój Dużych Modeli Językowych pociągnął za sobą rozwój **Modeli Multimodalnych (MLLM)**, czyli takich które mogą przyjąć na wejściu więcej niż jeden typ danych. W tym przypadku, omówimy możliwość analizy obrazu i tekstu jednocześnie.

Architektura MLLM bazuje na trzech komponentach:

1. **Enkoder wizyjny** (enkoder obrazu)
 - o model wizyjny (np. **Vision Transformer**), dzieli obraz na mniejsze fragmenty, następnie przekształca te fragmenty w wektory (*embeddingi*), dając ostatecznie postać wielowymiarowych wektorów liczb (*tokenów wizyjnych*);
2. **Adapter wizja-tekst**
3. Duży Model Językowy (LLM)



Modele multimodalne

Rozwój Dużych Modeli Językowych pociągnął za sobą rozwój **Modeli Multimodalnych (MLLM)**, czyli takich które mogą przyjąć na wejściu więcej niż jeden typ danych. W tym przypadku, omówimy możliwość analizy obrazu i tekstu jednocześnie.

Architektura MLLM bazuje na trzech komponentach:

1. **Enkoder wizyjny** (enkoder obrazu)
 - model wizyjny (np. **Vision Transformer**), dzieli obraz na mniejsze fragmenty, następnie przekształca te fragmenty w wektory (*embeddingi*), dając ostatecznie postać wielowymiarowych wektorów liczb (*tokenów wizyjnych*);
2. **Adapter wizja-tekst**
 - komponent przekształcający wizualną reprezentację obrazu na reprezentację akceptowaną przez LLM, czyli sekwencję wektorów słów (tokenów). Istotna dla zachowania szczegółowości jest liczba tokenów wizyjnych oraz rozdzielczość obrazu.
3. Duży Model Językowy (LLM)



Rozwój Dużych Modeli Językowych pociągnął za sobą rozwój **Modeli Multimodalnych (MLLM)**, czyli takich które mogą przyjąć na wejściu więcej niż jeden typ danych. W tym przypadku, omówimy możliwość analizy obrazu i tekstu jednocześnie.

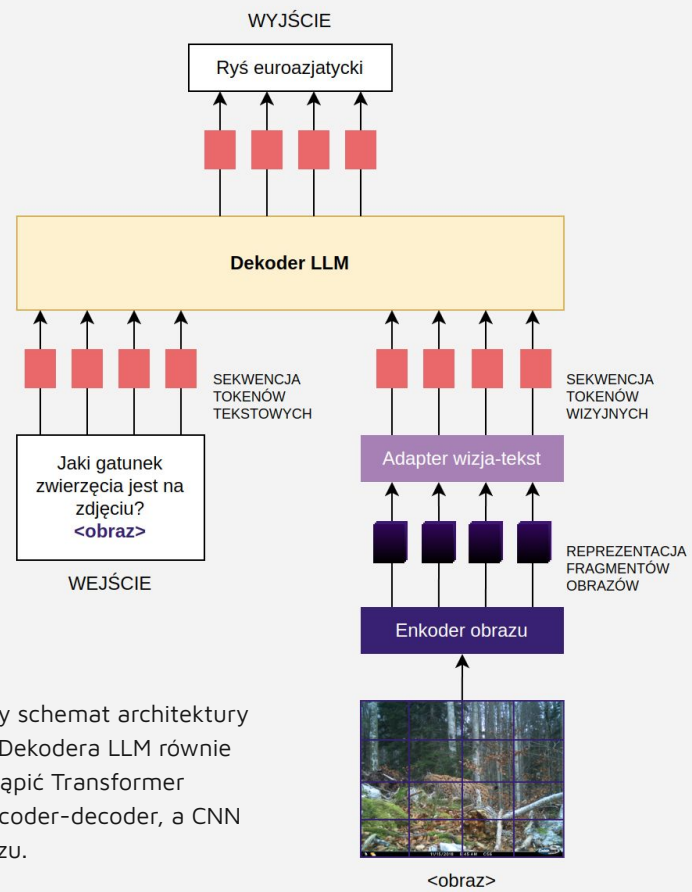
Architektura MLLM bazuje na trzech komponentach:

1. **Enkoder wizyjny** (enkoder obrazu)
 - o model wizyjny (np. **Vision Transformer**), dzieli obraz na mniejsze fragmenty, następnie przekształca te fragmenty w wektory (*embeddingi*), dając ostatecznie postać wielowymiarowych wektorów liczb (*tokenów wizyjnych*);
2. **Adapter wizja-tekst**
 - o komponent przekształcający wizualną reprezentację obrazu na reprezentację akceptowaną przez LLM, czyli sekwencję wektorów słów (tokenów). Istotna dla zachowania szczegółowości jest liczba tokenów wizyjnych oraz rozdzielczość obrazu.
3. Duży Model Językowy (LLM)



MLLM otworzyły nowy rozdział dla modeli konwersacyjnych, czyli umiejętność wizualnego odpowiadania na pytania (**Visual Question-Answering (VQA)**)



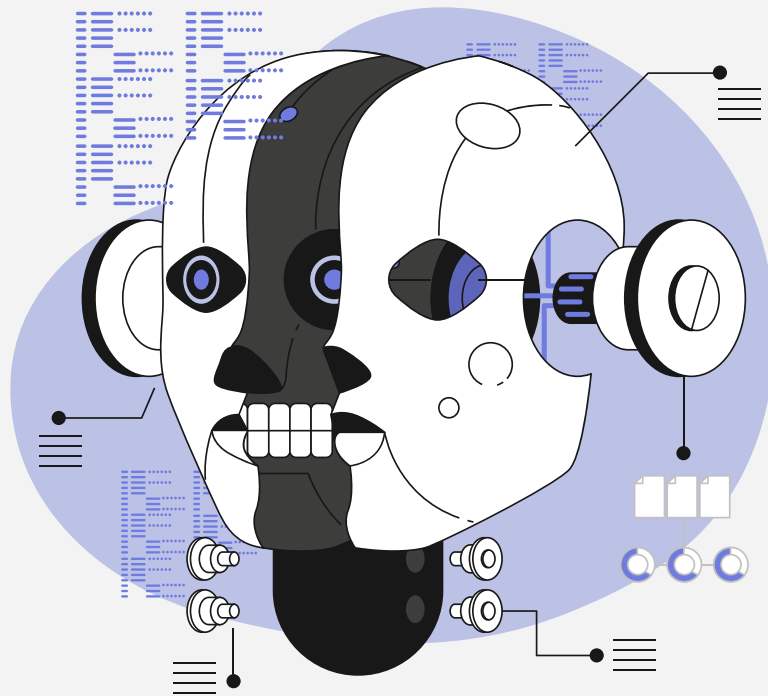


Rys 2. Uniwersalny schemat architektury MLLM. W miejscu Dekodera LLM równie dobrze może wystąpić Transformer o architekturze encoder-decoder, a CNN jako enkoder obrazu.





- Modele językowe i wizyjne
- **Florence-2**
- Phi-3.5-vision
- Fine-tuning
- Wnioski końcowe



Florence-2

Florence-2 to lekki VLM dostępny na licencji MIT autorstwa Microsoft Azure AI. Wykazuje wysoką skuteczność wykrywania obiektów na zdjęciach, generowania dla nich opisu oraz segmentacji. Florence-2 uczony był na 126 mln obrazów i 5.4 mld kompleksowych adnotacjach wizualnych (zbiór danych FLD-5B).

Florence-2 przyjmuje na wejściu odpowiednio przetworzone zdjęcie - koderem obrazu **Diverse Vision Transformer (DaViT)** - oraz instrukcję (zadanie) w postaci tekstu (statycznego *promptu*). Wyniki które generuje mogą uwzględniać wykryte obiekty bezpośrednio na zdjęciach (bounding boxy), zachować formę etykiet lub wygenerowanego opisu.



Florence-2 to lekki VLM dostępny na licencji MIT autorstwa Microsoft Azure AI. Wykazuje wysoką skuteczność wykrywania obiektów na zdjęciach, generowania dla nich opisu oraz segmentacji. Florence-2 uczony był na 126 mln obrazów i 5.4 mld kompleksowych adnotacjach wizualnych (zbiór danych FLD-5B).

Florence-2 przyjmuje na wejściu odpowiednio przetworzone zdjęcie - koderem obrazu **Diverse Vision Transformer (DaViT)** - oraz instrukcję (zadanie) w postaci tekstu (statycznego *promptu*). Wyniki które generuje mogą uwzględniać wykryte obiekty bezpośrednio na zdjęciach (bounding boxy), zachować formę etykiet lub wygenerowanego opisu.



Model wykazuje solidne właściwości rozpoznawania nieznanych wcześniej klas (**zero-shot**) oraz dostrajania (**fine-tuningu**).



Florence-2

Florence-2 dostępny jest w dwóch wariantach:

Model	Parametry (B)	Zużycie vRAM	Rozmiar pliku
Florence-2-base	0.23	1 368 MB	464.4 MB
Florence-2-large	0.77	11 218 MB	1.5 GB

Tab 3. Bazowe modele Florence-2 emitują natywnie typ float16.

Florence-2 można pobrać z repozytorium Hugging Face z użyciem pakietu `transformers`.



Florence-2 - pobieranie i inicjalizowanie modelu

```
1.  from transformers import AutoProcessor, AutoModelForCausalLM
2.  import torch
3.
4.  import supervision as sv
5.  from PIL import Image
6.
7.  CHECKPOINT = "microsoft/Florence-2-base" # lub Florence-2-large
8.
9.  DEVICE = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
10.
11.  model = AutoModelForCausalLM.from_pretrained(
12.      CHECKPOINT,
13.      trust_remote_code=True
14.  ).to(DEVICE)
15.
16.  processor = AutoProcessor.from_pretrained(
17.      CHECKPOINT,
18.      trust_remote_code=True
19.  )
```



Florence-2 - detekcja obiektu (<OD>)

```
21. example = "european_lynx.jpg"
22.
23. image = Image.open(example)
24.
25. task = "<OD>"
26. text = "<OD>"
27.
28. inputs = processor(
29.     text=text,
30.     images=image,
31.     return_tensors="pt"
32. ).to(DEVICE)
```

```
33. generated_ids = model.generate(
34.     input_ids=inputs["input_ids"],
35.     pixel_values=inputs["pixel_values"],
36.     max_new_tokens=1024,
37.     num_beams=3
38. )
39.
40. generated_text = processor.batch_decode(
41.     generated_ids,
42.     skip_special_tokens=False
43. )[0]
44.
45.
46. response =
47.     processor.post_process_generation(
48.         generated_text,
49.         task=task,
50.         image_size=(image.width, image.height)
51.     )
```



Florence-2 - detekcja obiektu (<OD>)



```
{'<OD>': {'bboxes': [[  
    792.1199951171875,  
    885.85595703125,  
    1763.27197265625,  
    1360.031982421875]],  
    'labels': ['lynx']}}}
```

Rys 3a. Przykład działania detekcji obiektowej Florence-2-base. Model zwraca bbox wraz z etykietą wykrytego obiektu.



Florence-2 - generowanie opisu (<MORE_DETAILED_CAPTION>)

```
21. example = "european_lynx.jpg"
22.
23. image = Image.open(example)
24.
25. task =
    "<MORE_DETAILED_CAPTION >"
26. text =
    "<MORE_DETAILED_CAPTION >"
27.
28. inputs = processor(
29.     text=text,
30.     images=image,
31.     return_tensors="pt"
32. ).to(DEVICE)

33. generated_ids = model.generate(
34.     input_ids=inputs["input_ids"],
35.     pixel_values=inputs["pixel_values"],
36.     max_new_tokens=1024,
37.     num_beams=3
38. )
39.
40. generated_text = processor.batch_decode(
41.     generated_ids,
42.     skip_special_tokens=False
43. )[0]
44.
45.
46. response =
    processor.post_process_generation(
47.     generated_text,
48.     task=task,
49.     image_size=(image.width, image.height)
50. )
```



Florence-2 - generowanie opisu (<MORE_DETAILED_CAPTION>)



```
{'<MORE_DETAILED_CAPTION>': 'The image shows a lynx walking through a forest. The lynx is walking on a patch of fallen leaves and rocks, with trees in the background. The ground is covered in fallen leaves, and there are fallen branches and twigs scattered around. The sky is overcast, and the overall mood of the image is peaceful and serene.'}
```

Rys 3b. Przykład działania detekcji obiektowej Florence-2-base. Model zwraca opis zdjęcia do wykrytych obiektów.



Florence-2 - segmentacja (<REFERRING_EXPRESSION_SEGMENTATION>)

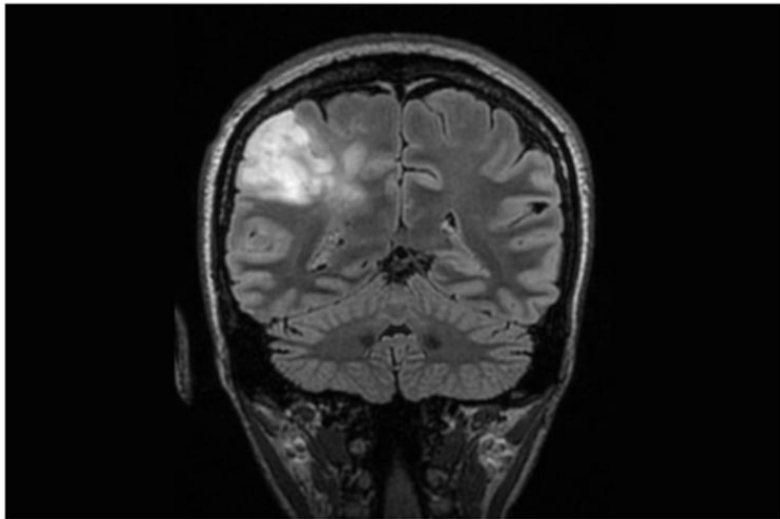
```
21. example = "oligodendroglioma-card.jpg"
22.
23. image = Image.open(example)
24.
25. task = "<REFERRING_EXPRESSION_SEGMENTATION >"
26. text = "<REFERRING_EXPRESSION_SEGMENTATION > the
    brain"
27.
28. inputs = processor(
29.     text=text,
30.     images=image,
31.     return_tensors="pt"
32. ).to(DEVICE)

33.
34. (...)
35.
```

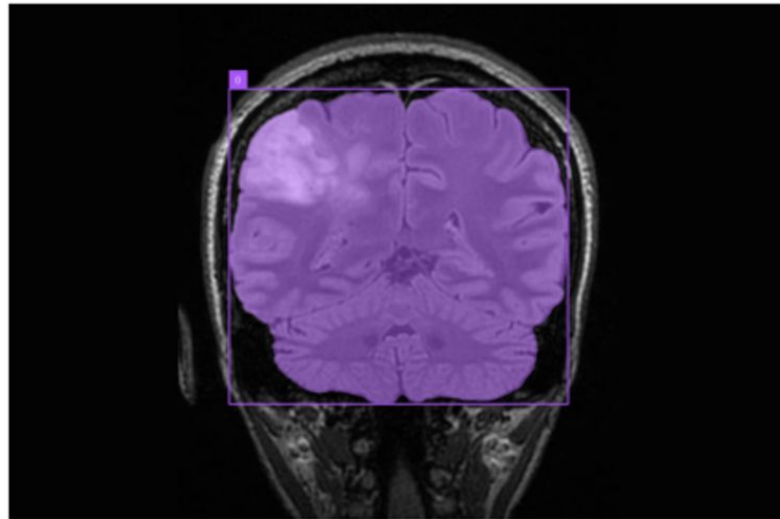


Florence-2 - segmentacja (<REFERRING_EXPRESSION_SEGMENTATION>)

Obraz wejściowy



Segmentacja wykrytego obiektu

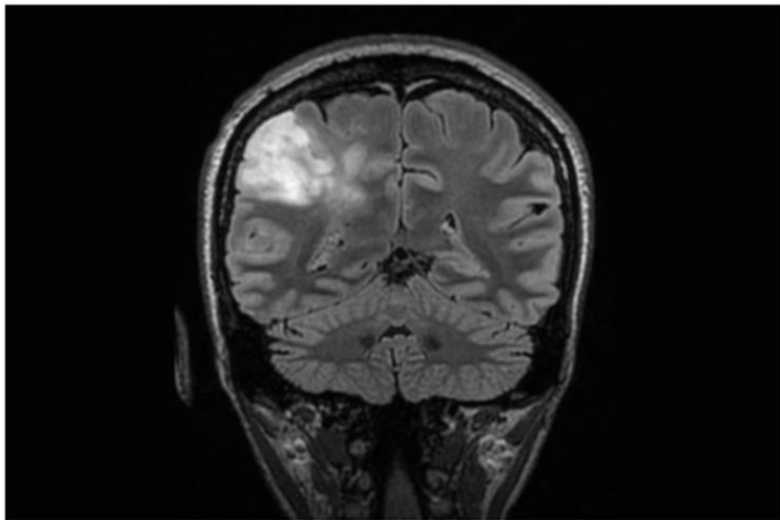


Rys 4. Przykład działania segmentacji Florence-2-large.
Model zwraca segment na podstawie słowa kluczowego.



Florence-2 - segmentacja (<REFERRING_EXPRESSION_SEGMENTATION>)

Obraz wejściowy



Rys 4. Przykład działania segmentacji Florence-2-large.
Model zwraca segment na podstawie słowa kluczowego.

```
{'<MORE_DETAILED_CAPTION>': 'The image is a black and white MRI scan of the brain. The scan is a cross-sectional view of the head and neck, showing the various areas of the human brain, including the cerebrum, cerebellum, and brainstem. The image is taken from a top-down perspective, with the brain in the center of the image and the surrounding area on the left side. The brain appears to be slightly tilted, as if it is tilted slightly to the right. The cerebellums are visible in the top left corner of the scan, and the brainstem is located in the bottom right corner.'
```

```
The image also shows the location of the cerebrospinal cord, which is visible on the top right corner, which extends from the top of the skull to the bottom left corner. This is likely the location where the brain is located. The spinal cord is also visible, as well as the spinal cord and the spinal nerves. The nerves are also visible on both sides of the neck and the head. Overall, the image appears to have been taken at night, as the background is dark and out of focus.'}
```

Florence-2 - Optical Character Recognition (<OCR>)

```
21. example = "original.jpg"
22.
23. image = Image.open(example)
24.
25. task = "<OCR>"
26. text = "<OCR>"
27.
28. inputs = processor(
29.     text=text,
30.     images=image,
31.     return_tensors="pt"
32. ).to(DEVICE)

33. generated_ids = model.generate(
34.     input_ids=inputs["input_ids"],
35.     pixel_values=inputs["pixel_values"],
36.     max_new_tokens=1024,
37.     num_beams=3
38. )
39.
40. generated_text = processor.batch_decode(
41.     generated_ids,
42.     skip_special_tokens=False
43. )[0]
44.
45.
46. response =
47.     processor.post_process_generation(
48.         generated_text,
49.         task=task,
50.         image_size=(image.width, image.height)
51.     )
```



Florence-2 - Optical Character Recognition (<OCR>)



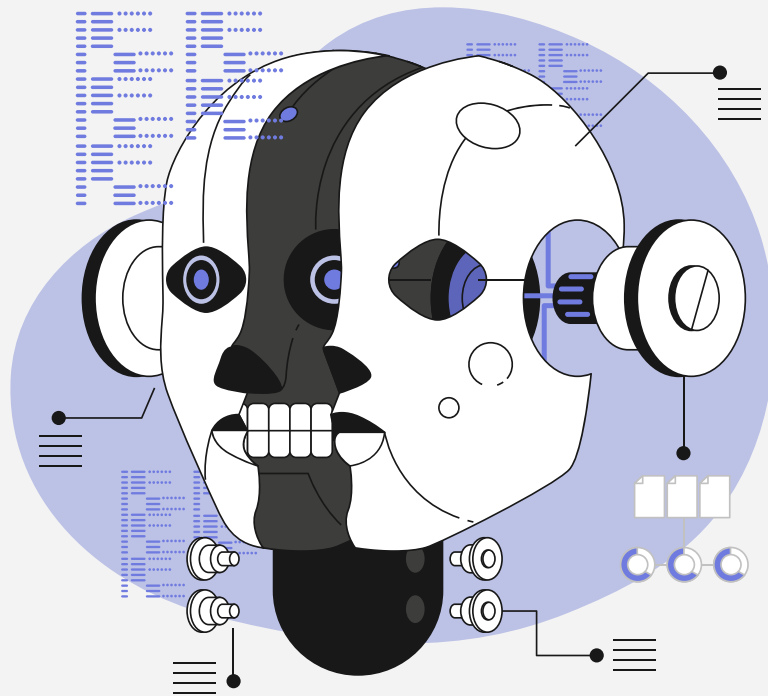
```
{ '<OCR>':  
  'MORELOSPYG-117-B' }
```

Rys 5. Przykład działania OCR
Florence-2-large.





- Modele językowe i wizyjne
- Florence-2
- **Phi-3.5-vision**
- Fine-tuning
- Wnioski końcowe



Phi-3.5-vision

Phi-3.5-vision dostępny jest na licencji MIT autorstwa Microsoft. Model ten wykazuje wysoką skuteczność generowania bardziej szczegółowego opisu (o szerszym kontekście) dla wykrytych obiektów. Phi-3.5-vision uczony był na 500B tokenów. Przyjmuje na wejściu odpowiednio przetworzone zdjęcie lub kolekcję zdjęć a nawet film (w formie klatek) oraz instrukcję w postaci tekstu. Wyniki które generuje mają format wygenerowanego opisu.

Phi-3.5-vision pozwala na pracę w **trybie konwersacji (chatu)**, czyli na pogłębianie kolejnymi instrukcjami interpretacji zdjęć. Przykładowo w początkowej fazie promptem otrzymujemy detekcję obiektową, a w kolejnej interpretację aktywności czy zachowania obiektu (np. człowieka).



Phi-3.5-vision

Phi-3.5-vision dostępny jest na licencji MIT autorstwa Microsoft. Model ten wykazuje wysoką skuteczność generowania bardziej szczegółowego opisu (o szerszym kontekście) dla wykrytych obiektów. Phi-3.5-vision uczony był na 500B tokenów. Przyjmuje na wejściu odpowiednio przetworzone zdjęcie lub kolekcję zdjęć a nawet film (w formie klatek) oraz instrukcję w postaci tekstu. Wyniki które generuje mają format wygenerowanego opisu.

Phi-3.5-vision pozwala na pracę w **trybie konwersacji (chatu)**, czyli na pogłębianie kolejnymi instrukcjami interpretacji zdjęć. Przykładowo w początkowej fazie promptem otrzymujemy detekcję obiektową, a w kolejnej interpretację aktywności czy zachowania obiektu (np. człowieka).

Model wykazuje dużą skuteczność w interpretacji wykresów i tabel. Można go użyć do porównywania obrazów między sobą ale także do podsumowywania filmów.



Phi-3.5-vision

Phi-3.5 dostępny jest w trzech wariantach:

Model	Parametry (B)	Zużycie vRAM	Rozmiar pliku
Phi-3.5-mini	3.8	7 550 MB	7.6 GB
Phi-3.5-MoE	6.6	84 GB	83.9 GB
*Phi-3.5-vision	4.2	8 172 MB	8.3 GB

Tab 3. Bazowe modele Phi-3.5.

Phi-3.5 można pobrać z repozytorium Hugging Face z użyciem pakietu `transformers`.



Phi-3.5-vision - pobieranie i inicjalizowanie modelu

```
1.  from transformers import AutoProcessor, AutoModelForCausalLM
2.  import torch
3.
4.  from PIL import Image
5.
6.  CHECKPOINT = "microsoft/Phi-3.5-vision-instruct"
7.
8.  DEVICE = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
9.
10. model = AutoModelForCausalLM.from_pretrained(
11.     CHECKPOINT,
12.     device_map="cuda",
13.     trust_remote_code=True,
14.     torch_dtype="auto",
15.     _attn_implementation="flash_attention_2" # A100, H100, A6000 wymaga paczki flash_attn
16. ).to(DEVICE)
17.
18. processor = AutoProcessor.from_pretrained(
19.     CHECKPOINT,
20.     trust_remote_code=True,
21.     num_crops=16 # wartość 16 w przypadku jednego zdjęcia, a 4 do multi-frames
22. )
```



Phi-3.5-vision - analiza wykresu

```
23. example = "sml-chart-example.jpg"
24.
25. images = [
26.     Image.open(example)
27. ]
28.
29. placeholder = "<|image_1|>"
30.
31. USER_PROMPT = "Which conclusion can be extracted from the
    following chart?"
32.
33. # Prompt w formacie chatu
34. messages = [
35.     {
36.         "role": "user",
37.         "content": f"{placeholder}\n{USER_PROMPT}"
38.     }
39. ]
40.
41. prompt = processor.tokenizer.apply_chat_template(
42.     messages,
43.     tokenize=False,
44.     add_generation_prompt=True
45. )
46.
47. inputs = processor(
48.     prompt,
49.     images,
50.     return_tensors="pt"
51. ).to(DEVICE)
```

```
51. generation_args = {
52.     "max_new_tokens": 1000,
53.     "temperature": 0.3,
54.     "do_sample": True,
55. }
56.
57. generate_ids = model.generate(
58.     **inputs,
59.     eos_token_id=processor.tokenizer.eos_token_id,
60.     **generation_args
61. )
62.
63. # Usuwamy tokeny wejściowe
64. generate_ids = generate_ids[:,
    inputs["input_ids"].shape[1]:]
65.
66. response = processor.batch_decode(
67.     generate_ids,
68.     skip_special_tokens=True,
69.     clean_up_tokenization_spaces=False
70. )[0]
```



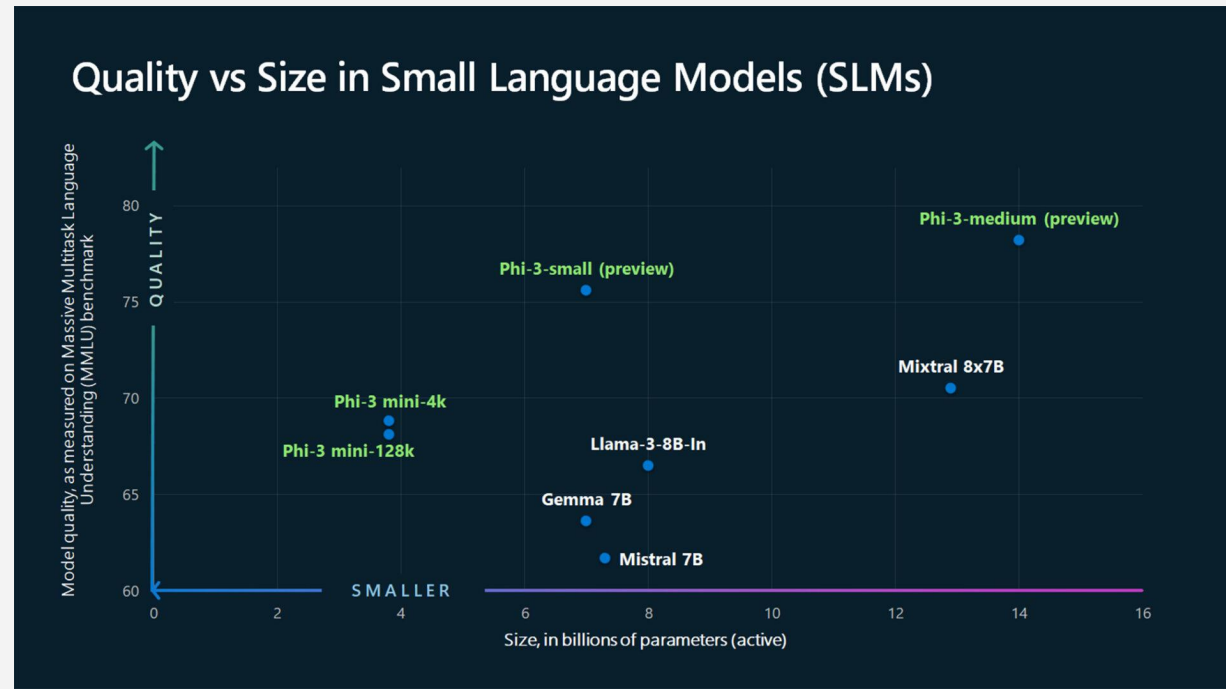
Phi-3.5-vision - analiza wykresu

```
23. example = "sml-chart-example.jpg"
24.
25. images = [
26.     Image.open(example)
27. ]
28.
29. placeholder =
30.
31. USER_PROMPT =
    following chat
32.
33. # Prompt w formacie
34. messages = [
35.     {
36.         "role": "user",
37.         "content": f"{placeholder}\n{USER_PROMPT}"
38.     }
39. ]
40.
41. prompt = processor.tokenizer.apply_chat_template(
42.     messages,
43.     tokenize=False,
44.     add_generation_prompt=True
45. )
46.
47. inputs = processor(
48.     prompt,
49.     images,
50.     return_tensors="pt"
51. ).to(DEVICE)
```

Zrozumienie, jak sterować parametrami takimi jak temperatura, top_p, maksymalna długość i sekwencje zatrzymujące, może przekształcić doświadczenia z LLM w coś zupełnie nowego, pozwalając uzyskiwać dużo lepsze odpowiedzi od modeli.

"Inżynieria promptu to klucz do drzwi percepcji LLMów"

```
51. generation_args = {
52.     "max_new_tokens": 1000,
53.     "temperature": 0.3,
54.     "do_sample": True,
55. }
56.
57. generate_ids = model.generate(
58.     **inputs,
59.     eos_token_id=processor.tokenizer.eos_token_id,
60.     **generation_args
61. )
62.
63. # Usuwamy tokeny wejściowe
64. generate_ids = generate_ids[:,
65.     inputs["input_ids"].shape[1]:]
66.
67. response = processor.batch_decode(
68.     generate_ids,
69.     skip_special_tokens=True,
70.     clean_up_tokenization_spaces=False
71. )[0]
```

Rys 6. Przykład działania analizy wykresu przez Phi-3.5-vision. Model zwraca opis zgody ze strukturą zadaną przez prompt.

The chart is a **scatter plot** titled 'Quality vs Size in Small Language Models (SLMs)'. It shows the relationship between the size of various small language models (in billions of parameters) and their quality, which is measured on a multitask benchmark. **The x-axis represents the size of the models**, ranging from 0 to 16 billion parameters, and **the y-axis represents the quality**, ranging from 60 to 80. The models plotted are Phi-3-small, Phi-3-medium, Mixtral 8x7B, Llama-3-8B-In, Gemma 7B, Mistral 7B, Phi-3 mini-4k, and Phi-3 mini-128k. The chart suggests that there is a **general trend where smaller models tend to have lower quality**, but there are exceptions. For instance, the Mixtral 8x7B model, which is relatively small, has a high quality score, indicating that **size is not the only factor determining quality**.

Phi-3.5-vision - interakcja z modelem

```
1. def generate_response(messages: List[dict], images: List[Image]) -> str:
2.     prompt = processor.tokenizer.apply_chat_template(
3.         messages,
4.         tokenize=False,
5.         add_generation_prompt=True
6.     )
7.
8.     inputs = processor(
9.         prompt,
10.        images,
11.        return_tensors="pt"
12.    ).to(DEVICE)
13.
14.    generation_args = {
15.        "max_new_tokens": 300,
16.        "temperature": 0.7,
17.        "do_sample": True
18.    }
19.
20.    generate_ids = model.generate(
21.        **inputs,
22.        eos_token_id=processor.tokenizer.eos_token_id,
23.        **generation_args
24.    )
25.
26.    generate_ids = generate_ids[:, inputs["input_ids"].shape[1]:]
27.
28.    response = processor.batch_decode(
29.        generate_ids,
30.        skip_special_tokens=True,
31.        clean_up_tokenization_spaces=False
32.    ) [0]
33.
34.    return response
```



Phi-3.5-vision - interakcja z modelem

```
(...)  
  
35.     image1 = Image.open("polityk1.png")  
36.     image2 = Image.open("polityk2.png")  
37.  
38.     # Historia chatu  
39.     conversation = []  
40.  
41.     conversation.append({  
42.         "role": "user",  
43.         "content": "<|image_1|>\nDescribe this image in detail."  
44.     })  
45.  
46.     response1 = generate_response([conversation[-1]], [image1])
```



Phi-3.5-vision - interakcja z modelem

```
(...)  
  
35.     image1 = Image.open("polityk1.png")  
36.     image2 = Image.open("polityk2.png")  
37.  
38.     # Historia chatu  
39.     conversation = []  
40.  
41.     conversation.append({  
42.         "role": "user",  
43.         "content": "<|image_1|>\nDescribe this image in detail."  
44.     })  
45.  
46.     response1 = generate_response([conversation[-1]], [image1])
```

The image features a middle-aged man with short brown hair, wearing a dark suit and blue tie, speaking into a microphone. He is standing in front of a backdrop that includes the European Union flag and a logo that resembles the United Kingdom's Union Jack. The setting suggests a formal event, possibly a conference or press briefing.



Phi-3.5-vision - interakcja z modelem

```
(...)  
  
35. image1 = Image.open("polityk1.png")  
36. image2 = Image.open("polityk2.png")  
37.  
38. # Historia chatu  
39. conversation = []  
40.  
41. conversation.append({  
42.     "role": "user",  
43.     "content": "<|image_1|>\nDescribe this image in detail."  
44. })  
45.  
46. response1 = generate_response([conversation[-1]], [image1])  
47.  
48. conversation.append({"role": "assistant", "content": response1})  
49.  
50. conversation.append({  
51.     "role": "user",  
52.     "content": "Could this person be a politician?"  
53. })  
54.  
55. response2 = generate_response(conversation, [image1])
```

The image features a middle-aged man with short brown hair, wearing a dark suit and blue tie, speaking into a microphone. He is standing in front of a backdrop that includes the European Union flag and a logo that resembles the United Kingdom's Union Jack. The setting suggests a formal event, possibly a conference or press briefing.



Phi-3.5-vision - interakcja z modelem

```
(...)  
  
35. image1 = Image.open("polityk1.png")  
36. image2 = Image.open("polityk2.png")  
37.  
38. # Historia chatu  
39. conversation = []  
40.  
41. conversation.append({  
42.     "role": "user",  
43.     "content": "<|image_1|>\nDescribe this image in detail."  
44. })  
45.  
46. response1 = generate_response([conversation[-1]], [image1])  
47.  
48. conversation.append({"role": "assistant", "content": response1})  
49.  
50. conversation.append({  
51.     "role": "user",  
52.     "content": "Could this person be a politician?"  
53. })  
54.  
55. response2 = generate_response(conversation, [image1])
```

The image features a middle-aged man with short brown hair, wearing a dark suit and blue tie, speaking into a microphone. He is standing in front of a backdrop that includes the European Union flag and a logo that resembles the United Kingdom's Union Jack. The setting suggests a formal event, possibly a conference or press briefing.

Yes, the attire, the presence of the European Union flag, and the formal setting suggest that this person could be a politician.



Phi-3.5-vision - interakcja z modelem

```
(...)

35. image1 = Image.open("polityk1.png")
36. image2 = Image.open("polityk2.png")
37.
38. # Historia chatu
39. conversation = []
40.
41. conversation.append({
42.     "role": "user",
43.     "content": "<|image_1|>\nDescribe this image in detail."
44. })
45.
46. response1 = generate_response([conversation[-1]], [image1])
47.
48. conversation.append({"role": "assistant", "content": response1})
49.
50. conversation.append({
51.     "role": "user",
52.     "content": "Could this person be a politician?"
53. })
54.
55. response2 = generate_response(conversation, [image1])
56.
57. conversation.append({"role": "assistant", "content": response2})
58.
59. conversation.append({
60.     "role": "user",
61.     "content": "<|image_2|>\nNow, describe this new image and
62.               compare it to the previous one."
63. })
64.
65. response3 = generate_response(conversation, [image1, image2])
66.
```

The image features a middle-aged man with short brown hair, wearing a dark suit and blue tie, speaking into a microphone. He is standing in front of a backdrop that includes the European Union flag and a logo that resembles the United Kingdom's Union Jack. The setting suggests a formal event, possibly a conference or press briefing.

Yes, the attire, the presence of the European Union flag, and the formal setting suggest that this person could be a politician.



Phi-3.5-vision - interakcja z modelem

```
(...)

35. image1 = Image.open("polityk1.png")
36. image2 = Image.open("polityk2.png")
37.
38. # Historia chatu
39. conversation = []
40.
41. conversation.append({
42.     "role": "user",
43.     "content": "<|image_1|>\nDescribe this image in detail."
44. })
45.
46. response1 = generate_response([conversation[-1]], [image1])
47.
48. conversation.append({"role": "assistant", "content": response1})
49.
50. conversation.append({
51.     "role": "user",
52.     "content": "Could this person be a politician?"
53. })
54.
55. response2 = generate_response(conversation, [image1])
56.
57. conversation.append({"role": "assistant", "content": response2})
58.
59. conversation.append({
60.     "role": "user",
61.     "content": "<|image_2|>\nNow, describe this new image and
62.               compare it to the previous one."
63. })
64.
65. response3 = generate_response(conversation, [image1, image2])
66.
```

The image features a middle-aged man with short brown hair, wearing a dark suit and blue tie, speaking into a microphone. He is standing in front of a backdrop that includes the European Union flag and a logo that resembles the United Kingdom's Union Jack. The setting suggests a formal event, possibly a conference or press briefing.

Yes, the attire, the presence of the European Union flag, and the formal setting suggest that this person could be a politician.

The second image shows a man who appears to be the same person as in the first image, but the resolution is lower, and the man's expression and posture are different. He is gesturing with one hand and seems to be making a point or argument. The backdrop is not clearly visible, but it seems to be a formal event similar to the first image. The comparison suggests that the first image is of higher quality and provides a clearer view of the person's facial features and attire, while the second image shows him in a different moment of speaking.

Phi-3.5-vision - interakcja z modelem

```
(...)  
  
41. conversation.append({  
42.     "role": "user",  
43.     "content": "<|image_1|>\nDescribe this image in detail."  
44. })  
45.  
46. response1 = generate_response([conversation[-1], [image1]])  
47.  
48. conversation.append({"role": "assistant", "content": response1})  
49.  
50. conversation.append({  
51.     "role": "user",  
52.     "content": "Could this person be a politician?"  
53. })  
54.  
55. response2 = generate_response(conversation, [image1])  
56.  
57. conversation.append({"role": "assistant", "content": response2})  
58.  
59. conversation.append({  
60.     "role": "user",  
61.     "content": "<|image_2|>\nNow, describe this new image and  
62.         compare it to the previous one."  
63. })  
64.  
65. response3 = generate_response(conversation, [image1, image2])  
66.  
67. conversation.append({"role": "assistant", "content": response3})  
68.  
69. conversation.append({  
70.     "role": "user",  
71.     "content": "Hypothetically, if the people from both pictures were running for president, which one should be chosen and why?"  
72. })
```

(...)

(...)

(...)

Phi-3.5-vision - interakcja z modelem

(...)

```
41. conversation.append({
42.     "role": "user",
43.     "content": "<|image_1|>\nDescribe this image in detail."
44. })
45.
46. response1 = generate_response([conversation[-1]], [image1])
47.
48. conversation.append({"role": "assistant", "content": response1})
49.
50. conversation.append({
51.     "role": "user",
52.     "content": "Could this person be a politician?"
53. })
54.
55. response2 = generate_response(conversation, [image1])
56.
57. conversation.append({"role": "assistant", "content": response2})
58.
59. conversation.append({
60.     "role": "user",
61.     "content": "<|image_2|>\nNow, describe this new image and
62.               compare it to the previous one."
63. })
64.
65. response3 = generate_response(conversation, [image1, image2])
66.
67. conversation.append({"role": "assistant", "content": response3})
68.
69. conversation.append({
70.     "role": "user",
71.     "content": "Hypothetically, if the people from both pictures were running for president, which one should be chosen and why?"
72. })
```

(...)

(...)

(...)

The man in the first image appears more composed and professional, with a relaxed posture and a confident demeanor. His suit is neatly fitted, and he seems to be in control of the situation, which could be interpreted as a sign of experience and competence. The second man, while also professional, has a more animated expression and gesture, which could be seen as either passionate or lacking control. (...)

Phi-3.5-vision - interakcja z modelem

```
(...)  
  
41. conversation.append({  
42.     "role": "user",  
43.     "content": "<|image_1|>\nDescribe this image in detail."  
44. })  
45.  
46. response1 = generate_response([conversation[-1], [image1]])  
47.  
48. conversation.append({"role": "assistant", "content": response1})  
49.  
50. conversation.append({  
51.     "role": "user",  
52.     "content": "Could this person be a politician?"  
53. })  
54.  
55. response2 = generate_response(conversation, [image1])  
56.  
57. conversation.append({"role": "assistant", "content": response2})  
58.  
59. conversation.append({  
60.     "role": "user",  
61.     "content": "<|image_2|>\nNow, describe this new image and  
62.         compare it to the previous one."  
63. })  
64.  
65. response3 = generate_response(conversation, [image1, image2])  
66.  
67. conversation.append({"role": "assistant", "content": response3})  
68.  
69. conversation.append({  
70.     "role": "user",  
71.     "content": "Hypothetically, if the people from both pictures were running for president, which one should be chosen and why?"  
72. })
```

(...)

(...)

(...)

The man in the first image appears more composed and professional, with a relaxed posture and a confident demeanor. His suit is neatly fitted, and he seems to be in control of the situation, which could be interpreted as a sign of experience and competence. The second man, while also professional, has a more animated expression and gesture, which could be seen as either passionate or lacking control. **Without additional context, it's challenging to determine which would be the better choice for a president. Both could be successful, depending on the political climate, the voters' preferences, and the specific issues being addressed.**

Phi-3.5-vision - interakcja z modelem

(...)

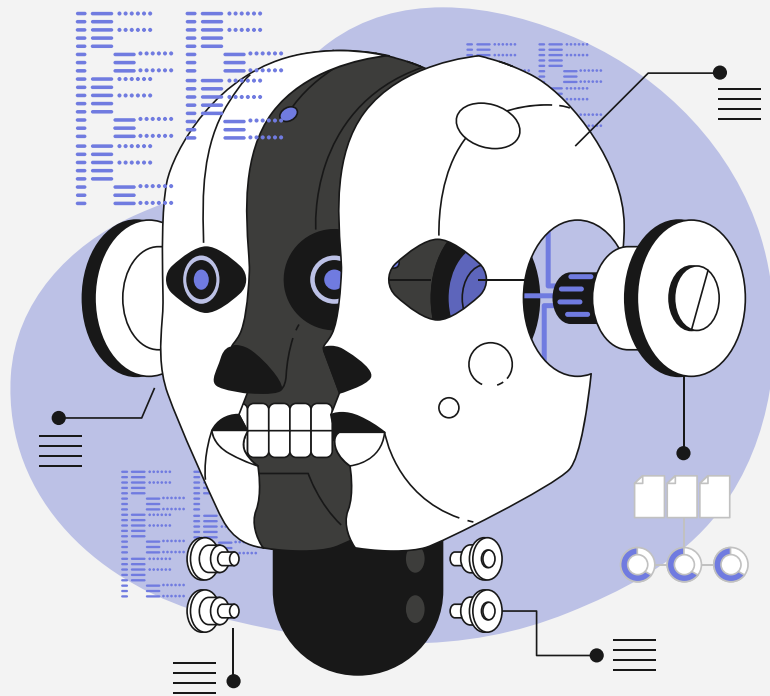
```
41. conversation.append({
42.     "role": "user",
43.     "content": "<|im
44. })
45.
46. response1 = generat
47.
48. conversation.append
49.
50. conversation.append
51.     "role": "user",
52.     "content": "Coul
53. })
54.
55. response2 = generat
56.
57. conversation.append
58.
59. conversation.append
60.     "role": "user",
61.     "content": "<|im
62.                 comp
63. })
64.
65. response3 = generat
66.
67. conversation.append({"role": "assistant", "content": response3})
68.
69. conversation.append({
70.     "role": "user",
71.     "content": "Hypothetically, if the people from both pictures were running for president, which one should be chosen and why?"
72. })
```

(...)

composed and professional, with a
or. His suit is neatly fitted, and he
which could be interpreted as a sign
ond man, while also professional,
sture, which could be seen as either
additional context, it's challenging
er choice for a president. Both
the political climate, the voters'
being addressed.



- Modele językowe i wizyjne
- Florence-2
- Phi-3.5-vision
- **Fine-tuning**
- Wnioski końcowe



Fine-tuning jako droga do adaptacji modelu

Zdolność rozpoznawania podstawowych wzorców przez modele językowe jest często wystarczająca. Natomiast, do opanowania określonych zadań specjalistycznych wymagany jest dodatkowe szkolenie.

Dodatkowy trening pomaga modelowi lepiej przewidywać wyniki w przypadku konkretnego zadania. To dodatkowe szkolenie, nazywane **dostrajaniem**, pozwala osiągnąć praktyczny wymiar LLM na naszych danych.



Fine-tuning jako droga do adaptacji modelu

Zdolność rozpoznawania podstawowych wzorców przez modele językowe jest często wystarczająca. Natomiast, do opanowania określonych zadań specjalistycznych wymagany jest dodatkowe szkolenie.

Dodatkowy trening pomaga modelowi lepiej przewidywać wyniki w przypadku konkretnego zadania. To dodatkowe szkolenie, nazywane **dostrajaniem**, pozwala osiągnąć praktyczny wymiar LLM na naszych danych.

Dostrajanie wymaga mniej mocy obliczeniowej w porównaniu do uczenia modelu od zera.



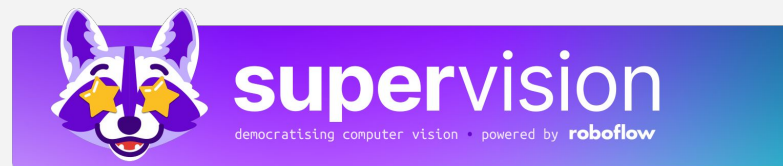
Dostrajanie oparte na instrukcjach (np. do zadania generowania odpowiedzi)



Dostrajanie pojedynczego zadania (np. do zadania podsumowywania dokumentów)



Fine-tuning jako droga do adaptacji modelu



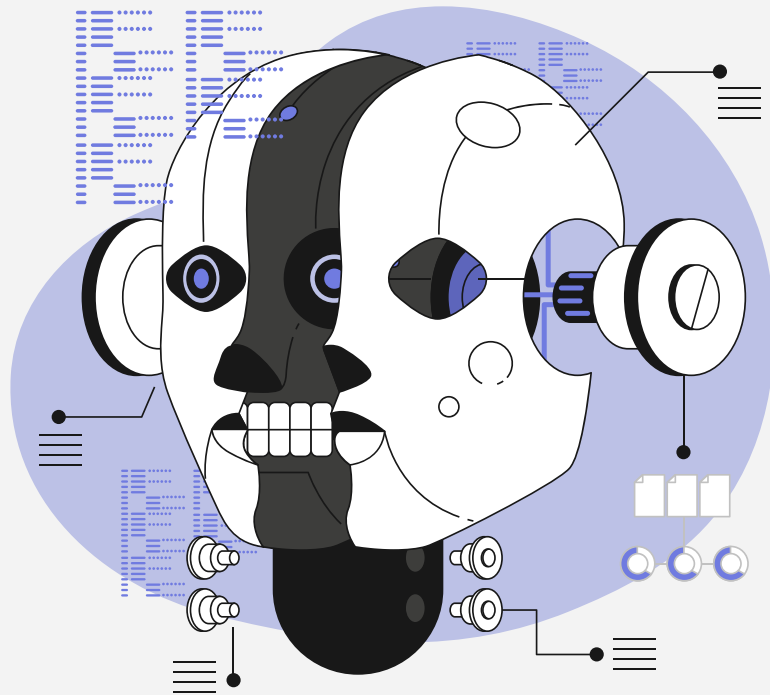
Phi-3.5-vision finetuning recipe

This is the official support of Phi-3.5-vision finetuning using huggingface libraries. Please `cd` to the code directory [vision_fineting](#) before running the following commands.





- Modele językowe i wizyjne
- Florence-2
- Phi-3.5-vision
- Fine-tuning
- **Wnioski końcowe**



Praktyczne zastosowanie modeli językowych

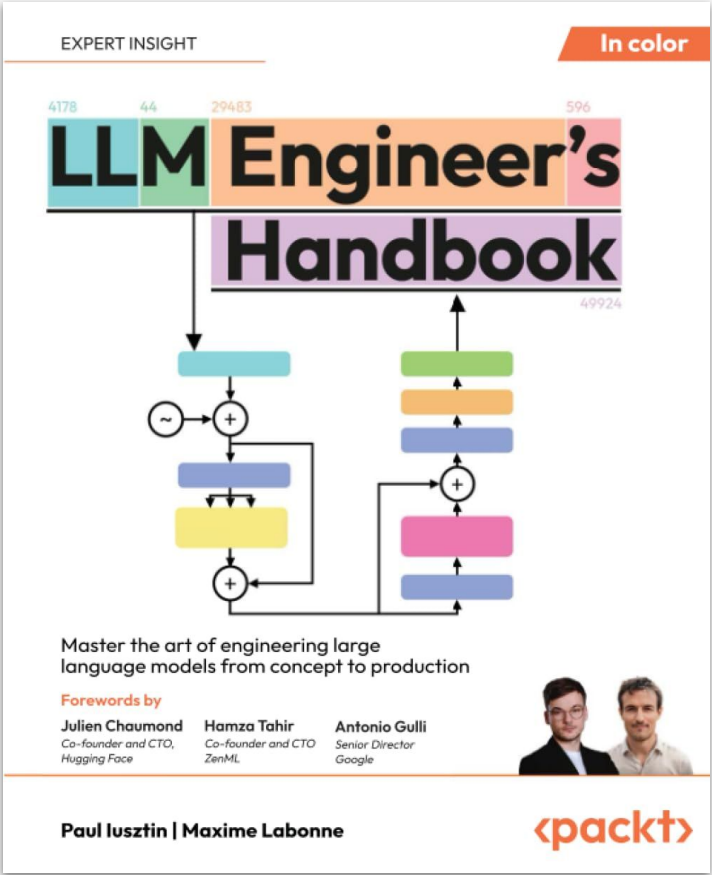
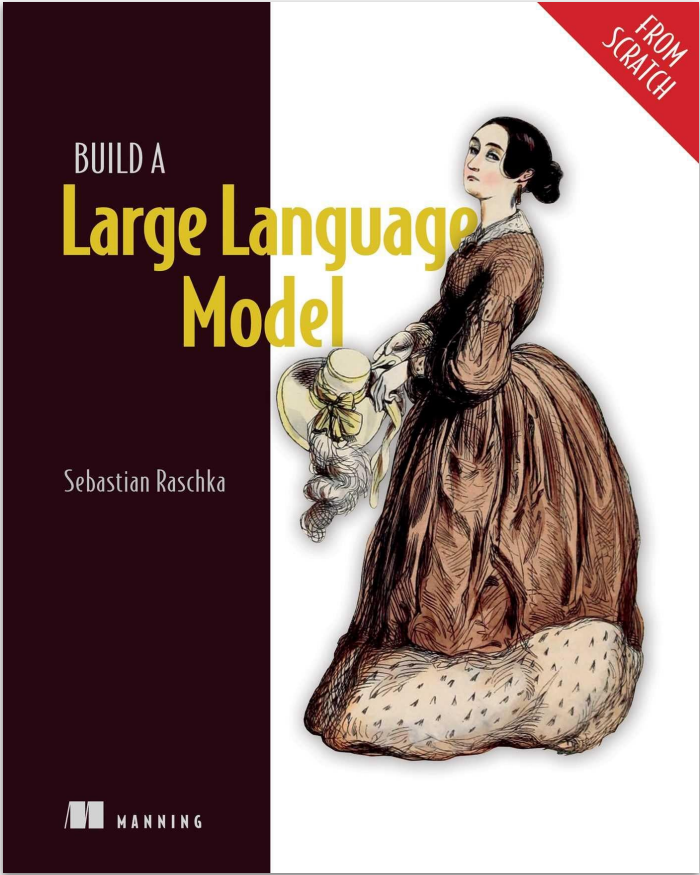
- Chatboty do obsługi klienta, head hunting (automatyczne przetwarzanie CV), analiza sentymentów w e-commerce, automatyzacja marketingu (spersonalizowane treści, komunikaty), prasówka;
- "Sanity check" dokumentacji prawnej (analiza zapisów, przepisów, wyłączeń na okoliczność wystąpienia oczywistych błędów czy niespójności);
- Przetwarzanie dokumentów analogowych i cyfrowych do standardowego modelu relacyjnego. Tworzenie agenta (asystenta) wyposażonego w wiedzę przedsiębiorstwa (digitalizacja know-how firmy);
- Wyposażanie NPC w nowy rodzaj relacji z użytkownikiem (realistyczne i bardziej responsywne postacie cyfrowe);
- Opieka senioralna (asystent AI dbający o komunikację, kalendarz wizyt, dawkovanie leków);



Wnioski końcowe

- Uruchamianie lokalnie SML/VLM to praktyczna i tania alternatywa dla LLMów chmurowych. Niski próg wejścia, wielozadaniowość, efektywne dostrajanie modeli własnymi danymi.
- Rozwój (M)LLM-ów upowszechnia rozwój **agentów AI**, programów będących w stanie półautonomicznie wykonywać proste cyfrowe zadania używając innych programów czy serwisów;
- Koszt tworzenia prostego oprogramowania (np. CRUD, integrację) zacznie spadać "na łeb na szyję";
- Wszystko na to wskazuje, że niebawem będziemy mogli kupić ekwiwalent ChatGPT do domu, jako urządzenia, które nie będzie musiało korzystać z modeli w chmurze;
- Wydanie **GPT-5** oraz **Grok-3** w grudniu 2024 rzuci nowe światło na krajobraz modeli multimodalnych.





Reference

- MM1: Methods, Analysis & Insights from Multimodal LLM Pre-training (arXiv:2403.09611, v4, 18.04.2024)
- Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks (arXiv:2311.06242, v1, 10.11.2023)
- Florence-2: Open Source Vision Foundation Model by Microsoft
- Florence 2 - The Best Small VLM Out There?
- Master different vision tasks with pre-trained Florence-2 | Community Q&A
- Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone (arXiv:2404.14219, v4, 30.08.2024)
- Phi-3CookBook - Microsoft's Phi-3 family
- GPT-4 Architecture, Infrastructure, Training Dataset, Costs, Vision, MoE Demystifying GPT-4: The engineering tradeoffs that led OpenAI to their architecture (07.2023)
- Ustawienia LLM – jak otrzymywać lepsze odpowiedzi od AI? (12.2023)
- “Transformer - jak działa rewolucyjna architektura?”, hAI Magazine 1/2024
- “Jak budować efektywne modele multimodalne?”, hAI Magazine 1/2024
- “Modele językowe”, hAI Magazine 1/2024
- Zrozumienie dostrajania LLM: dostosowywanie modeli dużych języków do Twoich unikalnych wymagań



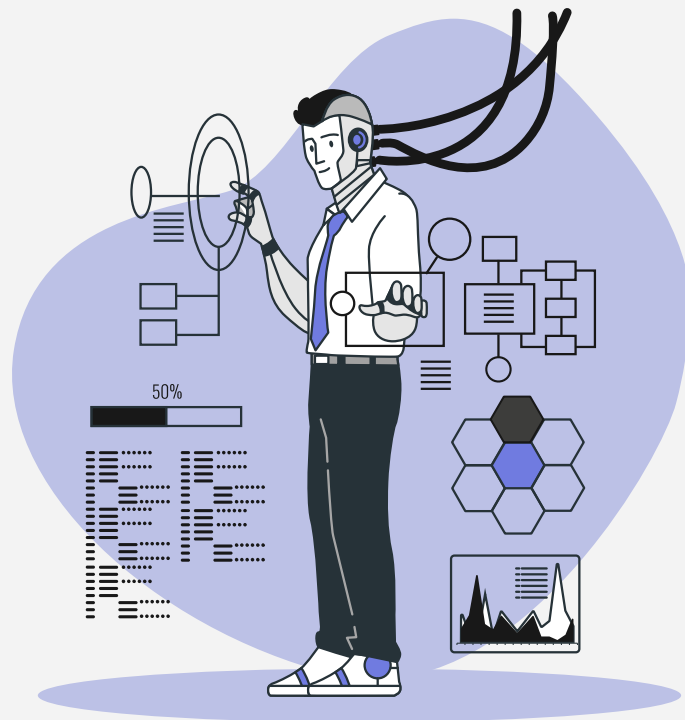
Dziękuję za uwagę.
Zapraszam do dyskusji.



@ptynecki



@piotrtynecki



Phi-3.5-vision - interakcja z modelem

(...)

```
41. conversation.append({
42.     "role": "user",
43.     "content": "<img alt='A man in a dark suit and red tie speaking into a microphone, with a European Union flag in the background.'" data-bbox="238 191 484 766"/>
44. })
45.
46. response1 = generate_response(
47.     conversation.append({"role": "assistant", "content": response1})
48. )
49.
50. conversation.append(
51.     {"role": "user",
52.      "content": "Could you describe the man in the picture?"}
53. )
54.
55. response2 = generate_response(
56.     conversation.append({"role": "assistant", "content": response2})
57. )
58.
59. conversation.append(
60.     {"role": "user",
61.      "content": "<img alt='A man in a dark suit and blue tie speaking into a microphone, with a red and white flag in the background.'" data-bbox="514 191 761 766"/>
62.      "content": "Could you describe the man in the picture?"
63.     })
64.
65. response3 = generate_response(
66.     conversation.append({"role": "assistant", "content": response3})
67. )
68.
69. conversation.append({
70.     "role": "user",
71.     "content": "Hypothetically, if the people from both pictures were running for president, which one should be chosen and why?"
72. })
```



(...)



composed and professional, with a or. His suit is neatly fitted, and he which could be interpreted as a sign and man, while also professional, stance, which could be seen as either **additional context, it's challenging** er choice for a president. Both e political climate, the voters' being addressed.