PRACA Z GPT-3 – WSTĘP

Tu omówiono GPT-3 z OpenAI, co oznacza, że poznasz funkcje GPT-2 i GPT-3 oraz kilka przykładów kodu. Krótkie przykłady kodu oparte na Pythonie wymagają obecnie Pythona 3.7, który można pobrać z Internetu, jeśli jeszcze go nie masz. Pierwsza część opisuje trzy główne typy dobrze znanych modeli LLM (modeli dużego języka): modele tylko kodera, modele tylko dekodera i modele koder-dekoder. Następnie poznasz szczegóły dotyczące rodziny modeli językowych opartych na GPT z OpenAI. Po krótkim opisie GPT-2 otrzymasz ogólne wprowadzenie do GPT-3, wraz z celem GPT-3 i kilkoma modelami opartymi na GPT-3. Druga sekcja zawiera opis możliwości GPT-3, porównanie GPT-3 i BERT oraz mocne i słabe strony GPT-3. Trzecia sekcja zagłębia się w GPT-3 Playground, który został udostępniony wszystkim (od końca 2021 roku). Dowiesz się, jak stworzyć aplikację GPT-3, a także jak przeprowadzać wnioskowanie z wykorzystaniem GPT-3. Ponadto dowiesz się, jak pracować z poleceniami w GPT-3. Przed przeczytaniem materiału w tym rozdziale pamiętaj, że SOTA to akronim od State-Of-The-Art (najnowocześniejszy). Ze względu na tempo rozwoju nowych technologii w świecie przetwarzania języka naturalnego (NLP), model SOTA w dowolnym momencie jest szybko zastępowany innym modelem SOTA, czasami w ciągu kilku tygodni.

RODZINA GPT: WSTĘP

Generalized Pre-trained Transformer, GPT, był pierwszym modelem autoregresyjnym opartym na architekturze transformatora. Za tworzenie modeli opartych na GPT odpowiada OpenAI. Modele GPT, GPT-2 i GPT-3 zostały ogłoszone w 2018, 2019 i 2020 roku i mają odpowiednio 117 M, 1,5 B i 175 B parametrów. Model GPT-4 został wydany w marcu 2023 roku i wydaje się, że ma 1,76 biliona parametrów (10 razy więcej niż GPT-3). Pierwotny model GPT został wstępnie wytrenowany na zbiorze danych BookCorpus. Biblioteka udostępnia wersje modelu do modelowania języka oraz wielozadaniowego modelowania języka/klasyfikacji wielokrotnego wyboru. Modele GPT stają się stopniowo coraz bardziej rozbudowane pod względem architektury i są trenowane na większych zbiorach danych. Na przykład GPT, GPT2 i GPT3 mają maksymalną długość sekwencji wynoszącą 1 kB, 1 kB i 2 kB, a także długość osadzenia słowa odpowiednio 768, 1600 i 12888. Przypomnijmy z rozdziału 5, że BERT ma architekturę wyłącznie kodera, podczas gdy GPT ma architekturę wyłącznie dekodera. Chociaż architektura transformatorowa została opracowana do wykonywania tłumaczenia maszynowego, część modeli GPT związana z transformatorem obejmowała nienadzorowane podejście do wstępnego trenowania. Co ciekawe, główny twórca BERT był pod wpływem architektury GPT, mimo że BERT opiera się na komponencie kodera, a nie na komponencie dekodera architektury transformatorowej.

Architektura rodziny GPT

Mocną stron modeli GPT jest ich zdolność do przetwarzania danych wejściowych w postaci tekstu i generowania danych wyjściowych w postaci tekstu. Modele GPT nie obsługują danych wejściowych w postaci innej niż tekst, takich jak dźwięk czy obraz, ponieważ nie są modelami multimodalnymi. W rozdziale 3 poznałeś modele autoregresyjne. Modele oparte na GPT są również modelami autoregresyjnymi, ponieważ opierają się na komponencie dekodera. Mówiąc w uproszczeniu, za każdym razem, gdy tekst wejściowy jest przesyłany do modelu opartego na GPT, przypisuje on procenty słowom, które potencjalnie mogą pojawić się w tekście odpowiedzi. Konkretne procenty są obliczane przez sam dekoder, który może wykorzystywać różne strategie obliczania tych procentów. Należy pamiętać, że można określić maksymalną liczbę tokenów, które GPT ma wyświetlić w wygenerowanym tekście wyjściowym.

Czym jest GPT-2?

GPT-2 to większa i lepsza wersja oryginalnego modelu GPT. GPT-2 został wstępnie wytrenowany na bazie WebText, która składa się ze stron internetowych z linków wychodzących w serwisie Reddit, z 3 lub więcej „karmami” (tj. głosami na plus). Należy zauważyć, że GPT i GPT-2 zostały wytrenowane na różnych zbiorach danych: pierwszy był trenowany głównie na bazie BookCorpus, a drugi na stronach internetowych, blogach i artykułach informacyjnych linkowanych z Reddita. GPT-2 to oparty na transformatorze algorytm LLM, który oferuje wyłącznie dekoder i może generować tekst wysokiej jakości. GPT-2 obsługuje do 2048 tokenów, co zostało zastąpione przez GPT-3 i GPT-4.

Architektura GPT-2

GPT-2 zapewnia wysoką jakość danych wyjściowych poprzez losowe próbkowanie. Co więcej, GPT-2 osiągnął spójność dalekiego zasięgu w odniesieniu do odległości między tokenami, czego nie mogą osiągnąć sieci neuronowe RNN i sieci LSTM. Z punktu widzenia architektury, kluczowe punkty obejmują następujące szczegóły architektoniczne i związane z uczeniem:

• uwaga przyczynowa (porównaj z BERT)

• autoregresyjny model języka

• pojedynczy stos 48 bloków

• wymiarowość 768 i 12 nagłówków

• długość sekwencji 1024

• osadzanie pozycji

• 1,5 miliarda parametrów

• GELU w warstwie FF

GPT-2 nie został wytrenowany na żadnej zawartości Reddita. Zamiast tego został wytrenowany na zawartości stron internetowych, do których prowadzą linki z Reddita (lub z niego). Ponieważ GPT-2 angażuje uwagę przyczynową, oznacza to, że blok dekodera zawiera zamaskowaną samouwagę, co z kolei oznacza, że każdy token może „widzieć” tylko poprzedzające go tokeny w ciągu wejściowym. To ograniczenie osiągnięto, ustawiając wartości górnego trójkąta macierzy uwagi równe -oo, po czym funkcja aktywacji softmax() ustawia wszystkie wartości w macierzy w lewym górnym rogu na 0. Grupa badaczy przeprowadziła ciekawy eksperyment: wytrenowali model GPT-2 na tekście, a następnie „zamrozili” jego warstwy samouwagi i sprzężenia zwrotnego, a także dostroili model do różnych domen.

GPT-2 i CodeParrot

Listing przedstawia zawartość pliku code_parrot1.py, modelu opartego na GPT-2, który pokazuje, jak utworzyć instancję tokenizera i potoku (oba z biblioteki transformers) w celu generowania tekstu.

```
#pip3 install transformers

#Create an instance of a tokenizer and a pipeline
#for text generation using the CodeParrot model:
from transformers import pipeline

# Create a text generation pipeline with the
CodeParrot model:
text_generator = pipeline('text-generation',
model='codeparrot/codeparrot')

# Use the pipeline to generate text
input_text = "Once upon a time"
generated_text = text_generator(input_text,
max_length=50, do_sample=True, temperature=1.0)

# Print the generated text
print(generated_text[0]['generated_text'])
```

Listing rozpoczyna się od instrukcji import, a następnie inicjalizuje text_generator jako instancję funkcji pipe line(), wraz z text-generation jako zadaniem i codeparrot/codeparrot jako modelem. Następnie inicjalizowana jest zmienna ciągu input_text, po której następuje zmienna generated_text, inicjowana wynikiem zwracanym przez wywołanie text_generator() z parametrem input_text (i innymi zmiennymi). Końcowy fragment kodu wyświetla wygenerowany tekst. UWAGA: Pobrany model jest większy niż 6 GB, a dane wyjściowe wyświetlane poniżej mogą nieznacznie różnić się od tych widocznych po uruchomieniu tego przykładowego kodu. Uruchom kod w Listing za pomocą poniższego polecenia, a zobaczysz, jak model jest automatycznie pobierany (kolejne wywołania tego kodu będą używać pobranego modelu):

python3.7 code_parrot1.py

Wynik poprzedniego polecenia Pythona pokazany jest tutaj wraz z tekstem wyświetlanym podczas pobierania modelu:

```
python3.7 code_parrot1.py
Downloading: 100%|                          | 259/259
[00:00<00:00, 59.7kB/s]
Downloading: 100%|                        | 485k/485k
[00:00<00:00, 1.09MB/s]
Downloading: 100%|                        | 270k/270k
[00:00<00:00, 761kB/s]
Downloading: 100%|                        | 821k/821k
[00:00<00:00, 1.33MB/s]
Downloading: 100%|                        | 90.0/90.0
[00:00<00:00, 59.2kB/s]
/Library/Frameworks/Python.framework/Versions/3.7/
lib/python3.7/site-packages/transformers/models/
auto/modeling_auto.py:787: FutureWarning: The
class `AutoModelWithLMHead` is deprecated and
will be removed in a future version. Please use
`AutoModelForCausalLM` for causal language models,
`AutoModelForMaskedLM` for masked language models
and `AutoModelForSeq2SeqLM` for encoder-decoder
models.
  FutureWarning,
Downloading: 100%|                          | 927/927
[00:00<00:00, 238kB/s]
Downloading:   8%|                       | 495M/5.75G
[00:19<02:22, 39.6MB/s]
"""
```

Tokenizer GPT-2

Listing wyświetla zawartość pliku gpt2_tokenizer1.py, który pokazuje, jak GPT-2 przeprowadza tokenizację ciągu tekstowego.

```
from transformers import GPT2Tokenizer

sentence = "I love Chicago pizza"
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
tokens1 = tokenizer(sentence)


print("-> sentence:",sentence)
print("-> tokens1: ",tokens1)
print()

tokens2 = tokens1['input_ids']
print("-> tokens2: ",tokens2)
print()

#Convert the tokens in into strings using
tokenizer.decode:
tokens3 = [tokenizer.decode(x) for x in tokens2]
print("-> tokens3: ",tokens3)

decoded2 = [tokenizer.decode(x).strip().lower() for
x in tokens2]
print("-> decoded2:",decoded2)

tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
tokens4 = tokenizer('My favorite word is syzygy')
tokens4 = tokens4['input_ids']
print("-> tokens4: ",tokens4)

decoded3 = [tokenizer.decode(x).strip().lower() for
x in tokens4]
print("-> decoded3:",decoded3)

# decoded3: ['my', 'favorite', 'word', 'is', 'sy',
'zy', 'gy']
# syzygy is tokenized as 'sy', 'zy', and 'gy'
```

Listing rozpoczyna się od instrukcji importu, następnie inicjalizuje zmienną sentence jako ciąg tekstowy i inicjalizuje tokenizer jako instancję klasy GPT2Tokenizer. Kolejny fragment kodu wywołuje metodę tokenizer() w celu wygenerowania tokenów numerycznych dla każdego słowa w zmiennej sentence, a także wywołuje metodę tokenizer.decode() w celu „odzyskania” oryginalnych słów w zdaniu. Ponieważ GPT-2 nie rozpoznaje słowa „syzygy”, przeprowadza tokenizację podsłowną, aby wygenerować tokeny „sy”, „zy” i „gy”. W rezultacie generowanych jest 7 tokenów dla 4 słów wejściowych w zmiennej sentence. Uruchom kod z Listingu a zobaczysz następujący wynik:

```
-> sentence: I love Chicago pizza
-> tokens1:  {'input_ids': [40, 1842, 4842, 14256],
'attention_mask': [1, 1, 1, 1]}

-> tokens2:  [40, 1842, 4842, 14256]

-> tokens3:  ['I', ' love', ' Chicago', ' pizza']
-> decoded2: ['i', 'love', 'chicago', 'pizza']
-> tokens4:  [3666, 4004, 1573, 318, 827, 7357,
1360]
-> decoded3: ['my', 'favorite', 'word', 'is', 'sy',
'zy', 'gy']
```

Dla porównania BERT generuje następujący wynik, który zawiera [CLS], [SEP] i prefiks „##” dla słów podrzędnych:

```
['[CLS]', 'my', 'favorite', 'word', 'is', 'sy',
'##zy', '##gy', '[SEP]']
```

Pytania i odpowiedzi z wykorzystaniem GPT-2

Listing wyświetla zawartość pliku gpt2_qna2.py, który pokazuje, jak odpowiadać na pytania za pomocą GPT-2.

```
from transformers import pipeline

qna = pipeline('question-answering')

qc_pair1 = {
    'question': 'Who is the inventor of the Theory
of Relativity?',


    'context': 'Albert Einstein invented the Theory
of Relativity',
}

qc_pair2 = {
  'question': 'Which wine and cheese are from
Italy',
  'context':  'Barolo wine from Italy goes well
with Parmigiano cheese',
}

if __name__ == "__main__":
  result = qna (qc_pair1)
  print("result:")
  print(result)
  print("-----------\n")

  result = qna (qc_pair2)
  print("result:")
  print(result)
  print("-----------\n")
```

Listing rozpoczyna się od instrukcji importu, a następnie inicjalizuje zmienną qna jako instancję klasy potoku, która również określa pytanie-odpowiedź. Następnie zmienne gc_pair1 i gc_pair2 są inicjalizowane jako słowniki Pythona zawierające dwa klucze. Zauważ, że druga para gc_pair2 w rzeczywistości obejmuje dwa pytania zamiast jednego: odpowiedź zawiera tylko odpowiedź na drugą część pytania o ser i ignoruje pytanie o wino. Uruchom kod w Listingu, a zobaczysz następujący wynik:

```
result:
{'score': 0.9971326589584351, 'start': 0, 'end':
15, 'answer': 'Albert Einstein'}
-----------

result:
{'score': 0.2554819583892822, 'start': 38, 'end':
48, 'answer': 'Parmigiano'}
-----------
```

Optimus: Połączenie BERT i GPT-2

BERT ma architekturę wyłącznie enkodera, podczas gdy GPT-2 i Megatron mają architekturę wyłącznie dekodera. Innym interesującym modelem opartym na transformatorach jest Optimus, który łączy BERT i GPT-2 wraz ze zmienną ukrytą z (względem dekodera) za pomocą dwóch technik:

• Dodanie z jako wektora dla dekodera.

• Dodanie z do dolnej warstwy dekodera.

Ta dodatkowa warstwa jest zaangażowana w każdy etap procesu dekodowania.

GPT-2 I GENEROWANIE TEKSTU

Listing wyświetla zawartość pliku gpt2_text_gen1.py, który pokazuje, jak przeprowadzić generowanie tekstu za pomocą modelu GPT-2.

```
from transformers import pipeline

# default uses the GPT-2 model:
gen = pipeline("text-generation")
prompt = "I love Chicago deep dish pizza"
result = gen(prompt)
print("prompt:",prompt)
print("result:",result)
print()
```

Listing rozpoczyna się od instrukcji import, a następnie inicjuje zmienną gen jako instancję klasy potoku, która również określa generowanie tekstu. Następnie zmienna prompt jest inicjowana za pomocą promptu w postaci ciągu tekstowego. Kolejna część Listingu inicjuje zmienną result za pomocą wyniku z fragmentu kodu gen(prompt), a następnie wyświetlana jest zawartość promptu i result. Uruchom kod w Listingu, a zobaczysz następujący wynik:

```
Some weights of GPT2Model were not initialized
from the model checkpoint at gpt2 and are newly
```

```
initialized: ['h.0.attn.masked_bias', 'h.1.attn.
masked_bias', 'h.2.attn.masked_bias', 'h.3.attn.
masked_bias', 'h.4.attn.masked_bias', 'h.5.attn.
masked_bias', 'h.6.attn.masked_bias', 'h.7.attn.
masked_bias', 'h.8.attn.masked_bias', 'h.9.attn.
masked_bias',
'h.10.attn.masked_bias', 'h.11.attn.masked_bias']
You should probably TRAIN this model on a down-
stream task to be able to use it for predictions
and inference.
Setting `pad_token_id` to `eos_token_id`:50256 for
open-end generation.
Setting `pad_token_id` to `eos_token_id`:50256 for
open-end generation.
Setting `pad_token_id` to `eos_token_id`:50256 for
open-end generation.


prompt: I love Chicago deep dish pizza
result: [{'generated_text': 'I love Chicago deep
dish pizza. I love it as well.\n\nI never thought I
would love the idea of a pizza, but I am glad that
I did. I know there is already a dedicated Chicago
pizza section in the neighborhood. I'}]


result2: [{'generated_text': 'I love Chicago deep
dish pizza! When it was starting to come to light
that these guys were making this, they took the
pizza to the local pizzeria and it was amazing. My
boyfriend also enjoyed it and loves to have these
places in town.'}, {'generated_text': 'I love
Chicago deep dish pizza. And the pizza is really
tasty.\n\nA quick thing that you should take note
of this week is the recent return of the Nachos.
The Nachos are a side dish in Chicago and some
of'}, {'generated_text': "I love Chicago deep dish
pizza and feel compelled to cook some of these.
It made me really want to learn more about pizza
cooking in a healthy way, but I don't want to
keep my focus on the best pizza recipe, and don't
want"}]


result3: [{'generated_text': "I love Chicago deep
dish pizza. Chicago deep dish, Chicago deep dish


pizza. So yeah, that's gotta work!\n\nA whole lot
of"}]
```

Na listingu znajduje się zawartość pliku gpt2_text_gen2.py, która pokazuje, jak przeprowadzić generowanie tekstu przy użyciu modelu GPT-2.

```
from transformers import pipeline

text_gen = pipeline("text-generation")

# specify a max_length of 50 tokens and sampling
"off":
prefix_text = "What a wonderful"
generated_text = text_gen(prefix_text, max_
length=50, do_sample=False)[0]

print("-> #1 generated_text['generated_text']:")
print(generated_text['generated_text'])
print("------------------------------\n")

#output #1:
#What a wonderful thing about this is that it's a
very simple and simple way to get your hands on a
new game.
#The game is a simple, simple game. It's a simple
game. It's a simple game. It's

prefix_text = "Once in a "
generated_text = text_gen(prefix_text, max_
length=50, do_sample=False)[0]

print("-> #2 generated_text['generated_text']:")
print(generated_text['generated_text'])
print("------------------------------\n")

prefix_text = "Once in a blue "
generated_text = text_gen(prefix_text, max_
length=50, do_sample=False)[0]


print("-> #3 generated_text['generated_text']:")
print(generated_text['generated_text'])
print("------------------------------\n")
```

Listing rozpoczyna się od instrukcji import, a następnie inicjalizuje zmienną text_gen jako instancję klasy potoku, która również określa generowanie tekstu. Następnie zmienna prefix_text jest inicjalizowana jako ciąg tekstowy, a zmienna generated_text jest inicjalizowana danymi wyjściowymi z text_gen() i trzema parametrami, po czym wyświetlana jest zawartość zmiennej generated_text. Poprzednia sekwencja kroków jest powtarzana jeszcze dwukrotnie, z różnymi wartościami zmiennej prefix_text. Uruchom kod w Listing, a zobaczysz następujący wynik:

```
Some weights of GPT2Model were not initialized
from the model checkpoint at gpt2 and are newly
initialized: ['h.0.attn.masked_bias', 'h.1.attn.
masked_bias', 'h.2.attn.masked_bias', 'h.3.attn.
masked_bias', 'h.4.attn.masked_bias', 'h.5.attn.
masked_bias', 'h.6.attn.masked_bias', 'h.7.attn.
masked_bias', 'h.8.attn.masked_bias', 'h.9.attn.
masked_bias',
'h.10.attn.masked_bias', 'h.11.attn.masked_bias']
You should probably TRAIN this model on a down-
stream task to be able to use it for predictions
and inference.
Setting `pad_token_id` to `eos_token_id`:50256 for
open-end generation.
-> #1 generated_text['generated_text']:
What a wonderful thing about this is that it's a
very simple and simple way to get your hands on a
new game.

The game is a simple, simple game. It's a simple
game. It's a simple game. It's
-------------------------------

Setting `pad_token_id` to `eos_token_id`:50256 for
open-end generation.
-> #2 generated_text['generated_text']:
Once in a vernacular, the word "carnage" is used to
describe a large, open, and well-lit place.


The word "carnage" is used to describe a large,
open, and well-
-------------------------------

Setting `pad_token_id` to `eos_token_id`:50256 for
open-end generation.
-> #3 generated_text['generated_text']:
Once in a blue urn, you can see the "C" in the
center of the "C" and the "A" in the bottom right
corner.

The "C" is the "A" and the "A" are
-------------------------------
```

Modele typu „mix-and-match”

Teraz, gdy znasz już modele oparte na BERT i GPT-2, możliwe jest łączenie różnych typów modeli opartych na transformatorach, czyli łączenie modelu enkodera z modelem dekodera. Na przykład, chociaż nie podano tutaj żadnego kodu, możliwe jest łączenie BERT z GPT-2. Inna kombinacja polega na użyciu BERT zarówno jako enkodera, jak i dekodera. Ponieważ BERT jest modelem opartym na enkoderze i nie obsługuje funkcji opartych na dekoderze, wymagany jest dodatkowy kod. Na szczęście obiekt EncoderDecoderModel z Hugging Face zawiera metodę from_encoder_decoder_pretrained(), która działa jak „łącznik”, umożliwiający łączenie różnych typów wstępnie wytrenowanych modeli w celu utworzenia nowego modelu enkodera/dekodera. Skupmy się teraz na GPT-3, który jest tematem następnej sekcji.

CZYM JEST GPT-3?

Generative Pre-trained Transformer 3, czyli GPT-3, to jeden z najbardziej zaawansowanych modeli językowych opracowanych przez OpenAI. Jego wydanie oznaczało znaczący skok w możliwościach modeli językowych. GPT-3 stanowi znaczący krok naprzód w zakresie możliwości modeli językowych

sztucznej inteligencji. Jego zdolność do generalizacji na podstawie niewielu przykładów i generowania spójnego tekstu w szerokim zakresie tematów i zadań jest imponująca. Jednak, jak w przypadku wszystkich narzędzi, jego możliwości wiążą się z odpowiedzialnością za jego użytkowanie i potencjalne implikacje jego wyników. GPT-3 to trzecia iteracja serii GPT firmy OpenAI. Podczas gdy GPT-2, jego poprzednik, imponował już liczbą 1,5 miliarda parametrów, GPT-3 przyćmiewa go liczbą 175 miliardów parametrów, co czyni go największym modelem w serii GPT (stan na wrzesień 2021 r.). GPT-3, podobnie jak jego poprzednicy, wykorzystuje architekturę transformatorową, która stała się standardem dla modeli językowych SOTA ze względu na możliwość obsługi zależności dalekiego zasięgu w tekście. GPT-3 został wytrenowany na bazie danych licencjonowanych, danych tworzonych przez trenerów oraz danych publicznie dostępnych w wielu językach. Ta ogromna ilość danych, w połączeniu z ogromną liczbą parametrów, pozwala GPT-3 osiągnąć niezwykłą wydajność w szerokim zakresie zadań bez danych treningowych specyficznych dla danego zadania. Jedną z przełomowych cech GPT-3 jest możliwość uczenia się na podstawie kilku prób. Oznacza to, że wystarczy podać GPT-3 kilka przykładów zadania, a on spróbuje je uogólnić i wykonać. Jest to kontrast z wieloma innymi modelami, które wymagają gruntownego dostrojenia do konkretnych zadań. Na przykład, jeśli podasz GPT-3 kilka przykładów tłumaczenia z języka angielskiego na francuski, może on następnie spróbować przetłumaczyć nowe zdania, nawet jeśli nie zostały one wyraźnie dostrojone do tłumaczenia. Ze względu na imponującą wszechstronność GPT-3, istnieje mnóstwo zastosowań wykorzystujących GPT-3, takich jak:

- generowanie tekstu (np. pisanie esejów lub poezji)
- odpowiadanie na pytania w oparciu o dostarczone informacje
- tłumaczenie międzyjęzykowe
- symulowanie postaci w grach wideo
- generowanie kodu na podstawie opisu

Istnieją inne ważne kwestie dotyczące GPT-3 (co jest zasadniczo prawdą w przypadku wszystkich programów LLM), takie jak wiarygodność wyników. Chociaż GPT-3 może generować spójny tekst, z natury nie odróżnia prawdy od fałszu. W związku z tym może generować niepoprawne lub mylące odpowiedzi. Kolejnym problemem są kwestie etyczne związane z potencjalnym nadużyciem, takim jak generowanie mylących informacji, fałszywych wiadomości lub nieodpowiednich treści. Trzeci problem dotyczy intensywności wykorzystania zasobów: modele szkoleniowe, takie jak GPT-3, wymagają znacznych zasobów obliczeniowych, co z kolei podnosi kwestię śladu węglowego programów LLM. Czwartym problemem jest brak „wyjaśnialności". Nie wiemy dokładnie, dlaczego GPT-3 (i programy LLM ogólnie) dostarczają konkretnej odpowiedzi.

Cztery modele GPT-3

Istnieją cztery modele GPT-3, które różnią się liczbą parametrów i warstw. Najmniejszy model składa się z 12 warstw i 125 milionów parametrów, a największy z 96 warstw i 175 miliardów parametrów. Jak można się domyślić, zwiększenie liczby warstw i parametrów poprawia zdolność uczenia się, a jednocześnie zwiększa związane z tym koszty i czas przetwarzania. Reakcja opinii publicznej na GPT-3 wahała się od szoku, sceptycyzmu i niepokoju po zdumienie. Co ciekawe, nawet prezes OpenAI (Sam Altman) zgodził się, że niektóre stwierdzenia wyolbrzymiają możliwości GPT-3. Kluczowym wyróżnikiem GPT-3 jest jego zdolność do wykonywania określonych zadań bez konieczności precyzyjnego dostrajania. Z kolei inne modele zazwyczaj wymagają zestawów danych specyficznych dla danego zadania i generalnie nie sprawdzają się tak dobrze w przypadku zadań niezwiązanych z tym

zagadnieniem. GPT-3 jest w rzeczywistości rozszerzeniem modelu GPT-2, które zawiera więcej warstw i danych i jest 100 razy większe niż GPT-2. Chociaż GPT-3 został wytrenowany bez wiedzy o konkretnej domenie, może on wykonywać wiele zadań specyficznych dla danej domeny, w tym podsumowywanie tekstu i tłumaczenie językowe. W związku z tym GPT-3 może wykonywać zadania, w zakresie których nie został bezpośrednio wytrenowany. GPT-3 to model wstępnie wytrenowany. GPT-3 został wstępnie wytrenowany na ogromnej ilości danych i może obsługiwać wiele zadań w oparciu o uczenie się na małej liczbie prób. GPT-3 można trenować, wykorzystując dane w różnych formatach, w tym JSONL (linie JSON), zwykły tekst i CSV, najlepiej ustrukturyzowane w taki sposób, aby algorytm trenujący mógł przetwarzać dane i efektywnie się z nich uczyć. Przykład danych w formacie JSONL przedstawiono poniżej:

```
{"prompt": "<prompt text>", "completion": "<ideal
generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal
generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal
generated text>"}
```

Dla Twojej wygody, OpenAI udostępnia narzędzie CLI do przygotowywania danych do konwersji danych na dokument JSONL. Najważniejszymi następcami GPT-3 opartymi na transformatorach są Gopher, Chinchilla i PaLM, które wszystkie przewyższają możliwości GPT-3 (i zostaną omówione później). Modele oparte na transformatorach tekstu do obrazu, DALL-E 2, Imagen i Parti, zawierają dodatkowe techniki, które pobudzają wyobraźnię wielu osób ze świata grafiki. Ponadto Flamingo i Gato to dwie odmiany GPT-3, które obsługują funkcjonalność multimodalną (tj. tekst, dźwięk, obraz i wideo).

Architektura GPT-3

Ważnym aspektem modeli autoregresyjnych jest ich zdolność do przewidywania przyszłej wartości zmiennej docelowej na podstawie jej wartości bezpośrednio poprzedzającej: nie wymagają one historii poprzednich wartości. Natomiast modele regresyjne wykorzystują wartości wielu zmiennych do przewidywania przyszłej wartości zmiennej docelowej. Architektura GPT-3 różni się od modeli takich jak BERT i GPT-2, ponieważ ma następujące cechy:

• uwaga przyczynowa (w porównaniu z BERT)

• autoregresyjny model języka

• 96 warstw (gpt-3.5-turbo)

• wymiarowość 12288 i 96 głowic

• długość sekwencji 2048 (1024 dla GPT-2)

• osadzanie pozycji

• GELU w warstwie FF

• 175 miliardów parametrów

Wymagania i parametry GPT-3

Minimalne wymagania dla GPT-3 zależą od rozmiaru modelu, którego szczegóły przedstawiono tutaj:

• GPT-3 „mały” wymaga co najmniej 16 GB pamięci RAM i GPU z co najmniej 8 GB pamięci VRAM.

• GPT-3 „średni” wymaga co najmniej 32 GB pamięci RAM i GPU z

co najmniej 16 GB pamięci VRAM.

• GPT-3 „duży” wymaga 600 GB pamięci RAM i wyspecjalizowanego procesora graficznego z 250 GB pamięci VRAM.

Podane wartości są wartościami minimalnymi. Są to minimalne wymagania i można oczekiwać wyższych wartości dla dobrej wydajności. Chociaż GPT-3 ma parametry 175 B, nie jest porównywalny. Dla Twojej wygody, OpenAI udostępnia narzędzie CLI do przygotowywania danych do konwersji danych na dokument JSONL. Najważniejszymi następcami GPT-3 opartymi na transformatorach są Gopher, Chinchilla i PaLM, które wszystkie przewyższają możliwości GPT-3 (i zostaną omówione później). Modele oparte na transformatorach tekstu do obrazu, DALL-E 2, Imagen i Parti, zawierają dodatkowe techniki, które pobudzają wyobraźnię wielu osób ze świata grafiki. Ponadto Flamingo i Gato to dwie odmiany GPT-3, które obsługują funkcjonalność multimodalną (tj. tekst, dźwięk, obraz i wideo).

Architektura GPT-3

Ważnym aspektem modeli autoregresyjnych jest ich zdolność do przewidywania przyszłej wartości zmiennej docelowej na podstawie jej wartości bezpośrednio poprzedzającej: nie wymagają one historii poprzednich wartości. Natomiast modele regresyjne wykorzystują wartości wielu zmiennych do przewidywania przyszłej wartości zmiennej docelowej. Architektura GPT-3 różni się od modeli takich jak BERT i GPT-2, ponieważ ma następujące cechy:

• uwaga przyczynowa (w porównaniu z BERT)

• autoregresyjny model języka

• 96 warstw (gpt-3.5-turbo)

• wymiarowość 12288 i 96 głowic

• długość sekwencji 2048 (1024 dla GPT-2)

• osadzanie pozycji

• GELU w warstwie FF

• 175 miliardów parametrów

Wymagania i parametry GPT-3

Minimalne wymagania dla GPT-3 zależą od rozmiaru modelu, którego szczegóły przedstawiono tutaj:

• GPT-3 „mały” wymaga co najmniej 16 GB pamięci RAM i GPU z co najmniej 8 GB pamięci VRAM.

• GPT-3 „średni” wymaga co najmniej 32 GB pamięci RAM i GPU z

co najmniej 16 GB pamięci VRAM.

• GPT-3 „large” wymaga 600 GB pamięci RAM i wyspecjalizowanego procesora graficznego z 250 GB pamięci VRAM.

Podane na tej liście wartości są wartościami minimalnymi. Są to minimalne wymagania, a dla uzyskania dobrej wydajności można oczekiwać wyższych wartości. Chociaż GPT-3 ma 175 B parametrów, nie jest porównywalny z innymi modelami LLM, które mają 540 B parametrów, a nawet 1 bilion parametrów. Zdecydowana większość parametrów GPT-3 jest wykorzystywana w procesach obliczeniowych, które umożliwiają modelowi zrozumienie i generowanie tekstu na podstawie wzorców poznanych podczas

treningu. Na poprzedniej stronie internetowej omówiono sześć strategii uzyskiwania lepszych wyników za pomocą podpowiedzi.

MODELE GPT-3

Główną różnicą między modelami GPT-3 jest liczba parametrów i warstw: jeden model zawiera 12 warstw i 125 milionów parametrów, podczas gdy inny model zawiera 96 warstw i 175 miliardów parametrów. Należy pamiętać, że dodawanie warstw i parametrów zwiększa wydajność modelu, a jednocześnie zwiększa koszty i czas przetwarzania. Dla porównania, GPT-3 ma 175 miliardów parametrów, GPT-2 ma 1,5 miliarda parametrów, a BERT-Large ma 340 milionów parametrów. Model GPT-3 został wytrenowany w całości na publicznie dostępnych zbiorach danych, obejmujących blisko 500 000 000 000 słów (niektóre z nich mogą zawierać treści obraźliwe). Jedną z zalet GPT-3 są możliwości uczenia się na podstawie niewielkiej liczby powtórzeń. Można podawać dłuższe podpowiedzi z poprawnymi przykładami, aby poprawić jakość odpowiedzi modelu GPT-3. Takie przykłady działają jak trening w czasie rzeczywistym i poprawiają wyniki zwracane przez GPT-3 bez modyfikowania jego parametrów. OpenAI pobiera opłatę za tokeny w monicie wejściowym oraz za tokeny wyjściowe zwracane w odpowiedzi z GPT-3. Największy model GPT-3 ma 96 warstw uwagi, z których każda zawiera nagłówki o wymiarach 96x128. Ponadto GPT-3 składa się ze 175 miliardów parametrów i został wytrenowany na ponad 500 gigabajtach tekstu, aby nauczyć się przewidywać następne słowo w ciągu tekstowym dostarczonym przez użytkownika. Modele GPT-n są trenowane na nieoznakowanym korpusie tekstowym, a nie na danych z etykietami, i dokonują precyzyjnego dostrajania dla poszczególnych zadań. Od momentu premiery w maju 2020 roku, OpenAI stale dodaje nowe funkcjonalności do GPT-3. Na przykład, GPT-3 umożliwia edycję tekstu i modyfikowanie bloków kodu.

MODELE GPT-3

Główną różnicą między modelami GPT-3 jest liczba parametrów i warstw: jeden model zawiera 12 warstw i 125 milionów parametrów, podczas gdy inny model zawiera 96 warstw i 175 miliardów parametrów. Należy pamiętać, że dodawanie warstw i parametrów zwiększa wydajność modelu, a jednocześnie zwiększa koszty i czas przetwarzania. Dla porównania, GPT-3 ma 175 miliardów parametrów, GPT-2 ma 1,5 miliarda parametrów, a BERT-Large ma 340 milionów parametrów. Model GPT-3 został wytrenowany w całości na publicznie dostępnych zbiorach danych, obejmujących blisko 500 000 000 000 słów (niektóre z nich mogą zawierać treści obraźliwe). Jedną z zalet GPT-3 są możliwości uczenia się na podstawie niewielkiej liczby powtórzeń. Można podawać dłuższe podpowiedzi z poprawnymi przykładami, aby poprawić jakość odpowiedzi modelu GPT-3. Takie przykłady działają jak trening w czasie rzeczywistym i poprawiają wyniki zwracane przez GPT-3 bez modyfikowania jego parametrów. OpenAI pobiera opłatę za tokeny w monicie wejściowym oraz za tokeny wyjściowe zwracane w odpowiedzi z GPT-3. Największy model GPT-3 ma 96 warstw uwagi, z których każda zawiera nagłówki o wymiarach 96x128. Ponadto GPT-3 składa się ze 175 miliardów parametrów i został wytrenowany na ponad 500 gigabajtach tekstu, aby nauczyć się przewidywać następne słowo w ciągu tekstowym dostarczonym przez użytkownika. Modele GPT-n są trenowane na nieoznakowanym korpusie tekstowym, a nie na danych z etykietami, i dokonują precyzyjnego dostrajania dla poszczególnych zadań. Od momentu premiery w maju 2020 roku, OpenAI stale dodaje nowe funkcjonalności do GPT-3. Na przykład, GPT-3 umożliwia edycję tekstu i modyfikowanie bloków kodu. Microsoft ogłosił 22 września 2020 roku, że uzyskał licencję na wyłączne użytkowanie GPT-3; Chociaż każdy może korzystać z publicznego API i eksperymentować z GPT-3 Playground, tylko Microsoft ma dostęp do bazowego modelu GPT-3. Rodzina GPT-3 obsługuje następujące publicznie dostępne modele, z których każdy ma inną liczbę parametrów (i inną cenę):

• GPT-3 Ada

• GPT-3 Babbage

• GPT-3 Curie

• GPT-3 Davinci

Model Ada jest najszybszy/najtańszy, natomiast model Davinci jest najwydajniejszy/najdroższy. Cztery największe modele mają od 2,7 miliarda do 175 miliardów parametrów. Na podstawie tych szczegółów możemy spekulować, że:

• Ada ma 2,7 miliarda parametrów.

• Babbage ma 6,7 miliarda parametrów.

• Curie ma 13 miliardów parametrów.

• Davinci ma 175 miliardów parametrów.

Silniki GPT-3 Ada i Davinci różnią się 75-krotnym współczynnikiem kosztów.

GPT-3 kontra modele wielojęzyczne

W kontekście wielojęzyczności języki o ograniczonej ilości danych stanowią wyzwanie. Jednym z podejść, które może okazać się korzystne, jest technika łączenia tekstów w wielu językach w jeden zbiór danych, który jest faktycznie wykorzystywany w zbiorze danych MM z Facebooka, umożliwiającym tłumaczenie tekstu między dowolną parą 100 języków. Innym interesującym rezultatem jest wykorzystanie modelu LLM wyszkolonego w jednym języku do wykonywania zadań w innym języku, co może również generować wyniki w językach o ograniczonej ilości danych. Dla przypomnienia, GPT-3 należy do grupy wielkoskalowych autoregresyjnych modeli językowych, które charakteryzują się następującymi cechami:

• modele uczące się w oparciu o niewielką liczbę prób

• nie wymagają dostrajania

• wykonują wiele zadań językowych

Jednak takie modele są trenowane głównie na zbiorach danych w języku angielskim i są lepsze od innych modeli językowych trenowanych na zbiorach danych w języku innym niż angielski. W szczególności modele oparte na XGLM przewyższają GPT-3 w wielu zadaniach opartych na języku.

JAKI JEST CEL GPT-3?

Celem wstępnie wytrenowanego modelu GPT-3 jest bezpośrednia ocena modelu na podstawie danych testowych nowych zadań; tj. GPT-3 zasadniczo pomija dane treningowe nowych zadań i koncentruje się bezpośrednio na danych testowych, działając jako model uczący się w oparciu o niewielką liczbę prób. GPT-3 osiągnął wydajność SOTA w kilku zadaniach NLP bez dostrajania, kosztem ponad 10 000 000 dolarów. Niektóre zestawy danych użyte do trenowania GPT-3 można pobrać z tego repozytorium Github (tylko do odczytu): https://github.com/openai/gpt-3

GPT-3 przykuł uwagę wielu osób ze względu na różne zadania, które wykonywał, w tym automatyczne generowanie kodu. Na przykład, jeden z użytkowników wpisał akapit tekstu opisujący następującą aplikację internetową:

1. przycisk zwiększający sumę o 3 USD

2. przycisk zmniejszający sumę o 5 USD

3. przycisk wyświetlający bieżącą sumę

GPT-3 utworzył następnie aplikację React z poprzednią funkcjonalnością, co wywołało różne reakcje: niektórych rozbawiła tak uproszczona aplikacja, podczas gdy inni zastanawiali się nad swoim przyszłym bezpieczeństwem pracy. Podaj GPT-3 początkową sekwencję słów, a GPT-3 wygeneruje różne odpowiedzi, takie jak generowanie kodu, artykuły prasowe, wiersze, a nawet żarty. GPT-3 wygenerował interesujący wiersz o Elonie Musku („Twoje tweety to plaga"), którego fragment można przeczytać tutaj:

https://www.businessinsider.com/elon-musk-poem-tweets-gpt-3-openai-2020-8

Jedną z wyróżniających cech GPT-3 jest jego zdolność do rozwiązywania niewidocznych zadań NLP: wynika to z faktu, że GPT-3 został wytrenowany na bardzo dużym korpusie. GPT-3 wykorzystuje również uczenie się metodą „fragmentów" (ang. few-shot learning) i może wykonywać następujące zadania:

• tłumaczenie języka naturalnego na kod dla stron internetowych

• rozwiązywanie złożonych problemów medycznych typu „pytanie-odpowiedź"

• tworzenie tabelarycznych raportów finansowych

• pisanie kodu do trenowania modeli uczenia maszynowego

GPT-3 zaskoczył ludzi swoją zdolnością do generowania zarówno prozy, jak i poezji. Elon Musk jest jednym z założycieli OpenAI, którzy stworzyli GPT-3, na podstawie którego powstał poniższy wiersz o Elonie Musku (dostępny pod adresem https://www.businessinsider.com/elon-musk-poem-tweets-gpt-3-openai-2020-8):

„Komisja Papierów Wartościowych i Giełd (SEC) powiedziała: »Musk,/twoje tweety to plaga./Mogą cię naprawdę kosztować pracę,/jeśli nie przestaniesz/tweetować w nocy«./… Wtedy Musk krzyknął: »Dlaczego?/Tweety, które napisałem, nie są złośliwe,/nie piszę wielkimi literami/i jestem pewien, że moje tweety są czyste./Ale twoje tweety potrafią poruszyć rynki/i dlatego jesteśmy wściekli./Możesz być geniuszem/i miliarderem,/ale to nie daje ci prawa do bycia nudziarzem!«

CO MOŻE ZROBIĆ GPT-3?

Odpowiedzi tekstowe generowane przez GPT-3 mogą osiągnąć jakość porównywalną z ludzką. Jednak obecnie nie ma wsparcia dla formatów danych, takich jak audio czy wideo. GPT-3 może jednak współpracować z danymi opartymi na obrazach za pomocą DALL-E. GPT-3 to model statystyczny, który określa rozkład prawdopodobieństwa słów w celu wygenerowania odpowiedniego tekstu w odpowiedzi na ciąg wejściowy. W przeciwieństwie do innych modeli trenowanych pod kątem konkretnych zadań, GPT-3 został zaprojektowany do wykrywania wzorców w ciągach tekstowych, co zapewnia następujące korzyści:

• GPT-3 to model ogólnego przeznaczenia (nie specyficzny dla danego zadania).

• GPT-3 nie wymaga ponownego trenowania, aby obsługiwać nowe monity.

• GPT-3 osiąga wydajność SOTA w przypadku wielu zadań przetwarzania języka naturalnego (NLP).

Wydajność GPT-3 prawdopodobnie wynika z faktu, że GPT-3 został wytrenowany na zbiorze danych liczącym ponad 500 gigabajtów. Chociaż rozmiar i koszt GPT-3 są zaporowe dla zdecydowanej większości firm, GPT-3 jest dostępny za pośrednictwem prostego i ekonomicznego API. Należy pamiętać, że GPT-3 opiera się na danych z określonego momentu, które mogą pochodzić sprzed kilku miesięcy, zamiast na ciągłym trenowaniu.

Sztuczna inteligencja konwersacyjna

Aspekt sztucznej inteligencji konwersacyjnej GPT-3 może być wykorzystany w różnych implementacjach, z których niektóre obejmują:

• generowanie recenzji (np. filmów i restauracji)

• wstępne przetwarzanie danych wprowadzanych przez użytkownika

• reklamy

• chatboty

Należy jednak zachować ostrożność w przypadku sztucznej inteligencji. Jeden z medycznych chatbotów opartych na GPT-3 zasugerował fałszywemu pacjentowi (który miał myśli samobójcze), aby popełnił samobójstwo: https://artificialintelligence-news.com/2020/10/28/medical-chatbot-openai-gpt3-patient-kill-themselves/

GPT-3 ma osiem różnych rozmiarów modelu (od 125 MB do 175 MB parametrów), a najmniejszy model GPT-3 ma rozmiar zbliżony do BERT-Base i RoBERTa-Base, z 12 warstwami uwagi, które z kolei mają głowicę wymiarową 12x64. Dla porównania, największy model GPT-3 jest dziesięć razy większy niż T5-11B (poprzedni rekordzista) i ma 96 warstw uwagi, które z kolei mają głowice wymiarowe 96x128. Szczegóły dotyczące dostosowywania modelu GPT-3 znajdują się na stronie https://openai.com/blog/customized-gpt3/. Ponadto OpenAI dodało obsługę precyzyjnego dostrajania GPT-3, która może być również dostępna dla innych modeli OpenAI. Podsumowanie tekstu i podsumowanie GPT-3 obsługuje następujące typy podsumowania tekstu, jak pokazano na poniższej liście:

• proste podsumowanie

• podsumowanie jednozdaniowe

• podsumowanie dostosowane do poziomu klasy

• podsumowanie ekstrakcji

Można znaleźć artykuły sugerujące dołączenie ciągu tl;dr; do bloku tekstu, aby GPT-3 wygenerował proste podsumowanie. Jako prostą ilustrację, wprowadź następujący blok tekstu w GPT-3 Playground :

```
GPT-3 has eight different model sizes (from 125M
to 175B parameters), and the smallest GPT-3 model
is about the size of BERT-Base and RoBERTa-Base,
with 12 attention layers that in turn have 12x64
dimension head. On the other hand, the largest
GPT-3 model is ten times larger than T5-11B (the
previous record holder), and has 96 attention
layers, which in turn have 96x128 dimension heads.
Details regarding how to customize a GPT3 model are
here
tl;dr;
```

Odpowiedź GPT-3 jest następująca:

```
GPT-3 has a much larger parameter count than any
other transformer-based model, making it the
largest transformer-based model ever trained.
```

Zwróć uwagę, że ciąg tl;dr; w bloku tekstu wejściowego znajduje się w osobnym wierszu: jeśli umieścisz ten ciąg w tym samym wierszu co tekst wejściowy, GPT-3 wygeneruje odpowiedź wielowierszową zamiast podsumowania jednozdaniowego. Zapoznaj się z sekcją dotyczącą klasyfikacji tekstu w rozdziale 8, aby zapoznać się z przykładem kodu w Pythonie, który pokazuje, jak przeprowadzić klasyfikację tekstu.

OGRANICZENIA GPT-3

GPT-3 może przetwarzać tylko tekst, co jest ograniczeniem, a nie słabością. Poniższy obszerny artykuł omawia różne słabości GPT-3: https://nautil.us/deep-learning-is-hitting-a-wall-14467

GPT-3 ma inne ograniczenia, które uniemożliwiają GPT-3 wykonywanie dowolnych zadań związanych z tekstem, jak pokazano na poniższej liście:

• ograniczony rozmiar danych wejściowych

• ograniczony rozmiar danych wyjściowych

• brak typu pamięci

Obecnie GPT-3 ma problemy z zadaniami wnioskowania języka naturalnego, takimi jak ANLI, lub zadaniami obejmującymi przyczynowość. Mimo że GPT-3 potrafi generować artykuły wysokiej jakości, w celu zapewnienia wysokiego stopnia spójności tekstu przeprowadzana jest edycja ludzka.

Mocne i słabe strony zadania GPT-3

Pomimo swoich ograniczeń, GPT-3 potrafi generować tekst o jakości zbliżonej do ludzkiej. Załóżmy na przykład, że GPT-3 otrzymuje tytuł i podtytuł oraz słowo „Artykuł", które służy jako słowo kluczowe. GPT-3 może następnie tworzyć krótkie artykuły, które często sprawiają wrażenie napisanych przez ludzi. Należy pamiętać, że w korpusie danych użytym do trenowania GPT-3 występuje stronniczość. GPT-3 nie potrafi integrować ani przeprowadzać rozumowania, które obejmuje prozaiczne zadania. Co więcej, GPT-3 generuje błędne wyniki, takie jak chatbot Replika, oparty na GPT-3, który „twierdził, że Bill Gates wynalazł COVID-19" i że szczepionki przeciwko COVID-19 są „niezbyt skuteczne". Według niektórych badaczy, programy nauczania języka angielskiego oparte na uczeniu głębokim są jak „papugi stochastyczne", które „wiele powtarzają, a niewiele rozumieją". Być może niespójności GPT-3 częściowo przyczyniły się do rozwoju GPT-4.

WYDAJNOŚĆ ZADAŃ GPT-3

W przypadku większości modeli zadanie tłumaczenia zdań z angielskiego na włoski wymaga tysięcy par zdań, aby modele te nauczyły się, jak wykonywać tłumaczenie. Dla porównania, GPT-3 nie wymaga etapu precyzyjnego dostrajania: może obsługiwać niestandardowe zadania językowe bez danych treningowych. Zatem GPT-3 ma zdolność do wykonywania określonych zadań bez specjalnego dostrajania, czego inne modele nie potrafią dobrze. Na przykład, GPT-3 można wytrenować do tłumaczenia tekstu, generowania kodu, a nawet pisania poezji. Co więcej, GPT-3 może to zrobić przy nie więcej niż 10 przykładach treningowych. GPT-3 nie jest jedynie uczniem z kilkoma próbami: może również działać jako uczeń z zerową próbą i uczeń z pojedynczą próbą. Dla porównania, GPT-3 jako uczeń z zerową próbą ma wyższą dokładność niż dostrojony model RoBERTa (który wcześniej miał wydajność SOTA). Pod względem rozumienia tekstu czytanego, GPT-3 najlepiej sprawdza się w zbiorach danych konwersacyjnych o swobodnej formie, a najgorzej w zbiorach danych obejmujących modelowanie ustrukturyzowanego dialogu. Jednak jako uczeń z kilkoma próbami w tym zadaniu, GPT-3 przewyższa dostrojony model bazowy BERT. Ponadto GPT-3 dobrze radzi sobie w zestawie danych SQuAD 2.0 ze Stanford, ale gorzej radzi sobie z pytaniami wielokrotnego wyboru. GPT-3 traktuje każdy ciąg wejściowy jako podpowiedź, aby określić najodpowiedniejszą odpowiedź: podpowiedzi o wyższej jakości generują odpowiedzi o wyższej jakości. Uzupełnienie to inne określenie ciągu odpowiedzi generowanego przez GPT-3.

CZYM RÓŻNIĄ SIĘ GPT-3 I BERT

Ta sekcja zawiera porównanie niektórych różnic między GPT-3 a BERT. Chociaż szczegóły są dostępne w wielu miejscach w tym i poprzednim rozdziale, ta sekcja zapewnia wygodniejsze zestawienie tych informacji. GPT-3 (model autoregresyjny) i BERT (model dwukierunkowy) uzyskują kontekst językowy na różne sposoby. W szczególności GPT-3 generuje dane w przód z danych wejściowych języka naturalnego, podczas gdy BERT może uzyskać kontekst językowy w obu kierunkach z konkretnego słowa. Co więcej, GPT-3 nie wymaga precyzyjnego dostrajania do wykonywania zadań, co różni się od innych wcześniejszych modeli przetwarzania języka naturalnego. Jednak precyzyjne dostrajanie za pomocą API OpenAI może skutkować poprawą wydajności w określonych zadaniach. Kolejną ważną zaletą GPT-3 w porównaniu z sieciami neuronowymi (RNN) i LSTM jest jego zdolność do utrzymywania informacji kontekstowej między słowami, które są znacznie bardziej oddalone od siebie w korpusie. Podczas gdy LSTM mogą utrzymywać kontekst dla słów oddalonych od siebie o kilkanaście pozycji, GPT-3 zapewnia znacznie szerszy zakres kontekstowy. Trzecią zaletą GPT-3 jest jego lepsza wydajność w porównaniu z uczeniem się na małej liczbie prób. Ponadto GPT-3 został wytrenowany na tak ogromnej ilości danych, że nie ma potrzeby trenowania GPT-3 na wielu zadaniach. Być może częściowo wyjaśnia to znaczną poprawę GPT-3 w porównaniu z mniejszymi modelami w tych zadaniach. Osiągnięcia GPT-3 sugerują, że rozmiar modelu wpływa na jego zdolność do prawidłowego wykonywania zadań. Mówiąc w uproszczeniu, autoregresja polega na uczeniu się jednokierunkowym, podczas gdy systemy dwukierunkowe – na uczeniu się dwukierunkowym. Oznacza to, że systemy regresyjne uczą się na podstawie poprzedniej sekwencji, aby przewidzieć kolejny element, podczas gdy systemy dwukierunkowe mogą wykorzystywać zarówno informacje przyszłe, jak i przeszłe. Z kolei BERT jest dwukierunkowy, co oznacza, że może uczyć się na podstawie wszystkiego, co znajduje się w kontekście modelu. Mając na uwadze powyższe punkty, BERT (i inne modele dwukierunkowe) doskonale sprawdza się w klasyfikacji dokumentów lub pytaniach sprawdzających rozumienie tekstu, gdzie przetwarza się coś jako całość, a następnie udziela odpowiedzi na jakiś temat. GPT-3 (i inne modele autoregresyjne) umożliwiły paradygmat „małej liczby prób", ponieważ przetwarza dane wejściowe, a następnie generuje dane wyjściowe. Transfer learning odnosi się do rezultatu precyzyjnego dostrojenia wstępnie wytrenowanego modelu. W rzeczywistości nawet zbiór danych składający się ze 100 próbek

do precyzyjnego dostrojenia może skutkować znaczną poprawą w wstępnie wytrenowanym modelu. Nadal jednak istnieje luka wydajnościowa między precyzyjnym dostrojeniem a uczeniem się „małej liczby prób”. Chociaż wydajność „małej liczby prób” jest bardzo dobra, a czasami nawet porównywalna z wydajnością na poziomie ludzkim, jakość wyników jest niespójna.

PLAC ZABAW GPT-3

OpenAI stworzyło GPT-3 Playground, narzędzie internetowe do wprowadzania monitów w polu tekstowym i odbierania uzupełnień z GPT-3. Plac zabaw obsługuje większość funkcji dostępnych bezpośrednio poprzez API GPT-3. Ponadto, umożliwia interakcję z GPT-3 bez pisania kodu. W istocie, OpenAI Playground umożliwia łatwe wykorzystanie GPT-3 do trenowania silnika w celu generowania danych tekstowych. Plac zabaw GPT-3 udostępnia również zestaw zapisanych monitów, zwanych ustawieniami wstępnymi. Pierwszym krokiem jest przejście do Placu zabaw GPT-3, gdzie zostaniesz poproszony o zalogowanie się na swoje konto: https://beta.openai.com/playground. Po zarejestrowaniu konta zobaczysz stronę główną, która składa się z trzech sekcji: sekcji „Rozpoczęcie”, sekcji „Plac zabaw” oraz sekcji, która zawiera listę rozwijaną i suwaki do dostrajania parametrów. Środkowa sekcja to tekst wejściowy dla GPT-3, który składa się z dwóch części:

1) sekwencji początkowej, czyli ciągu tekstowego „Text:”, po którym następuje jeden lub więcej bloków tekstowych (dostarczonych przez Ciebie), które dostarczają GPT-3 przykładowy tekst wyjściowy.
2) Drugi akapit zawiera ten sam ciąg tekstowy „Text:”, który wskazuje koniec tekstu wejściowego.

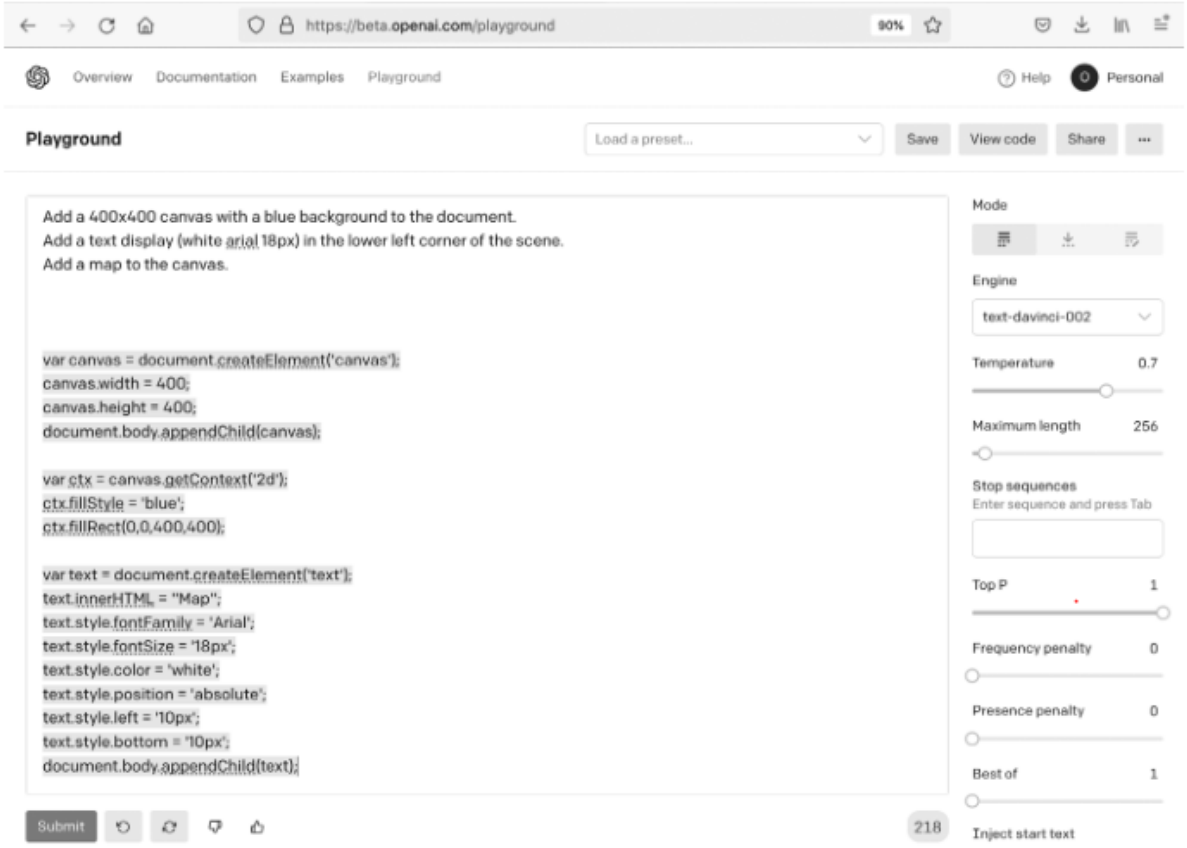
Odłóżmy opis suwaków, które pojawiają się wPlacu Zabaw GPT-3, na późniejszą część tego rozdziału i przejdźmy od razu do generowania kodu w Placu Zabaw GPT-3.

Generowanie kodu GPT-3 za pomocą silnika Davinci

Przejdź do Placu Zabaw GPT-3 i po sprawdzeniu, czy silnik Davinci jest wybrany, wprowadź następujące trzy zdania w Placu Zabaw (tj. w środkowej części ekranu):

```
Add a 400x400 canvas with a blue background to the
document.
Add a text display in the lower left corner of the
scene.
Add a map to the canvas.
```

Kliknij przycisk „Wyślij” w lewym dolnym rogu ekranu. GPT-3 generuje dane wyjściowe widoczne na rysunku.



Dla Twojej wygody Listing wyświetla zawartość pliku tekstowego gpt_3_generated_code.txt.

```
Add a 400x400 canvas with a blue background to the
document.
Add a text display in the lower left corner of the
scene.
Add a map to the canvas.

var canvas = document.createElement('canvas');
canvas.width = 400;
canvas.height = 400;
document.body.appendChild(canvas);

var ctx = canvas.getContext('2d');
ctx.fillStyle = 'blue';
ctx.fillRect(0,0,400,400);

var text = document.createElement('text');
text.innerHTML = "Map";
text.style.fontFamily = 'Arial';
text.style.fontSize = '18px';
text.style.color = 'white';
text.style.position = 'absolute';
text.style.left = '10px';
text.style.bottom = '10px';
document.body.appendChild(text);
```

GPT-3 może generować różne dane wyjściowe z tego samego zestawu monitów, co oznacza, że Twoje dane wyjściowe mogą różnić się od tych z Listingu. Na przykład, Listing wyświetla inny wygenerowany kod z tymi samymi trzema monitami, co Listing wcześniej.

```
Add a 400x400 canvas with a blue background to the
document.
Add a text display in the lower left corner of the
scene.
Add a map to the canvas.

<canvas id="myCanvas" width="400" height="400"
style="background-color:blue;"></canvas>

<script>

var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");

// add text display
ctx.font = "30px Arial";
ctx.fillStyle = "#ffffff";
ctx.fillText("Hello World", 10, 50);

// add map
var map = new Image();
map.onload = function(){
  ctx.drawImage(map, 0, 0);
}
map.src = "map.png";

// add player
var player = new Image();
player.onload = function(){
  ctx.drawImage(player, 0, 0);
}
player.src = "player.png";
```

Listing zawiera komunikaty tekstowe z instrukcjami tworzenia i umieszczania elementów interfejsu użytkownika na ekranie, a także dane wyjściowe wygenerowane przez GPT-3 za pomocą silnika Davinci. Początkowa część Listingu zawiera te same komunikaty, a najbardziej zewnętrzny element i jego zawartość to wygenerowana odpowiedź z GPT-3, która jest pokazana w Listingu.

```
Add a 400x400 canvas with a blue background to the
document.
Create 3 text input boxes that accept a digit


Make the height of the digits equal to 80px
Center the digit in each input box
Place the text input boxes in the same row
Place the row at the bottom of the document
Make the boxes 25% of the width of the div
Make the boxes 50% of the height of the div

<div style="width:80%;height:80%;text-align:center;
position:relative;bottom:0;">
  <input type="text" style="width:25%;height:50%;
  font-size:80px;text-align:center;"/>
  <input type="text" style="width:25%;height:50%;
  font-size:80px;text-align:center;"/>
  <input type="text" style="width:25%;height:50%;
  font-size:80px;text-align:center;"/>
</div>
```

PARAMETRY WNIOSKOWANIA

Po zakończeniu etapu dostrajania modelu LLM można ustawić wartości różnych parametrów wnioskowania. Interfejs API GPT-3 obsługuje wiele parametrów wnioskowania, z których niektóre przedstawiono poniżej:

• engine (mechanizm)

• prompt (monit)

• max_tokens (maks. tokeny)

• top_p (top_k)

• frequency_penalty (kara za częstotliwość)

• presence_penalty (kara za obecność)

• długość tokena

• stop tokeny

• temperature (temperatura)

Parametr wnioskowania silnika może być jednym z czterech modeli GPT-3, takich jak text-ada-001. Parametr prompt to po prostu tekst wejściowy, który podajesz. Parametr wnioskowania presence_penalty (kara za obecność) umożliwia uzyskanie bardziej trafnych odpowiedzi po określeniu wyższych wartości tego parametru. Parametr wnioskowania max_tokens określa maksymalną liczbę tokenów: przykładowe wartości to 100, 200 lub 256. Parametr wnioskowania top_p może być dodatnią liczbą całkowitą, która określa najwyższe wyniki do wybrania. Parametr frequency_penalty to parametr wnioskowania, który dotyczy częstotliwości powtarzania słów. Niższa wartość tego parametru zwiększa liczbę powtarzanych słów. Parametr długości tokenu określa całkowitą liczbę słów w sekwencji wejściowej przetwarzanej przez algorytm LLM (a nie maksymalną długość każdego tokenu). Parametr stop tokens kontroluje długość generowanego wyniku algorytmu LLM. Jeśli ten parametr jest równy 1, generowane jest tylko jedno zdanie, natomiast wartość 2 oznacza, że wygenerowany wynik jest ograniczony do jednego akapitu. Parametr top_k określa liczbę wybranych tokenów (k), z ograniczeniem, że wybrane tokeny mają najwyższe prawdopodobieństwo. Na przykład, jeśli k top jest równe 3, wówczas wybierane są tylko 3 tokeny o najwyższych prawdopodobieństwach. Parametr top_p jest liczbą zmiennoprzecinkową z zakresu od 0,0 do 1,0 i stanowi górną granicę sumy prawdopodobieństw wybranych tokenów. Na przykład, jeśli dyskretny rozkład prawdopodobieństwa składa się ze zbioru S = {0,1, 0,2, 0,3, 0,4}, a wartość parametru „p top" wynosi 0,3, wówczas można wybrać tylko tokeny o powiązanych prawdopodobieństwach 0,1 i 0,2. Zatem parametry k top i p top zapewniają dwa mechanizmy ograniczania liczby tokenów, które można wybrać.

Parametr temperatury

Parametr wnioskowania temperatury jest liczbą zmiennoprzecinkową z zakresu od 0 do 1 włącznie, a jego wartość domyślna wynosi 0,7. Jedną z interesujących wartości temperatury jest 0,8: spowoduje to, że GPT-3 wybierze kolejny token, który nie ma maksymalnego prawdopodobieństwa. Parametr wnioskowania temperaturowego T jest nieujemną liczbą zmiennoprzecinkową, której wartość wpływa na stopień, w jakim model wykorzystuje losowość. Mniejsze wartości parametru temperatury, bliższe 0, oznaczają mniejszą losowość (tj. są bardziej deterministyczne), podczas gdy większe wartości parametru temperatury oznaczają większą losowość. Parametr temperatury T jest bezpośrednio powiązany z funkcją softmax(), która jest stosowana w ostatnim kroku architektury transformatora.

Wartość T zmienia formułę funkcji softmax, jak opisano w dalszej części tej sekcji. Ważne jest, aby pamiętać, że wybór tokenów na podstawie funkcji softmax() oznacza, że wybrany token jest tokenem o najwyższym prawdopodobieństwie. Z kolei większe wartości parametru T umożliwiają losowość w wyborze kolejnego tokena, co oznacza, że token może zostać wybrany, nawet jeśli powiązane z nim prawdopodobieństwo jest mniejsze niż maksymalne prawdopodobieństwo. Choć może się to wydawać nieintuicyjne, niektóre wartości T (takie jak 0,8) dają tekst wyjściowy brzmiący bardziej naturalnie z perspektywy człowieka niż tekst wyjściowy, w którym tokeny są wybierane, jeśli mają maksymalne prawdopodobieństwo. Wreszcie, wartość temperatury równa 1 jest tym samym, co standardowa funkcja softmax(). Nawiasem mówiąc, sekcja „Uzupełnianie tekstu i temperatura" w rozdziale 8 zawiera przykład kodu w Pythonie, który wywołuje klasę Completion w OpenAI w celu uzupełnienia tekstu z dwiema różnymi wartościami temperatury (wyniki są interesujące).

Temperatura i funkcja softmax()

Parametr temperatury T pojawia się w mianowniku wykładnika stałej Eulera e w funkcji softmax(). Zatem zamiast licznika softmax() w postaci e^(xi), zmodyfikowana funkcja soft-max() zawiera wyrazy licznika w postaci e^(xi/T), gdzie {x1, x2, . . . , xn} tworzą zbiór liczb tworzących dyskretny rozkład prawdopodobieństwa. Dla przypomnienia, mianownik każdego wyrazu wygenerowanego przez funkcję soft-max() składa się z sumy wyrazów w zbiorze {e^(x1), e^(x2), . . . , e^(xn)}. Jednak mianownik wyrazów z parametrem temperatury T jest nieco inny: jest to suma wyrazów w zbiorze {e^(x1/T), e^(x2/T), . . . , e^(xn/T)}. Co ciekawe, funkcja softmax() z parametrem temperatury T jest taka sama jak rozkład Boltzmanna. Aby uzyskać więcej informacji na temat regulowanych parametrów wnioskowania, należy przeszukać internet. Poniższy fragment kodu w Pythonie zawiera przykład określania wartości różnych parametrów wnioskowania, w tym silnika GPT-3:

```
response = openai.Completion.create(
  engine="text-ada-001",
  prompt="",
  temperature=0.7,

  max_tokens=256,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0
)
```

PRZEGLĄD INŻYNIERII PROMPT

Różne modele językowe wymagają dostrojenia, aby wykonywać zadania niższego rzędu, co czasami wiąże się z koniecznością dołączenia jednej lub więcej warstw do modelu. Z kolei GPT-3 opiera się na pojedynczym modelu, który nie wymaga dodatkowych warstw ani dostrajania, aby wykonywać dodatkowe zadania. W rzeczywistości GPT-3 wykorzystuje system, w którym do danych wejściowych modelu można dodać zero, jeden lub kilka przykładów. Połączenie opisu zadania, przykładów (jeśli istnieją) i promptu zapewnia GPT-3 kontekst, dzięki czemu GPT-3 może przewidywać dane wyjściowe token po tokenie. Załóżmy na przykład, że opis zadania to zdanie „Przetłumacz z angielskiego na włoski", po którym następuje kilka par słów angielskich/włoskich jako przykłady. Wówczas podpowiedź byłaby angielskim słowem (np. „krowa"), które model tłumaczy na włoski („vacca"). Praca z podpowiedziami Zasadniczo podpowiedź to ciąg tekstowy, który dostarcza GPT-3 instrukcji dotyczących rodzaju odpowiedzi oczekiwanej na ciąg wejściowy (tj. podpowiedź). Ta technika jest zaskakująco skuteczna w pracy z modelami językowymi. Na przykład, jeśli prześlesz tę samą podpowiedź kilka razy do GPT-3, wyniki nie będą spójne (choć podobne). Jednym ze sposobów

poprawy spójności odpowiedzi jest inżynieria podpowiedzi, która w połączeniu z dostrajaniem umożliwia uzyskanie bardziej spójnych odpowiedzi. W szczególności inżynieria podpowiedzi obejmuje zaprojektowanie podpowiedzi, a także jej „uzupełnienie" (tj. odpowiedź z modelu), aby model działał w pożądany sposób. Dobrze dobrana konstrukcja podpowiedzi osiąga wyższy wskaźnik predykcji, a także ułatwia przetwarzanie wygenerowanych odpowiedzi. Jakość wyników uzyskanych za pomocą inżynierii podpowiedzi wymaga dużej liczby przykładowych podpowiedzi, które mogą się różnić w zależności od konkretnego zadania przetwarzania języka naturalnego (NLP).

Projektowanie poleceń

Dobrze zaprojektowana inżynieria poleceń umożliwia automatyzację zadań następczych znacznie łatwiej niż w przypadku tradycyjnych metod. Jednak dwa dobrze zaprojektowane polecenia dla tego samego zadania mogą generować zasadniczo różne rezultaty. Z nieformalnego punktu widzenia, dobrze zaprojektowane polecenia przypominają „dobry trening" w sporcie: oba zazwyczaj przynoszą pożądane rezultaty. Innymi słowy, dobrze napisany komunikat dostarcza szczegółowych i istotnych informacji w zwięzły sposób. Chociaż modele GPT zazwyczaj wymagają kilkuset przykładów do wykonania większości zadań następczych, to podejście jest oczywiście lepsze od innych technik, które mogą wymagać tysięcy przykładów szkoleniowych. Kolejnym aspektem, który należy wziąć pod uwagę, są koszty, jakie mogą zostać poniesione w związku z dostrajaniem i wnioskowaniem. Warto zbadać opłacalność płatnych usług, które oferują interfejsy API do wykonywania takich zadań. Co więcej, zadanie oceny dokładności prognoz może być bardziej złożone niż w przypadku tradycyjnych podejść, a szybkość wnioskowania jest zazwyczaj niska w przypadku dużych modeli LLM (takich jak GPT-3 lub GPT-J), ponieważ prognozy muszą być propagowane do ogromnej liczby parametrów. Rozważmy teraz czas potrzebny na wykonanie kilkuset tysięcy prognoz: może to zająć wiele godzin (a nawet dni). Zatem umiejętność formułowania dobrze sformułowanych podpowiedzi może być procesem uczenia się metodą prób i błędów, będącym połączeniem nauki i sztuki.

Podpowiedzi w łańcuchu myśli

Podpowiedzi w łańcuchu myśli (CoT) to kolejna technika, która obejmuje wieloetapową sekwencję odpowiedzi na pytania. Technika ta została opisana w artykule arXiv i polega na wstawianiu tekstu „Pomyślmy krok po kroku" przed każdą odpowiedzią. Twórcy techniki CoT twierdzą, że ich metoda Zero-shot-CoT przewyższa metodę Zero-shot LLM w wielu zadaniach. Co więcej, autorzy artykułu w arXiv zastosowali swoją technikę bez żadnych przykładów z niewielką liczbą prób na modelach LLM, takich jak model InstructGPT o parametrach 175 B i model PaLM o parametrach 540 B. Autorzy sugerują również podjęcie dodatkowych wysiłków w celu wydobycia wiedzy z modelu LLM o parametrach z zerową liczbą prób przed wykonaniem zadań precyzyjnego dostrajania.

Samospójność i CoT

Samospójność to technika samonadzorowana, która wykorzystuje podpowiedzi oraz technikę dekodowania, która zastępuje technikę zachłanną w CoT. Dokładniej, dokonuje wyboru z zestawu wygenerowanych próbek, zamiast polegać wyłącznie na podejściu zachłannym. Więcej informacji na temat samospójności można znaleźć w Internecie: https://arxiv.org/abs/2203.11171. Według autorów poprzedniego artykułu, ich „zasadą praktyczną" jest wygenerowanie listy około dwudziestu zdań, a następnie wybranie z niej najlepszego zdania. Co więcej, samospójność osiąga lepszą wydajność niż ogólne podpowiedzi CoT w modelach LLM, takich jak GPT-3, LaMDA i PaLM. (Więcej szczegółów w tabeli 2 w artykule arXiv).

Samospójność, CoT i nienadzorowane zbiory danych (LMSI)

Maszyny LLM osiągnęły imponujące wyniki dzięki wykorzystaniu wstępnie wytrenowanych modeli i precyzyjnego dostrajania, a także dużych zbiorów danych. Jednak dodatkowa poprawa wydajności wymaga zastosowania wysokiej jakości nadzorowanych zbiorów danych. Jedno z podejść obejmuje samodoskonalenie modelu językowego (LMSI), które ulepsza modele LLM przy użyciu danych nienadzorowanych. Technika ta wykorzystuje CoT z niewielką liczbą prób dla monitów, a także schemat głosowania większościowego do wyboru predykcji. Więcej informacji na temat samospójności, COT i nienadzorowanych zbiorów danych można znaleźć online: https://arxiv.org/pdf/2210.11610.pdf.

SZCZEGÓŁY DOTYCZĄCE INŻYNIERII PROMPT

Inżynieria Prompt jest kluczowym aspektem pracy z modelami LLM, do których należą oczywiście GPT-3 i GPT-4. W miarę zdobywania doświadczenia w pracy z modelami LLM, społeczność stale opracowuje najlepsze praktyki i strategie, aby uzyskać jak najlepsze rezultaty z tych modeli. Inżynieria komend odnosi się do sztuki i nauki tworzenia komend wejściowych, które ukierunkowują wyniki modelu językowego w pożądanym kierunku. Biorąc pod uwagę rozległą wiedzę i możliwości modeli takich jak GPT-3, sposób formułowania pytań lub komend może znacząco wpłynąć na jakość, dokładność i trafność odpowiedzi.

Znaczenie inżynierii natychmiastowej

Poniższa lista zawiera kilka powodów dotyczących znaczenia inżynierii natychmiastowej, a następnie krótki opis każdego z nich:

• Wskazówki dotyczące modelu

• Jakość wyników

• Specyfikacja zadania

Wskazówki dotyczące modelu: LLM nie mają określonego celu ani intencji. Generują odpowiedzi w oparciu o wzorce wyuczone z danych. Dobrze skonstruowany monit kieruje modelem w kierunku generowania bardziej dokładnych i trafnych odpowiedzi.

Jakość wyników: Różne monity mogą dawać odpowiedzi o różnej jakości. Udoskonalony monit może wywołać bardziej szczegółową lub zniuansowaną odpowiedź.

Specyfikacja zadania: W przypadku konkretnych zadań, takich jak generowanie kodu lub rozwiązywanie problemów matematycznych, sposób strukturyzowania monitu może mieć kluczowe znaczenie dla uzyskania pożądanego wyniku.

Rodzaje podpowiedzi

Poniższa lista zawiera kilka powodów dotyczących znaczenia inżynierii podpowiedzi, a następnie krótki opis każdego rodzaju podpowiedzi:

• podpowiedzi otwarte

• podpowiedzi zamknięte

• podpowiedzi instruktażowe

• podpowiedzi z przewodnikiem

• podpowiedzi iteracyjne

• tokeny temperatury i maks.

Podpowiedzi otwarte: Są ogólne i szerokie, pozwalając modelowi określić kierunek odpowiedzi. Przykładem tego typu podpowiedzi jest „Opowiedz mi o Układzie Słonecznym”. Podpowiedzi zamknięte: Szukają konkretnych odpowiedzi lub faktów, na przykład „Kto był pierwszym prezydentem Stanów Zjednoczonych?”. Podpowiedzi instruktażowe: Nakłaniają model do wykonania określonego zadania lub wygenerowania treści w określony sposób. Przykładem jest „Przetłumacz poniższy tekst z języka angielskiego na francuski:…”. Podpowiedzi z przewodnikiem: Dostarczają dodatkowego kontekstu lub wskazówek, które sterują odpowiedzią modelu. Przykładem tego podpowiedzi jest „Wyjaśnij mechanikę kwantową tak, jakbym był 10-latkiem”. Iteracyjne monity: Polegają na serii rozmów z modelem, udoskonalających zapytanie na podstawie jego poprzedniej odpowiedzi. Przypominają one rozmowę, w której stopniowo dopracowuje się pożądaną odpowiedź.

Temperatura i maksymalna liczba tokenów: Chociaż nie są to monity w tradycyjnym sensie, dostosowanie parametrów, takich jak „temperatura” (kontrolująca losowość wyników modelu) i „maksymalna liczba tokenów” (ograniczająca długość odpowiedzi), może również wpływać na odpowiedzi modelu.

Wytyczne dotyczące skutecznych monitów

W przypadku ChatGPT i podobnych modeli skuteczność monitów często zależy od kontekstu oraz konkretnych informacji lub oczekiwanej odpowiedzi. Istnieją jednak pewne rodzaje monitów, które są skuteczniejsze w uzyskiwaniu szczegółowych i zniuansowanych odpowiedzi z modelu. W przypadku ChatGPT i podobnych modeli należy jasno określić, jakiego rodzaju odpowiedzi się oczekuje. Monity z instrukcjami i wskazówkami pomagają w zapewnieniu tej jasności. Warto jednak również eksperymentować i iterować swoje monity, ponieważ czasami nawet nieznacznie inne sformułowanie monitu może prowadzić do znacząco różnych wyników. Poniższa lista zawiera listę wytycznych dotyczących skutecznego tworzenia monitów, a następnie krótki opis każdego z nich:

- Bądź precyzyjny.
- Eksperymentuj.
- Wykorzystuj parametry.
- Przeanalizuj i iteruj.

Bądź precyzyjny: Jasno określ, czego szukasz. Jeśli chcesz krótkiego podsumowania, określ dokładnie, czego oczekujesz. Podaj kontekst: Podanie krótkiego tła może pomóc w ukierunkowaniu odpowiedzi modelu.

Eksperyment: Często potrzeba kilku prób, aby dopracować polecenie i uzyskać pożądany wynik. Nie wahaj się przeformułować lub podać dodatkowe instrukcje.

Wykorzystaj parametry: Jeśli używasz modelu takiego jak GPT-3 bezpośrednio za pośrednictwem API, eksperymentuj z parametrami, takimi jak temperatura, aby uzyskać różne style odpowiedzi.

Przejrzyj i udoskonalaj: Stale udoskonalaj swoje polecenia w oparciu o otrzymywane wyniki. To iteracyjne podejście może z czasem prowadzić do bardziej efektywnych poleceń.

Przykłady skutecznych poleceń dla ChatGPT

Polecenia z instrukcjami i wskazówkami są często najskuteczniejsze w przypadku ChatGPT. Te polecenia dostarczają kontekstu lub wskazówek, albo instruują model, aby odpowiedział w określony sposób. Pomagają one zawęzić szeroki zakres potencjalnych odpowiedzi modelu do czegoś bardziej

szczegółowego i zgodnego z intencją użytkownika. Poniżej przedstawiono kilka przykładów ulepszonych poleceń.

Instrukcja jednoznaczna: Zamiast „II wojna światowa” użyj „Podaj krótkie podsumowanie II wojny światowej”.

Podanie kontekstu: Zamiast „Wyjaśnij teorię względności”, użyj „Wyjaśnij teorię względności Einsteina w prostych słowach komuś bez wykształcenia fizycznego”.

Prośba o konkretny format: Zamiast „Pętle w Pythonie”, użyj „Pokaż mi przykład pętli for w Pythonie”.

Wskazywanie głębi lub poziomu szczegółowości: Zamiast „Opowiedz mi o czarnych dziurach”, użyj „Przedstaw mi szczegółowy przegląd aktualnego stanu wiedzy naukowej na temat czarnych dziur”.

Stworzenie scenariusza: Zamiast „Jak założyć firmę?”, użyj „Wyobraź sobie, że jestem świeżo upieczonym absolwentem studiów wyższych z doświadczeniem w inżynierii oprogramowania. Jak powinienem założyć startup technologiczny?”

Informacje porównawcze lub kontrastujące: Zamiast „Czym jest socjalizm?”, użyj „Porównaj i zestaw socjalizm z kapitalizmem pod względem zasad ekonomicznych”.

Prośba o kroki lub proces: Zamiast „Pieczenie ciasta”, użyj „Wymień krok po kroku proces pieczenia ciasta czekoladowego”. Sterowanie tonem lub stylem: Zamiast „Opowiedz mi historię”, użyj „Opowiedz mi krótką, humorystyczną historię o kocie i psie”.

UCZENIE SIĘ NA KILKU SZTUKACH I DOSTRAJANIE STYLÓW LLMS

Uczenie się na kilku próbach to sposób na zapewnienie dodatkowego szkolenia dla LLM bez aktualizacji parametrów LLM. Istnieje jednak limit rozmiaru (tj. danych) związany z uczeniem się na kilku próbach. Dostrajanie pozwala jednak na trenowanie modelu na znacznie większym zbiorze danych niż w przypadku uczenia się na kilku próbach. W rezultacie można osiągnąć lepsze wyniki w przypadku większej różnorodności zadań dla LLM. GPT-3 obsługuje uczenie się na kilku próbach, dostrajanie i uczenie się oparte na podpowiedziach, które zostaną omówione w kolejnych podsekcjach. Dowiesz się o trzech głównych typach podpowiedzi, a także o kompromisach między uczeniem się na kilku próbach a dostrajaniem oraz między dostrajaniem a podpowiedziami. Dodatkowo, jedna z sekcji zawiera sugestie dotyczące wyboru modelu GPT-3 dla Twoich zadań. Chociaż bardzo duże modele LLM zazwyczaj dobrze reagują na uczenie się metodą „mniej prób”, mniejsze modele LLM niekoniecznie poprawiają się dzięki uczeniu się metodą „mniej prób”, nawet po uwzględnieniu wielu przykładów. Ponadto uwzględnienie przykładów zmniejsza część okna kontekstowego dostępną na uwzględnienie innych istotnych informacji. W poprzednim scenariuszu dostrajanie modelu LLM może być realną alternatywą, czyli techniką uczenia nadzorowanego, która obejmuje znacznie mniejszy, ale wysoce uporządkowany zbiór danych. Elementy zbioru danych są wykorzystywane do aktualizacji wag w modelu LLM. Należy zauważyć, że każdy element składa się z monitu i jego uzupełnienia, co różni się od nienadzorowanego etapu wstępnego treningu modeli LLM. Ponieważ dostrajanie obejmuje aktualizację wag w modelu LLM, technika ta różni się od uczenia się metodą „zero-shot”, „one-shot” i „mniej prób”, które nie aktualizują wag modelu LLM. Pełne dostrajanie aktualizuje wszystkie parametry modelu LLM, co oznacza, że tworzona jest nowa wersja wstępnie wytrenowanego modelu. Jednakże wymagania dotyczące pamięci do przechowywania wszystkich parametrów wstępnie wytrenowanego LLM mogą być znacznie większe.Wstępne trenowanie polega na trenowaniu modelu LLM z wykorzystaniem ogromnej ilości niestrukturyzowanych danych tekstowych za pomocą samonadzorowanego uczenia się. Natomiast dostrajanie to proces uczenia nadzorowanego, obejmujący znacznie mniejszyzbiór danych składający się z oznaczonych przykładów, które służą do

aktualizacji wag modelu LLM. Co więcej, oznaczone przykłady to pary szybkich uzupełnień, a proces dostrajania rozszerza proces trenowania modelu, poprawiając jego zdolność do generowania prawidłowych uzupełnień dla określonego zadania.

Uczenie się metodą „małej liczby prób"

GPT-3 obsługuje uczenie się metodą „małej liczby prób", co umożliwia dostarczanie podpowiedzi z poprawnymi przykładami w celu poprawy odpowiedzi modelu. W rezultacie nie ma potrzeby aktualizacji parametrów w GPT-3, ponieważ próbki do uczenia się metodą „małej liczby prób" pełnią funkcję treningu dla modelu. W rzeczywistości GPT-3 pokazał, jak uczenie się metodą „małej liczby prób" osiąga znaczące rezultaty nawet bez aktualizacji parametrów lub danych specyficznych dla zadania. GPT-3 jako pierwszy pokazał, że modele LLM mogą być wykorzystywane do uczenia się metodą „małej liczby prób" i mogą osiągać imponujące rezultaty bez konieczności gromadzenia danych specyficznych dla zadania na dużą skalę lub aktualizacji parametrów modelu. Użytkownicy mogą przesyłać swoje niestandardowe zestawy danych treningowych do dostrojonych wersji GPT-3, które są również hostowane w OpenAI i dostępne za pośrednictwem jego interfejsów API. W zależności od rozmiaru niestandardowych zestawów danych, dostrajanie GPT-3 może trwać od kilku minut do kilku godzin. Jednak dostrajanie większych modeli wymaga mniejszej ilości danych niż w przypadku mniejszych modeli.

Uczenie się metodą „mniej niż jeden strzał" i podpowiedzi

Jednym z interesujących aspektów GPT-3 jest uczenie się metodą „mniej niż jeden strzał" poprzez podpowiedzi, czyli technika dostarczająca instrukcje w postaci ciągów wejściowych. Takie podpowiedzi dostarczają GPT-3 przykładów oczekiwanego typu odpowiedzi, które GPT-3 może skutecznie wykorzystać do zapewnienia odpowiedzi zgodnych z informacjami zawartymi w podpowiedziach. Przeczytaj sekcję „Moc podpowiedzi" w rozdziale 8, aby zapoznać się z wyraźnym kontrastem między typem wyników generowanych przez GPT-3 na podstawie podpowiedzi pozytywnej i negatywnej. GPT-3 wykorzystuje również „prawdopodobieństwo warunkowe", które działa w następujący sposób:

biorąc pod uwagę zbiór słów, oblicza prawdopodobieństwo, że dane słowo będzie następnym słowem w tym zbiorze.

W rezultacie GPT-3 może przyjąć zbiór słów jako dane wejściowe i dokładnie przewidzieć następne słowo. W rzeczywistości GPT-3 potrafi znacznie więcej: biorąc pod uwagę ciąg wejściowy, GPT-3 może generować całe artykuły, które mają sens (tj. nie są ciągami bezsensownych słów). Niestety, niektóre stwierdzenia GPT-3 mogą być nieprawdziwe.

Dostrajanie za pomocą dostrajania symboli

Nowoczesną (w momencie druku tej książki) techniką dostrajania jest dostrajanie symboli, opracowana przez Google. Dostrajanie symboli koncentruje się na mapowaniu danych wejściowych na etykiety, co pozwala na lepsze dostrajanie.

LoRA, kwantyzacja i QLoRA

W tej sekcji krótko omówiono LoRA i QLoRA, które są przydatne do dostrajania LLM. Adaptery niskiego rzędu (LoRA) to niewielkie zestawy parametrów, które można trenować. Podczas procesu dostrajania architektury transformatora, te niewielkie zestawy parametrów są wstrzykiwane do każdej warstwy transformatora. Ponadto, te niewielkie zestawy wag są aktualizowane podczas procesu dostrajania, podczas gdy oryginalne wagi modelu pozostają zamrożone. Główną zaletą tej techniki jest znaczne zmniejszenie liczby parametrów możliwych do trenowania dla zadań downstream. Co więcej, Hugging

Face stworzył bibliotekę Parameter Efficient Fine-Tuning (PEFT), która obsługuje LoRA. Kwantyzacja polega na przechowywaniu danych w typach danych o mniejszej precyzji. Na przykład, można użyć 8-bitowego typu danych (tj. bajtu) do przechowywania danych 16-bitowych lub 32-bitowych. Oczywiście, ta redukcja liczby bitów powoduje utratę precyzji. Co ciekawe, kwantyzacja prowadzi do modelu, którego zmniejszony rozmiar może przeważyć nad utratą precyzji. Trzecim tematem tej sekcji jest kwantyzowana adaptacja niskiego rzędu (QLoRA). QLoRA łączy kwantyzację i adaptację niskiego rzędu, aby osiągnąć efektywne dostrajanie modeli sztucznej inteligencji. Zaletą QLoRA jest redukcja pamięci podczas etapu dostrajania modelu LLM.

PODSUMOWANIE

Ta sekcja rozpoczęła się od omówienia niektórych funkcji GPT-2 i GPT-3, a także kilku przykładów kodu w Pythonie. Następnie poznano trzy główne typy znanych modeli LLM (modeli dużego języka): modele tylko kodera, modele tylko dekodera i modele koder-dekoder. Następnie zapoznano się z rodziną modeli językowych opartych na GPT z OpenAI, a następnie z ogólnym wprowadzeniem do GPT-3, jego celów i kilku modeli opartych na GPT-3. Ponadto przedstawiono porównanie GPT-3 i BERT oraz mocne i słabe strony GPT-3. Ponadto zapoznano się z platformą GPT-3 Playground, gdzie można przeprowadzać wnioskowanie z wykorzystaniem GPT-3. Na koniec dowiedziałeś się, jak tworzyć monity w GPT-3.