Dawid Woźniak
Informatyka - Uczenie Maszynowe I Sztuczna Inteligencja

Uczenie ze Wzmocnieniem

Językowe Przedszkole

1. Opis ćwiczenia

Celem laboratorium było wykorzystanie metody GRPO do zmiany zachowania wytrenowanego już dużego modelu językowego. Jako model został użyty gpt-2. Podstawowym zadaniem było zmuszenie modelu do używania głównie 4 literowych słów. Alternatywny cel ćwiczenia to zmuszenie modelu do używania wypowiedzi zawierających jak najwięcej słów.

2. Wyniki

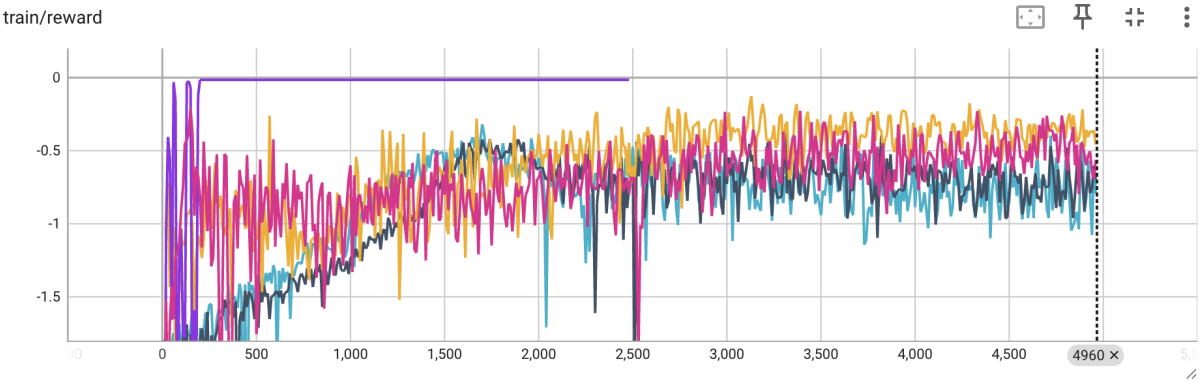
2.1 Słowa 4 literowe

```python
def reward_len(completions, **kwargs):
    if type(completions) == list:
        lst = []

        for completion in completions:
            reward = 0
            splitted = completion.split(" ")
            for phrase in splitted:
                reward -= abs(4 - len(phrase))

            lst.append(reward / len(splitted))
        return lst
    elif type(completions) == str:
        reward = 0
        splitted = completions.split(" ")
        for phrase in splitted:
            reward -= abs(4 - len(phrase))
        return reward / len(splitted)
```

Rys. 1. Funkcja nagrody dla słów 4 literowych.

Funkcja nagrody dla każdego słowa liczyła $abs(4 - len(phrase))$ co miało premiować słowa 4 literowe. Dodatkowo, skumulowana nagroda została podzielona przez ilość słów w wypowiedzi. Bez tej normalizacji, model uczył się odpowiadać krótszymi wypowiedziami. Testowane były następujące konfiguracje:

default - defaultowe parametry dla grpo trainer config
config_1 - $max\_completion\_lenght = 512$
config_2 - $max\_completion\_lentgh = 512, epsilon = 0.5, learning\_rate = 1e - 5$
config_3 -
$max\_completion\_length = 256, epsilon = 0.7, learning\_rate = 1e - 4, beta = 0.1,$
$gradient\_accumulation\_steps = 2, num\_generations = 16$
config_4 -
$max\_completion\_length = 256, epsilon = 1, learning\_rate = 1e - 3, beta = 0.2,$
$gradient\_accumulation\_steps = 4, num\_generations = 16$



Run ↑	Value	Step	Relative
config_1	-0.7933	4,960	6.052 hr
config_2	-0.5622	4,960	6.299 hr
config_3	-0.4584	4,960	3.513 hr
config_4	-0.0156	2,480	2.078 hr
default	-0.6955	4,960	3.451 hr

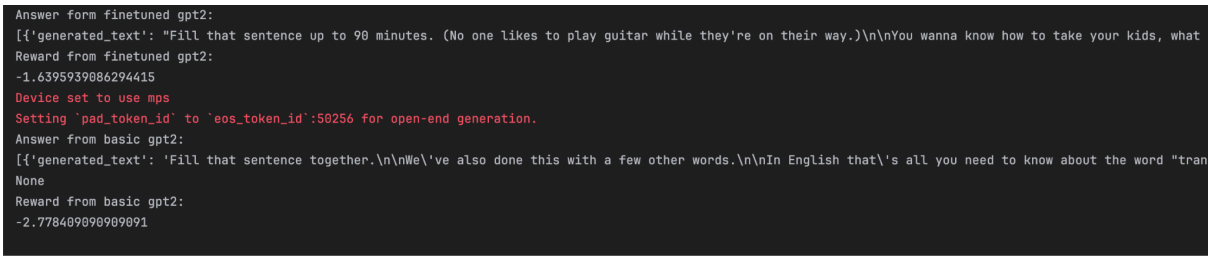
Rys. 2. Przebieg funkcji nagrody w czasie uczenia.

Najlepsze wyniki zostały uzyskane dla config_3. Dla config_4, nagroda od pewnego momentu wynosiła ciągle zero, co początkowo może wydawać się pożądanym efektem, jednak po inferencji modelu widać, że model nauczył się używać tylko jednego 4 literowego słowa.

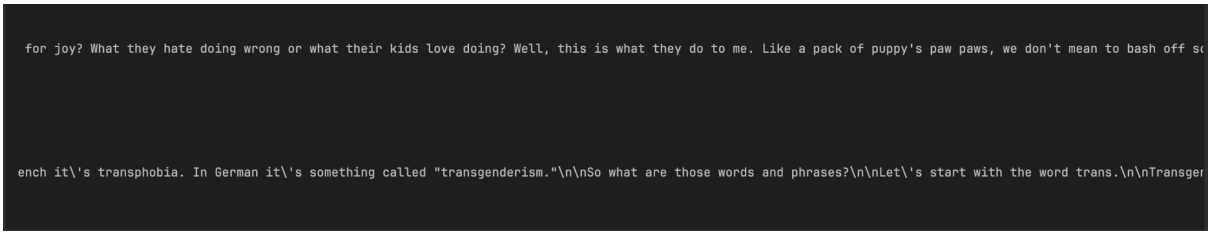
```
Answer form finetuned gpt2:
[{'generated_text': "Tomorrow I'm gonna learn for exam last last last last last last last last last last l
ast last last last last last last last last last last last last last last last last last last last last last la
st last last last last last last last last last last last last last last"}]
```

Rys. 3. Przykład wygenerowanego tekstu dla przetrowanego modelu.

Wydaje się, że wpływ na to mogą mieć parametry $gradient\_accumulation\_steps = 4$ $learning\_rate = 1e - 3$, $epsilon = 1$. Akumulowanie gradientu z takiej ilości kroków i używanie większej stałej uczącej oraz zwiększenie epsilonu może prowadzić do dużych aktualizacji wag w poszczególnych krokach i doprowadzić do przetrenowania.



Rys. 4. Przykład wygenerowanego tekstu dla poprawnego modelu..



Rys. 5. Przykład wygenerowanego tekstu dla poprawnego modelu.

Dla modelu uczonego z parametrami config_3, model odpowiada słowami bardziej zbliżonymi do słów 4 literowych oraz zachowuje sens wypowiedzi.

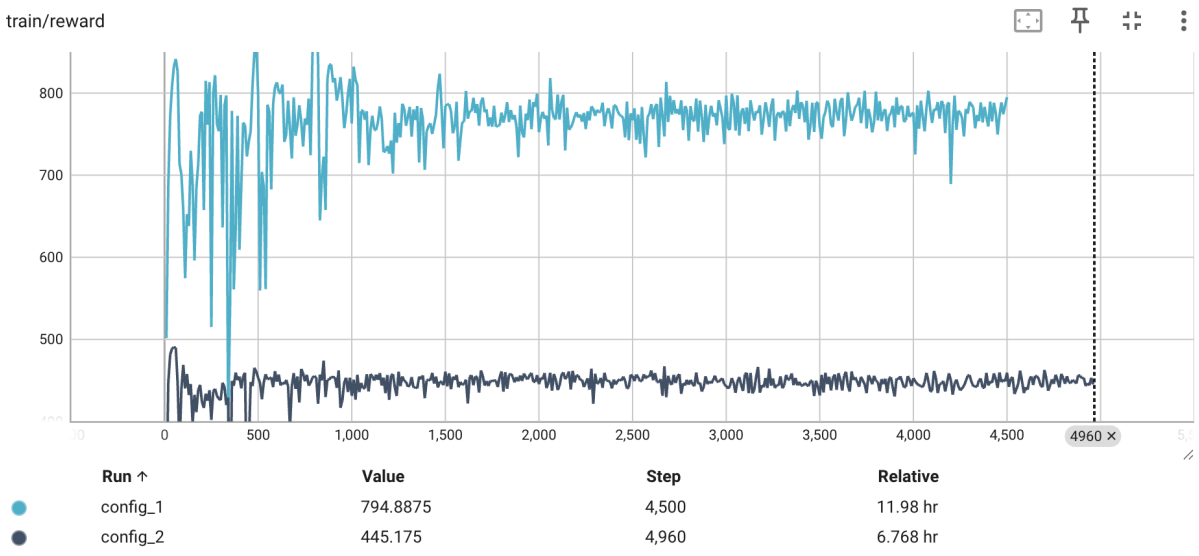
2.2 Wypowiedzi z dużą ilością słów

```python
def alternative_reward(completions, **kwargs):
    """
    Bonusing long completions
    """
    if type(completions) == list:
        lst = []

        for completion in completions:
            splitted = completion.split(" ")
            lst.append(len(splitted))

        return lst
    elif type(completions) == str:

        splitted = completions.split(" ")
        return len(splitted)
```

Rys. 6. Alternatywna funkcja nagrody.

W tej konfiguracji, funkcja nagrody zwracała ilość słów w odpowiedzi modelu, co miało spowodować używanie przez model dłuższych odpowiedzi.

Wzięto parametry z config_3 w poprzednim podpunkcie i przetestowano 2 wersje, $max\_completion\_length = 512$ oraz $max\_completion\_length = 900$



Rys. 7. Przebieg funkcji nagrody w czasie uczenia.

Jak widać na wykresie, dla obu konfiguracji model zaczyna otrzymywać maksymalną nagrodę na jaką pozwala maksymalna dozwolona ilość generowanych tokenów. W celu przetestowaniu wyników wygenerowałem 1000 odpowiedzi dla podstawowego i trenowanego modelu i obliczyłem wartości funkcji nagród. Statystyki zawiera poniższa tabela:

	Mediana	Średnia	Odchylenie Standardowe
gpt2-finetuned	40	38.96	5.75
gpt2	38	36.57	6.33

Tabela 1. Ewaluacja modelu dla alternatywnej funkcji nagrody.

Porównując wyniki, wydaje się że trening zadziałał i odpowiedzi modelu zawierają trochę więcej słów niż odpowiedzi podstawowego modelu gpt2.

3. Wnioski

Udało się spełnić cel laboratorium a więc wykorzystać grpo do finetuningu modelu gpt2 i zmuszenie go do używania słów 4 literowych. Alternatywny cel ćwiczenia w postaci używania wypowiedzi zawierających jak największą ilość słów również został spełniony. Samo laboratorium, pomogło mi zgłębić wiedzę na temat zastosowania metod uczenia ze wzmocnieniem w trenowaniu dużych modeli językowych. Metoda GRPO znacznie zmniejsza koszt obliczeniowy etapu RLHF w uczeniu modelu względem PPO, gdyż nie używa modelu

wartościującego, czyli krytyka, a często ten model ma porównywalną wielkość do naszego głównego modelu czyli polityki. Można by też było zastosować jakąś metodę PEFT, np. LORA co jeszcze bardziej powinno zmniejszyć koszt obliczeniowy treningu.