Trenowanie dużych modeli językowych

Źródło: https://airrival.pl/trenowanie-duzych-modeli-jezykowych/

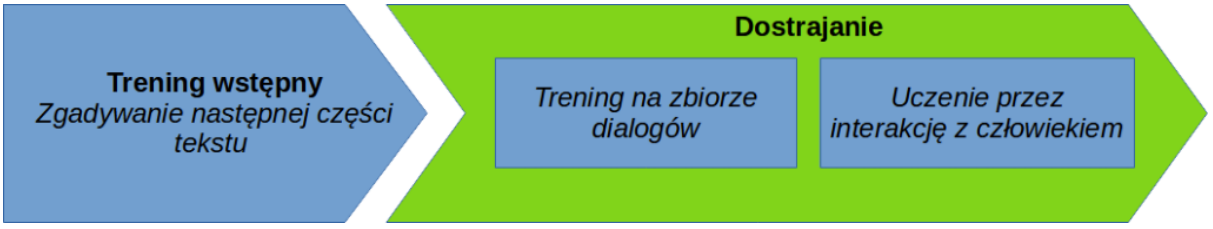
Rozpoczynamy cykl artykułów pod tytułem **Anatomia LLM**. Przyjrzymy się w nim dokładniej robiącym furorę sieciom neuronowym, znanym powszechnie jako **duże modele językowe,** czyli **Large Language Models**. Na początku naszkicujemy sobie, jak ogólnie przebiega trening modelu tej klasy oraz jakie dane są w nim wykorzystywane.

Etapy trenowania LLM

Szkolenie dużego modelu językowego zasadniczo możemy podzielić na dwa etapy:

1. Trening wstępny (ang. *pre-training*)
2. Dostrajanie (ang. *fine-tuning*)

W pierwszym etapie nasz model uczy się ogólnych zasad rządzących językiem, później natomiast jest ćwiczony w sensownym odpowiadaniu na pytania. To trochę tak jak z procesem nauki przez człowieka – żeby móc pisać powieści, najpierw musimy w ogóle nauczyć się mówić.



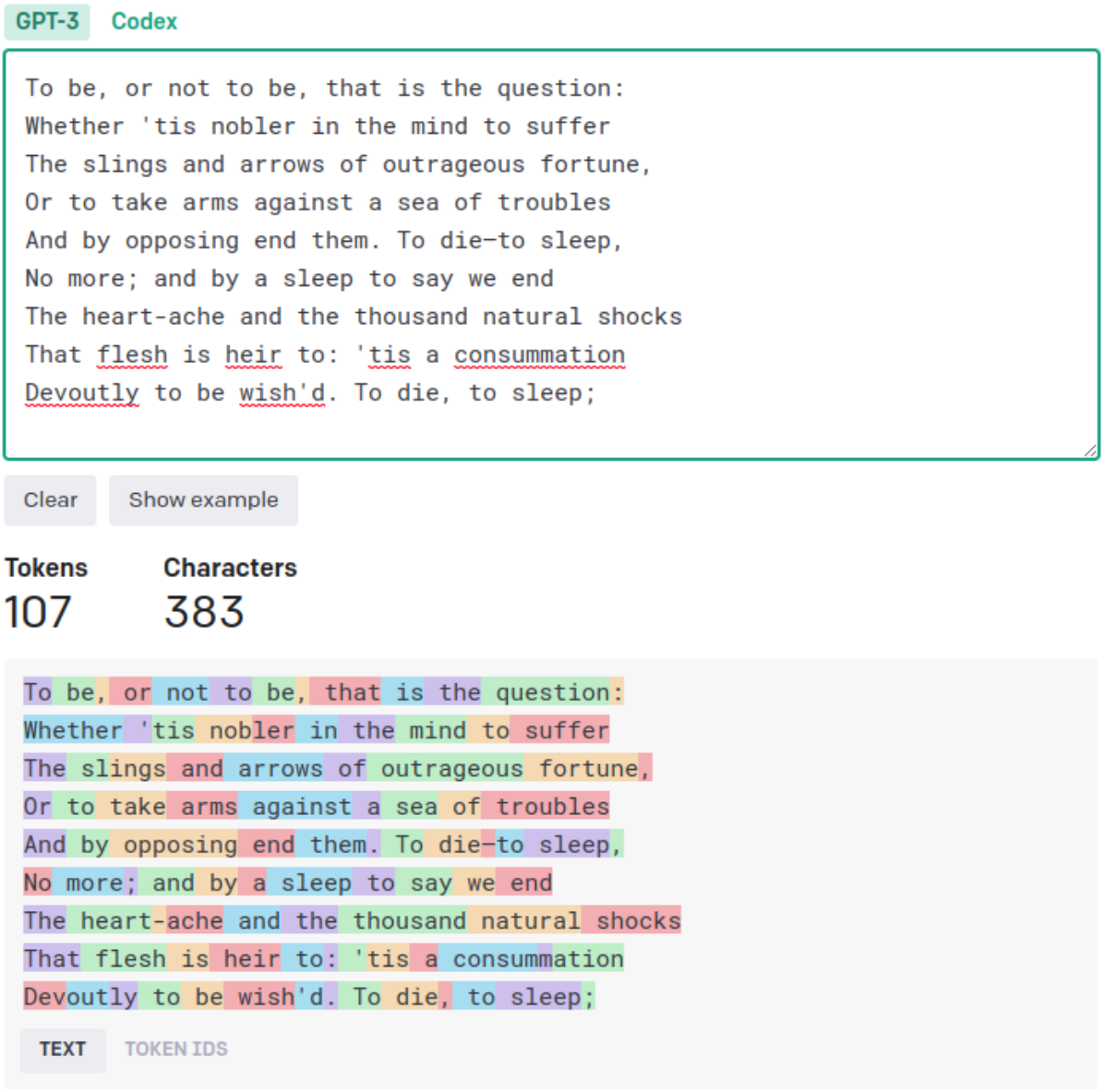
Obrazek zainspirowany schematem opublikowanym na Linkedinie przez Agnieszkę Mikołajczyk z Voicelab.ai.

Trening wstępny

W jego wyniku otrzymujemy tzw. model bazowy (ang. *base LLM*), którego zadaniem jest tylko i wyłącznie zgadywanie kolejnych części tekstu na podstawie poprzedzającego go fragmentu. Celowo nie piszę tutaj „kolejnych zdań” lub „kolejnych liter”, gdyż tekst w świecie modeli językowych dzielony jest na jednostki zwane **tokenami**.

Tokeny

Czym dokładnie jest token? Rzućmy okiem na to, jak działa tokenizator używany konkretnie w modelu ChatGPT (zachęcam, aby odwiedzić tę stronę i samemu trochę się nim pobawić). Na początek używam przykładu w języku angielskim, gdyż właśnie pod kątem mowy Szekspira zostało opracowane to akurat narzędzie do dzielenia tekstu na kawałki.



Jak widzimy, w przypadku języka angielskiego tokenem często jest całe słowe lub morfem.

Zerknijmy, w jaki sposób ChatGPT próbuje pokroić tekst w języku polskim używając swojego anglocentrycznego tokenizatora. Wychodzi mu to bardzo nieporadnie, jednak nie przeszkadza to modelowi aż tak bardzo, jak moglibyśmy się tego spodziewać.

Przykład użycia tokenizatora stworzonego pod kątem języka angielskiego do podziału na tokeny tekstu w języku polskim

Wejście vs wyjście

Ile następnych tokenów próbuje zgadnąć typowy duży model językowy? O dziwo, zazwyczaj jest to tylko jeden token. Długość tekstu wejściowego (określanego jako *kontekst*) jest natomiast o wiele większa i wynosi, przykładowo:

- GPT-3.5: 4096 tokenów
- Falcon: 2048 tokenów
- LLama 2: 4096 tokenów

Gdybyśmy dla uproszczenia przyjęli, że chcemy wytrenować mały model przyjmujący kontekst o długości dziesięciu tokenów, przewidujący standardowo jeden token naprzód, nasze przykłady treningowe mogłyby wyglądać mniej więcej tak, jak na poniższym obrazku.

Wejście (kontekst)										Wyjście
To	be	or	not	to	be	,	that	is	the	question
Dev	out	ly	to	be	wish	‘d	.	To	die	,
W	l	az	ł	k	ot	ek	na	p	ł	ot

Wyjście to pożądany wynik działania modelu dla danego kontekstu. W pierwszym przykładzie, model uzyska dobry wynik, jeśli za najbardziej prawdopodobne uzna wystąpienie tokenu „question”.

Jeżeli najczęściej prognozujemy tylko jeden token naprzód, to w jaki sposób ChatGPT i pokrewne mu rozwiązania są w stanie generować całe elaboraty? Po prostu, po zgadnięciu pierwszego tokenu nie kończymy generowania. Doklejamy go do naszego pierwotnego pytania i ponownie wrzucamy do modelu. I tak wiele razy, dopóki model nie zwróci specjalnego **tokenu końca sekwencji**, *EOS* (ang. *end-of-sequence* lub *end-of-sentence*) będącego symbolem końca tekstu. Z innych wartych wspomnienia tokenów mamy też **token początku sekwencji** (ang. *start-of-sequence* lub *beginning-of-sequence*) oraz **token-wypełniacz** (ang. *paddding token*). Tego ostatniego używamy, kiedy tekst wejściowy do modelu jest krótszy niż maksymalna liczba tokenów wejściowych. Podczas treningu model nauczy się go igorować, jednak tak czy inaczej musimy użyć wejścia pełnej długości.

Zanim przejdziemy dalej, wyjaśnijmy sobie kilka pojęć. Zdarzyło mi się tutaj parę razy nazwać wejście do modelu pytaniem, lecz w powszechnie używanej nomenklaturze występuje on zwykle jako **podpowiedź** (ang. *prompt*). Stąd też pochodzi nazwa nowopowstałej profesji, tj. *prompt enginneringu*. Ten ostatni termin przetłumaczyłbym jako *inżyniera podpowiedzi* lub *inżynieria odpytywania*. Tekst wygenerowany przez model często odnajdujemy z kolei pod nazwą **uzupełnienia** (*completion*).

Dane do treningu wstępnego

Trening wstępny odbywa się przy wykorzystaniu wielkich zwałów tekstu pozyskanych z Internetu. Popularnym wyborem jest CommonCrawl, czyli ogromny zbiór treści pisanej pobranej z 3.1 miliarda stron internetowych w różnych językach. Jego ostatnia wersja (maj/czerwiec 2023) waży ok. 390 terabajtów. Trening wstępny zużywa pokaźne ilości zasobów, ale za to jest dość łatwy z technicznego punktu widzenia. Dane treningowe nie muszę być ręcznie etykietowane przez ludzi w żaden dodatkowy sposób. Po prostu automatycznie serwujemy modelowi kolejne fragmenty tekstu.

Training Data

Falcon-7B was trained on 1,500B tokens of RefinedWeb, a high-quality filtered and deduplicated web dataset which we enhanced with curated corpora. Significant components from our curated copora were inspired by The Pile (Gao et al., 2020).

Data source	Fraction	Tokens	Sources
RefinedWeb-English	79%	1,185B	massive web crawl
Books	7%	110B	
Conversations	6%	85B	Reddit, StackOverflow, HackerNews
Code	3%	45B	
RefinedWeb-French	3%	45B	massive web crawl
Technical	2%	30B	arXiv, PubMed, USPTO, etc.

Tabelka prezentująca dane użyte do treningu modelu Falcon-7B. Oprócz wielkiej kolekcji stron internetowych (RefinedWeb) mamy tutaj również książki, atykuły naukowe czy rozmowy na forach wyodrębnione jako osobna kategoria tekstów. Wielkość tych zbiorów mierzona jest oczywiście w tokenach.

Po wykonaniu takiego treningu, model jest w stanie w sensowny sposób uzupełniać tekst. Nieraz będzie potrafił nawet udzielić logicznej odpowiedzi na pytanie. Wynika to jednak po prostu z faktu, że w użytych zbiorach danych tekstowych siłą rzeczy muszą pojawić się jakieś odpowiedzi

poprzedzone stosownymi pytaniami. Model po wstępnym szkoleniu zacznie już więc „kojarzyć fakty”. Nie zawsze jednak tak będzie.

Aby na własne oczy przekonac się o różnicach, porównajmy, jak będą wyglądać odpowiedzi modelu **Falcon-7B** w wersji bazowej i dostrojonej na pytanie ***Who won the last FIFA World Cup?***

Jak widzimy, otrzymaliśmy tekst, który mógłby znaleźć się w jakimś artykule, ale nie jest to konkretna odpowiedź na nasze pytanie. Dodatkowo zauważamy pewien błąd modelu, który na końcu – zanim wygerenerował token końca sekwencji – zdążył jeszcze bezsensownie zwrócić pojedynczy wyraz „Who”. Model dostrojony zadziałał zdecydowanie lepiej. Pomijając fakt, że ma on nieaktualne wiadomości, udzielił nam poprawnej odpowiedzi i to w sposób, którego od niego oczekiwaliśmy. Rzecz jasna, aby uzyskać ten lepszy model, musimy przeprowadzić **dostrajanie** modelu bazowego.

Dostrajanie

Techniki służące do przeobrażania względnie „mało rozgarniętego” modelu bazowego w wyrafinowane chat boty najłatwiej podzielić na dwie grupy, tak jak to przedstawiłem na grafice na początku tego artykułu.

Trening na zbiorze dialogów

Zastanwiałem się, jak przełożyć na polski nazwę tego rodzaju treningu: *instruction fine-tuning*, Zdecydowałem się na określić go mianem **treningu na zbiorze** (krótkich!) **dialogów**, gdyż o to w istocie w nim chodzi. Są to starannie dobrane przykłady, składające się z par *podpowiedź-uzupełnienie*. Często bedą to pary pytanie-odpowiedź, ale wśród zestawów zadań dla modelu znajdziemy również chociażby generowanie pytań dopasowanych do odpowiedzi. Poniżej przedstawiam przykłady z zestawu treningowego Natural Instructions od AllenAI.

Wejście	Wyjście	Rodzaj zadania
Sentence: Jack played basketball after school, after which he was very tired.Question: What did Jack do after the game?	He rested.	Generowanie odpowiedzi na pytanie
Sentence: He was born in China, so he went to the Embassy to apply for a U.S. Visa.	How long did it take for him to get the Visa?	Generowanie pytania do podanej odpowiedzi
Background Paragraph: Passive transport occurs when a substance passes through the cell membrane without needing any energy to pass through. This happens when a substance moves from an area where it is more concentrated to an area where it is less concentrated. Concentration is the number of particles of a substance in a given volume. Let's say you dissolve a teaspoon of salt in a cup of water. Then you dissolve two teaspoons of salt in another cup of water. The second solution will have a higher concentration of salt.	A man put two cups, cup A and cup B, filled with equal amounts of water on to a table and walked away to go check his mail. His son came along and saw the two cups and decided to put some sugar in them to make a tasty drink. The child poured two spoonfuls of sugar into cup A and three spoonfuls of sugar into cup B.	Generowanie historyjek

Możecie się zastanawiać, dlaczego poprzednio jako wyjście pokazałem jeden token, a teraz znów mamy tutaj całe zdania, a nawet dłuższe wypowiedzi. W rzeczywistości, podczas treningu model będzie zgadywać po jednym tokenie. Wyjście w tym kontekście oznacza pożdany tekst, który model powinien wygenerować zanim dojdzie do znanengo nam już tokenu końca sekwencji.

Zgromadzenie takich danych wymaga od nas z pewnością o wiele większych nakładów pracy, niż ma to miejsce w przypadku tworzenia zbiorów do treningu wstępnego. Tym razem musimy zaangażować do pracy mnóstwo osób, aby napisać satysfakcjonującą nas ilość dialogów.

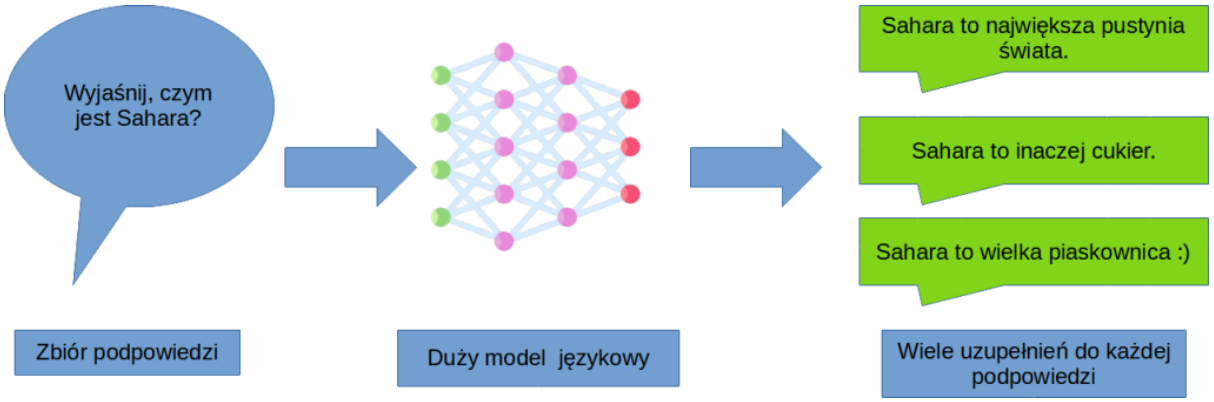
Pomijając te trudności, *instruction fine-tuning* ma też inne minusy. Mimo, iż w tego typu zbiorach często pojawi się coś co w szkole znamy jako „zadanie otwarte”, najwyżej kilka odpowiedzi może zostać uznanych za poprawne. Czyli, trzymając się dalej szkolnej metafory, odpowiedzi chatbota będą oceniane według stałego, z góry określonego klucza. Ograniczamy tym samym kreatywność modelu, gdyż w naszym zbiorze danych nie jesteśmy w stanie zawrzeć wyczerpującej listy wszystkich akceptowalnych odpowiedzi.

Uczenie przez interakcję z człowiekiem

Mówiąc precyzyjniej, jest to **uczenie przez wzmacnianie przez interakcję z człowiekiem (ang. *Reinforcement Learning from Human Feedback, RLHF*)**. Uczenie przez wzmacnianie to termin techniczny, określający pewną grupę algorytmów z dziedziny uczenia maszynowego. Polegają one na treningu modelu poprzez interakcję z pewnym środowiskiem. Przykładowo, model może zagrać milion razy w grę wyścigową i np. po pięciu tysiącach prób zaczyna rozumieć, że nie warto kończyć przejazdu dzwonem w przydrożne drzewo. Po milionie być może będzie w stanie wygrać wyścig, zachowując się jak doborowy kierowca rajdowy.

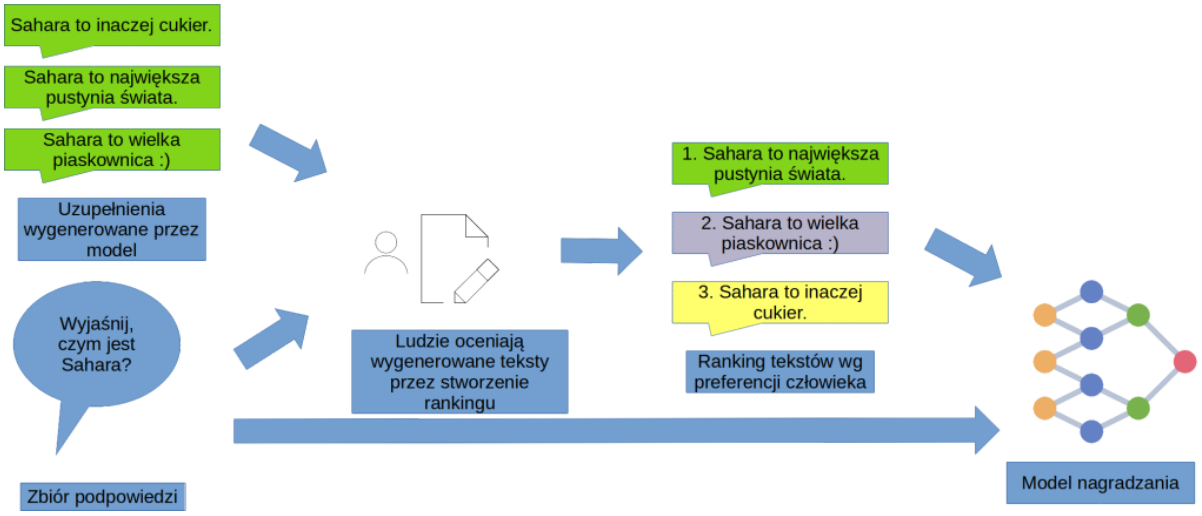
Wróćmy jednak do naszych modeli języka. LLM-y poddawane strojeniu RLHF możemy wcześniej opcjonalnie doszlifować na zbiorze pytań i odpowiedzi, ale przymusu nie ma, i od razu możemy przeskoczyć do uczenia przez wzmacnanianie.

W tym pierwszym kroku posiłkując się z góry przygotowaną listą pytań, pozwalamy naszemu modelowi nieco bardziej rozwinąć skrzydła i generujemy wiele różnych tekstów na podstawie tej samej podpowiedzi. Skorzystać przy tym możemy z tych samych przykładów, o których wspomniałem w poprzedniej części artykułu.



Wykorzystano ikonę z flaticon.com.

Następnie, prosimy respondentów o uszeregowanie odpowiedzi od najlepszej do najgorszej. Nasz zbiór będzie liczył tysiące odpowiedzi, więc musimy się sporo napracować. Po raz kolejny na etapie strojenia modelu okazuje się, że nie obejdzie się bez zatrudnienia do tego zadania dużej liczby osób.



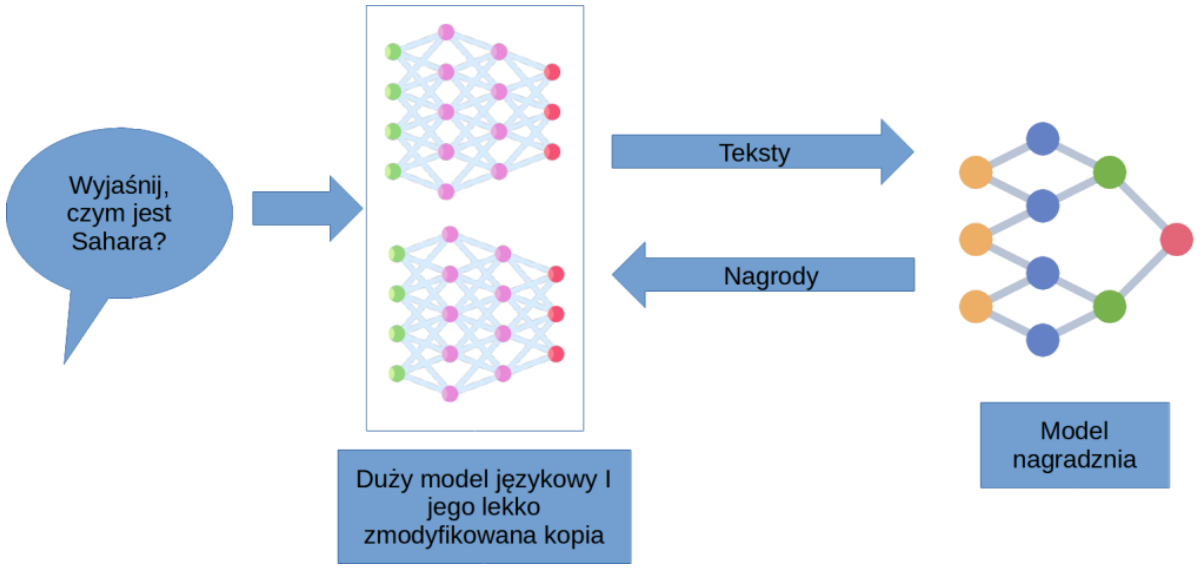
Wykorzystano ikonę z flaticon.com.

Żeby zrozumieć, co się tutaj dzieje, wyobraźmy sobie inny przykład. Gdybyśmy mieli dokonać oceny dziesięciu restauracji, moglibyśmy to zrobić na dwa sposoby – albo ułożyć je w kolejności od najlepszej do najgorszej, albo przyznać każdej np. od zera do pięciu gwiazdek. Oczywiście od gwiazdek możemy łatwo przejść do rankingu – wystarczy posortować po ocenie. Jesteśmy też w stanie – opierając się na pewnych założeniach – wykonać przekształcenie w drugą stronę. Możemy przykładowo założyć, że pierwsza pozycja w rankingu otrzymuje pięć gwiazdek, ostatnia zero, a wszystkie pozostałe wartości pośrednie, zależnie od ich miejsca.

Okazuje się, że w przypadku ręcznego oceniania tektów generowanych przez model, ludziom łatwiej jest w spójny sposób oceniać przykłady poprzez tworzenie rankingów niż poprzez przyznawanie naszych umownych gwiazdek. Ostatecznie jednak będziemy tych „gwiazdek" potrzebować, więc w sprytny sposób destylujemy je z utworzonych przez ludzi rankingów.

Następnie, na podstawie tych wartości trenujemy **model nagradzania** (ang. *reward model*), zwany również **modelem preferencji** (ang. *preference model*). Co ciekawe, to również jest model językowy, ale zamiast zgadywać następny token, próbuje on raczej przewidzieć, jak dany człowiek oceniłby uzupełnienie dopasowane do konkretnej podpowiedzi („ile gwiazdek by przyznał"). Tym sposobem nasz LLM otrzymuje wirtualnego partnera do zabawy w generowanie tekstu. Dzięki temu usprawniamy trening naszego modelu – próba dostrajania LLMa przez bezpośredni koktkat z człowiekiem trwałaby bowiem wieki.

Ostateczny trening odbywa się poprzez współdziałanie modeli – generujemy kilka alternatywnych uzupełnień podpowiedzi korzystając ze zmodyfikowanych kopii LLMa, a następnie model nagradzania stwierdza, który tekst jest najlepszy. Na tej podstawie modyfikujemy nasz LLM, by następnym razem udzielał lepszych odpowiedzi.



Wykorzystano ikonę z flaticon.com

To tyle, rzecz jasna w bardzo dużym uproszczeniu. Jeżeli interesują Was techniczne szczegóły poszczególnych etapów treningu, odsyłam do artykułów wylistowanych w bibliografii.

Bibliografia

1. Replit AI Team, *How to train your own Large Language Models*, dostęp: 13 lipca 2023
2. Avi Chawla, *LLM training and fine-tuning*, dostęp: 13 lipca 2023
3. Kamran Ahmed, *Introduction to LLMs*, dostęp: 13 lipca 2023
4. Clive Gomes, *Pre-training Large Language Models at Scale*, dostęp: 13 lipca 2023
5. Marine Carpuat, *Modeling language as a sequence of tokens*, dostęp: 13 lipca 2023
6. Steve Ickman, *INSTRUCT: Making LLM's Do Anything You Want*, dostęp: 13 lipca 2023
7. Utkarsh Ohm, *What we can learn from Google instruction finetuning its LLMs,* dostęp: 13 lipca 2023
8. Sebastian Raschka, *Finetuning LLMs Efficiently with Adapters*, dostęp: 13 lipca 2023
9. Tejpal Kumawat, *Basics of Prompt Engineering,* dostęp: 13 lipca 2023
10. Tammo Rukat, *Training Large Language Models: A high-speed overview*, dostęp 25 lipca 2023
11. Vladislav Lialin, Vijeta Deshpande, Anna Rumshisky. *Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning*, 2023
12. Jesse Mu, Prompting, Instruction Finetuning, and RLHF (Stanford Lecture), dostęp: 18 sierpnia 2023
13. Argilla, Train a reward model for RLHF, dostęp: 18 sierpnia 2023
14. Utkarsh Ohm, *What we can learn from Google instruction finetuning its LLMs | LinkedIn*, dostęp: 18 sierpnia 2023
15. code_your_own_AI (YouTube), *Instruction Fine-Tuning and In-Context Learning of LLM (w/ Symbols)* , dostęp: 18 sierpnia 2023
16. Nathan Lambert, Louis Castricato, Leandro von Werra and Alex Havrilla, *Illustrating Reinforcement Learning from Human Feedback (RLHF)*, dostęp: 18 sierpnia 2023
17. Hugging Face, *Padding and truncation*, dostęp: 18 sierpnia 2023
18. Shanul Es, *Reward Modeling for Large language models (with code)*
19. João Lages, *Reinforcement Learning from Human Feedback (RLHF) – a simplified explanation*