Obraz zawierający Grafika, Czcionka, logo, design

Opis wygenerowany automatycznie

Studium Magisterskie

Kierunek Big Data

Imię i nazwisko autora

Antoni Ziółkowski

Nr albumu 83133

Lora (outline jak na razie z fokusem na czas i pieniądz/fine-tuning llmów przez pojedyncze jednostki czy małe firmy)

Praca magisterska

pod kierunkiem naukowym

Dr Sebastiana Zająca

Instytut

Ekonometrii

Warszawa 2024

**Spis treści**

[Wstęp 6](#_Toc156047216)

[Rozdział 1. Ogólna charakterystyka dużych modeli językowych 7](#_Toc156047217)

[1.1 Historia modelowania językowego 7](#_Toc156047218)

[1.2 Architektura transformera 8](#_Toc156047219)

[1.2.1 Opis architektury 8](#_Toc156047220)

[1.3 Udoskonalenia architektury 10](#_Toc156047221)

[1.4 Proces uczenia dużych modeli językowych 11](#_Toc156047222)

[1.4.1 Metodologie uczenia 12](#_Toc156047223)

[Rozdział 2. LoRA: adaptacja LLMów macierzami niskiego rzędu 13](#_Toc156047224)

[2.1 Pojęcie adaptacji dużych modeli językowych 13](#_Toc156047225)

[2.1.1 LoRA 13](#_Toc156047226)

[2.2 Praktyczne zastosowanie w modelach 14](#_Toc156047227)

[2.3 Korzyści wynikające z zastosowania „LoRy” 14](#_Toc156047228)

[2.3.1 Oszczędność czasu i zasobów pieniężnych 14](#_Toc156047229)

[2.3.2 Dostępność dla mniejszych podmiotów 14](#_Toc156047230)

[2.4 Wyzwania i ograniczenia (zobaczę na bieżąco czy opłaca się pisać o downside’ach czy czymś innym) 14](#_Toc156047231)

[Rozdział 3. Implementacja praktyczna i ocena techniki adaptacji LoRA 14](#_Toc156047232)

[3.1 Projektowanie eksperymentów 14](#_Toc156047233)

[3.2 Metodologia 14](#_Toc156047234)

[3.3 Wyniki i analiza 14](#_Toc156047235)

[3.4 Dyskusja 14](#_Toc156047236)

[Rozdział 4. Przyszłość i rozwój dużych modeli językowych 14](#_Toc156047237)

[4.1 Nowe trendy i innowacje w LLM 14](#_Toc156047238)

[4.2 Rola adaptacji macierzami niskiego rzędu i innych narzędzi w rozwoju LLM-ów 14](#_Toc156047239)

[4.3 Wpływ społeczny i etyczny 14](#_Toc156047240)

[Zakończenie 15](#_Toc156047241)

[Bibliografia 16](#_Toc156047242)

[Spis tabel 17](#_Toc156047243)

[Spis wykresów 18](#_Toc156047244)

[Spis rysunków 19](#_Toc156047245)

[Streszczenie 20](#_Toc156047246)

Wstęp

Wrzucić na pewno krótką wspominkę/zarys historyczny NLP

Rozdział 1. Ogólna charakterystyka dużych modeli językowych

1.1 Architektura transformera

Architektura transformera miała skromne początki, przedstawiona została bowiem przez jej autorów głównie jako alternatywa do dotychczasowych sieci neuronowych stosowanych w tłumaczeniu maszynowym. Szybko okazało się jednak, że jej potencjalne zastosowania są znacznie szersze, zmieniła ona całkowicie krajobraz uczenia maszynowego. Cechuje się bowiem bardzo dobrą możliwością adaptacji do różnych zadań; można stosować ją także w scenariuszach niepowiązanych z procesowaniem języka naturalnego, na przykład składaniu białek, rozpoznawaniu obrazów, rozpoznawania mowy, czy uczenia ze wzmocnieniem[[1]](#footnote-1),[[2]](#footnote-2),[[3]](#footnote-3),[[4]](#footnote-4). Najistotniejszym zastosowaniem transformera w bieżących czasach są jednak generatywne modele językowe, bazują na niej produkty takie jak ChatGPT, Llama, czy Bard.

Głównym celem autorów było stworzenie sieci, która cechuje się lepszą pamięcią długoterminową i umiejętnością identyfikacji globalnych zależności w danych treningowych, w porównaniu do dotychczas używanych sieci rekurencyjnych i ich rozwinięć, takich jak LSTM czy GRU. Te bowiem, pomimo lat rozwoju, nadal nie radziły sobie zbyt dobrze z długimi sekwencjami danych wejściowych, a ponadto stwarzały trudności w procesie uczenia – rekurencję ciężko było zrównoleglić obliczeniowo.

1.1.1 Opis architektury

Transformer jest modelem seq2seq, zaproponowanym w prace „Attention Is All You Need”[[5]](#footnote-5), a jego najważniejszą cechą jest silne skupienie na i wykorzystanie mechanizmu Multi-Head Attention. Mechanizm uwagi w innych postaciach rozwijał się jeszcze przed transformerem, a za jego początek przyjmuje się pracę bowiem rozwijał się już w latach poprzedzających pracę Bahdanau et al. z 2014[[6]](#footnote-6). Oprócz zwiększonej ogólnej efektywności modelu w zadaniach tłumaczenia maszynowego czy generowania języka naturalnego, cechuje się on również dotychczas niespotykaną możliwością paralelizacji obliczeń. Stanowiło to swojego rodzaju mieszankę wybuchową dla rynku kart graficznych – kapitalizacja rynkowa lidera – korporacji NVIDIA, od 2017 do 2023 zwiększyła się z 117.26 miliardów dolarów do 2.272 bilionów dolarów[[7]](#footnote-7).

Danymi wejściowymi do modelu transformera są tokeny osadzone w *d*model wymiarowej przestrzeni wektorowej. Są one sumą wcześniej nauczonych osadzeń wejściowej sekwencji znaków i wektorów pozycyjnych. Jak bowiem zauważają autorzy, transformer nie wykorzystuje rekurencji ani konwolucji, zatem nie byłby w stanie nauczyć się zależności pozycyjnych między słowami bez dodania informacji pozycyjnych do ciągów wejściowych. Kodowanie pozycyjne zastosowane w „Attention Is All You Need” wyrażone jest następująco:

gdzie *pos* – pozycja

*i* –wymiar.

Źródło: Ashish Vaswani i in., „Attention Is All You Need” (arXiv, 1 sierpień 2023), http://arxiv.org/abs/1706.03762.

Transformer składa się z dwóch bloków: enkodera i dekodera. Budowa ta jest typowa i znana z poprzednich iteracji modelów seq2seq, skupiających się na tłumaczeniu maszynowym. Enkoder koduje ciągi znaków w języku obcym, a dekoder ma za zadanie wygenerowanie ciągu znaków w języku docelowym. W pierwotnej pracy obydwie warstwy ustawione są szeregowo i jest ich N=6.

Przygotowane sekwencje przekazywane są do enkodera, gdzie w pierwszym kroku przetwarzane są przy użyciu mechanizmu Multi-Head Attention. Pojedynczy składnik uwagi – „Scaled Dot-Product Attention” wyrażony jest następująco:

gdzie *Q* – macierz zapytań (queries)

*K* – macierz kluczy (keys)

*V* – macierz wartości (values)

*dk* – wymiar macierzy kluczy.

Mechanizm ten pozwala na kalkulację jak „ważne” są dla siebie nawzajem poszczególne sekwencje znaków; otrzymana wartość końcowa stanowi ważoną sumę wartości „value” z wagą, mierzoną jako zgodność zapytania i klucza. Uwaga w takiej postaci wykonywana jest na zestawach danych - V, K i Q są matrycami odpowiednich wartości. Pełna Multi-Head Attention natomiast, polega na rzucie liniowym wartości V, K i Q do przestrzeni o wymiarach dk, dv, dq przy pomocy warstwy liniowej. Obliczenia uwagi są następnie wykonywane równolege, a ich wyniki są łączone i ponownie rzutowane warstwą liniową.

gdzie projekcje są macierzami parametrów:

i .

W następnym kroku każda pozycja procesowana jest w pełni połączoną siecą feed-forward – dwie liniowe transformacje z funkcją aktywacji RELU pomiędzy. Ponadto, zastosowane są połączenia rezydualne, a zatem zarówno po bloku Multi-Head Attention i FFN, wykonywane są obliczenia addycji i normalizacji danych.

Blok dekodera zbudowany jest bardzo podobnie jak enkoder, z dwiema kluczowymi zmianami. Po pierwsze, matryce Q i K, są maskowane tak, aby sekwencje nie mogły stosować mechanizmu uwagi na tokenach „z przyszłości”. Innymi słowy, tokeny w środku sekwencji mogą się „uczyć” informacji jedynie o tokenach je poprzedzających. Drugą zmianą jest wprowadzenie drugiego bloku Multi-Head Attention, przez niektórych nazywanym Cross-Attention[[8]](#footnote-8). Tym sposobem sekwencje wejściowe bloku dekodera mogą „uważać” nie tylko na siebie, ale także na informacje z bloku enkodera, co jest kluczowe podczas tłumaczenia maszynowego.

1.2 Udoskonalenia architektury

Scaling up, multimodality

From models like GPT-2, which had 1.5 billion parameters, to GPT-3 with 175 billion parameters

Wraz z szerszym docenieniem transformera przez naukowców i firmy, podejmowane jest wiele działań w celu adaptacji i optymalizacji oryginalnej architektury do celi generowania tekstu/działania jako asystent. Jedną z najbardziej kluczowych zmian wykorzystywaną w większości modelów SOTA jest zupełne porzucenie bloku enkodera. Z empirycznego punktu widzenia, modele dekoder only sprawdzają się lepiej w open-text generation i prompt-based inference, natomiast modele enkoder-dekoder lepiej sprawdzają się w zadaniach klasyfikacyjnych i sequence to sequence. (bardzo dużo info tutaj jest ze str 6 papieru ULLP). Kluczowym jest również fakt, że porzucenie bloku encodera znacznie zmniejsza liczbę parametrów w modelach, a zatem pozytywnie wpływa na czas inferencji, koszty i skalowalność. Jak zauważają autorzy ULLP, modele enkoder-dekoder mają średnio 2x więcej parametrów od modelów wykorzystujących jedynie dekoder. Z perspektywy funkcjonalnej, w modelach enkoder-dekoder, sekwencje input procesowane są jedynie przez enkoder, natomiast dekoder zajmuje się sekwencjami target, a zatem obydwie z nich mają własne zestawy parametrów. W przypadku modeli porzucających enkoder, sekwencje input-target są łączone.

Wiele współczesnych modeli językowych wprowadza również zmiany w procesie kodowania pozycyjnego tokenów wejściowych. Przedstawiając oryginalną architekturę transformera wykorzystane zostało kodowanie tokenów w pozycjach bezwzględnych. Inni autorzy zaproponowali także a trainable absolute position encoding, a Su et al (roformer) osadzanie w pozycjach rotacyjnych; „ROPE koduje pozycję absolutną za pomocą macierzy rotacji, w tym samym momencie uwzględniając wyraźną zależność względnej pozycji w formułowaniu mechanizmu uwagi własnej”. Mechanizm ten cechuje się kilkoma plusami względem pozostałych metod: sequence length flexibility, wprowadza decaying inter-token dependency with increasing relative distances, capability of equipping the linear self-attention with relative position encoding.

Równie istotną innowacją są także transformery „rzadkie”. Cytat: Wymagania dotyczące pamięci i obliczeń takich sieci (transformerów) rosną kwadratowo wraz z długością sekwencji, co wyklucza ich użycie w długich sekwencjach. Autorzy wprowadzili zatem kilka rzadkich faktoryzacji macierzy atencji, co pozwoliło osiągnąć do tej pory niemożliwe długości sekwencji, jednocześnie nie tracąc przy tym wydajności modeli. Wykorzystując zaproponowane rozwiązanie, tokeny nie są maskowane macierzami trójkątnymi, lecz rzadkimi, jak na obrazku poniżej: (tutaj screen z sparse transfomers). Zastosowanie takich macierzy pozwala osiągnąć SOTA wyniki, jednocześnie przedłużając context length i zmniejszając ilość wymaganych obliczeń.

1.3 Proces uczenia dużych modeli językowych

Metody uczenia sieci bazujących na architekturze transformera, przewidywalnie, różni się w zależności od modelowanego problemu czy zadania jakie model ma wykonywać. W przypadku modeli stworzonych do modelowania językowego, a dokładniej generujących język naturalny, bardzo popularnych w bieżących czasach, chatbotów-asystentów, mamy do czynienia z modelami typu GPT – Generative Pre-trained Transformer. Ze względu na bardzo dużą złożoność języka naturalnego, modele tego typu cechują się ogromną liczbą parametrów – miliony, miliardy, a zatem nawet proces inferencji, tj. generowania tekstu z nauczonego już modelu, wymaga klastrów procesorów graficznych, aby obywał się on w racjonalnym czasie. Charakterystyka ta jeszcze istotniej prezentuje się jednak w procesie uczenia modeli, który zajmuje/kosztuje (tutaj przykład openai albo llamy), a zatem jest fizycznie niewykonalny przez pojedyncze jednostki bez korporacyjnego finansowania. Proces uczenia takiego asystenta zaczyna się od „pre-trainingu” t.j. trenowania parametrów na gigantycznym korpusie danych (unlabelled text); cały internet (tutaj ładniej opisać, sprawdzić w papierach, w gpt3 opisują czego i ile użyli). Oczywiście im lepsza jakość danych wejściowych, tym lepsze będą wyniki modelu, a zatem dużo uwagi w ostatnim czasie przykłada się do przygotowania/oczyszczania danych wejściowych. Dla przykładu, firma OpenAI silnie polegała na danych z platformy Reddit, zaciągając wątki mające 3 upvote’y lub więcej. Po wykonaniu uczenia wstępnego modele tego typu nie są niczym innym niż generatorami losowego tekstu i daleko im jeszcze do komercjalnie dostępnych produktów. Kluczowym etapem uczenia jest więc „fine-tuning” gdzie sekwencje wejściowe przyjmują postać konwersacji Asystent-Użytkownik o charakterze konwersacyjnym. W porównaniu do ogromnych zasobów potrzebnych do pre-trainingu, modele względnie szybko adaptują się do poszczególnych zadań używając fine-tuningu. (jakieś źródło/porównanie resourców potrzebnych do pre-train/fine-tune). Proces ten nie zapewnia jednak, że model będzie akceptowalny etycznie i moralnie, szczególnie biorąc pod uwagę skalę end-userów. Częste jest zatem, że firmy w tym momencie wykorzystują techniki takie jak RLHF, gdzie to ludzie oceniają tekst generowany przez model w celu poprawienia AI „alignmentu”. Warto w tym miejscu zwrócić uwagę, że proces zwany pre-trainingiem nie został wprowadzony wraz z genezą architektury transformera, był bowiem wykorzystywany już wcześniej i w innych architekturach: (Mikolov et al., 2013; Pennington et al., 2014; Neumann et al., 2018; Dai & Le, 2015; Howard & Ruder, 2018) – papier Unifying Language Learning Paradigms – strona 6

Również na tej samej stronie, niżej, opisane dlaczego decoder only sprawdzał się lepiej w open ended text generation

Info sprawdzić:

Generative pretraining (GP) was a long-established concept in machine learning applications.[16][17][18] It was originally used as a form of semi-supervised learning, as the model is trained first on an unlabelled dataset (pretraining step) by learning to generate datapoints in the dataset, and then it is trained to classify a labelled dataset.”

1.3.1 Metodologie uczenia

Rozdział 2. LoRA: adaptacja LLMów macierzami niskiego rzędu

2.1 Pojęcie adaptacji dużych modeli językowych

2.1.1 LoRA

**Tabela 1. Tabela testowa**

|  |  |
| --- | --- |
| 1 | 2 |
| 3 | 4 |

Źródło: ;kljlkjlkj

**Rysunek 1. Rysunek testowy**

Obraz zawierający tekst, zrzut ekranu, diagram

Opis wygenerowany automatycznie

2.2 Praktyczne zastosowanie w modelach

2.3 Korzyści wynikające z zastosowania „LoRy”

2.3.1 Oszczędność czasu i zasobów pieniężnych

2.3.2 Dostępność dla mniejszych podmiotów

2.4 Wyzwania i ograniczenia (zobaczę na bieżąco czy opłaca się pisać o downside’ach czy czymś innym)

Rozdział 3. Implementacja praktyczna i ocena techniki adaptacji LoRA

3.1 Projektowanie eksperymentów

3.2 Metodologia

3.3 Wyniki i analiza

3.4 Dyskusja

Rozdział 4. Przyszłość i rozwój dużych modeli językowych

4.1 Nowe trendy i innowacje w LLM

4.2 Rola adaptacji macierzami niskiego rzędu i innych narzędzi w rozwoju LLM-ów

4.3 Wpływ społeczny i etyczny

Zakończenie

Bibliografia

Spis tabel

**Tabela 1. Tabela testowa** 8

Spis wykresów

**Nie można odnaleźć pozycji dla spisu ilustracji.**

Spis rysunków

**Rysunek 1. Rysunek testowy** 8

Streszczenie

1. Alexey Dosovitskiy i in., „An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale” (arXiv, 3 czerwiec 2021), http://arxiv.org/abs/2010.11929. [↑](#footnote-ref-1)
2. John Jumper i in., „Highly Accurate Protein Structure Prediction with AlphaFold”, *Nature* 596, nr 7873 (26 sierpień 2021): 583–89, https://doi.org/10.1038/s41586-021-03819-2. [↑](#footnote-ref-2)
3. Anmol Gulati i in., „Conformer: Convolution-augmented Transformer for Speech Recognition” (arXiv, 16 maj 2020), http://arxiv.org/abs/2005.08100. [↑](#footnote-ref-3)
4. Lili Chen i in., „Decision Transformer: Reinforcement Learning via Sequence Modeling” (arXiv, 24 czerwiec 2021), http://arxiv.org/abs/2106.01345. [↑](#footnote-ref-4)
5. Ashish Vaswani i in., „Attention Is All You Need” (arXiv, 1 sierpień 2023), http://arxiv.org/abs/1706.03762. [↑](#footnote-ref-5)
6. Dzmitry Bahdanau, Kyunghyun Cho, i Yoshua Bengio, „Neural Machine Translation by Jointly Learning to Align and Translate” (arXiv, 1 wrzesień 2014), http://arxiv.org/abs/1409.0473. [↑](#footnote-ref-6)
7. „NVIDIA (NVDA) - Market Capitalization”, dostęp 13 marzec 2024, https://companiesmarketcap.com/nvidia/marketcap/. [↑](#footnote-ref-7)
8. Vaclav Kosar, „Cross-Attention in Transformer Architecture”, 28 grudzień 2021, https://vaclavkosar.com/ml/cross-attention-in-transformer-architecture. [↑](#footnote-ref-8)