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

Rozdział 1. Ogólna charakterystyka dużych modeli językowych

* 1. Historia modelowania językowego

1. początki – algorytmy, eliza etc
2. modele ngram
3. rnn,lstm
4. transformery

1.2 Architektura transformera

Architektura transformera miała skromne początki, przedstawiona została bowiem przez jej autorów głównie jako ulepszenie dotychczasowych sieci neuronowych stosowanych w tłumaczeniu maszynowym. Jak jednak okazało się w latach po jej zaproponowaniu, zmieniła ona całkowicie krajobraz uczenia maszynowego. Cechuje się ona bowiem prawie nieograniczonymi możliwościami adaptacji do różnych zadań; można stosować ją nawet w dziedzinach, gdzie do tej pory używane były bardzo wyspecjalizowane modele, np. CNN w rozpoznawaniu obrazów czy …… Najistotniejszym zastosowaniem transformera w bieżących czasach wydaje się być jednak modelowanie językowe, bowiem pozwolił on na kreację takich produktów jak ChatGPT przez OpenAI, Llama przez Meta, czy Bard od Google. Być może nie powinno to dziwić, biorąc pod uwagę, że od początku proponowana była do zadań związanych z NLP, natomiast nawet autorzy pierwotnej pracy nie spodziewali się wpływu jaki wywrze ona na świat deep learningu. (tutaj link do tweeta może być np)

Głównym celem autorów było stworzenie sieci, która cechuje się lepszą pamięcią globalną i umiejętnością identyfikacji globalnych zależności między danymi wejściowymi a wyjściowymi od dotychczas używanych RNN czy CNN. Te bowiem, pomimo lat rozwoju, nadal nie radziły sobie zbyt dobrze z długimi sekwencjami danych wejściowych, a ponadto stwarzały trudności w procesie uczenia – rekurencję i konwolucję ciężko było zrównoleglić obliczeniowo.

1.2.1 Opis architektury

Transformer jest modelem seq2seq, a jego najważniejszą cechą jest silne skupienie na i wykorzystanie mechanizmu uwagi własnej (Self Attention). Nie był on co prawda zaproponowany przez autorów „Attention is all you need”, bowiem rozwijał się już w latach poprzedzających pracę, (linki do modeli wykorzystujących self attention) jednakże w przeciwieństwie do poprzednich prób, gdzie mechanizm uwagi był raczej „doklejany” do istniejących modeli, tutaj wokół jego optymalizacji zbudowano całą architekturę. Co jednak wprowadzone zostało w pracy to rozwinięcie tego mechanizmu i zaproponowanie wielowątkowego mechanizmu uwagi własnej (multi head attention). Oprócz zwiększonej ogólnej efektywności modelu w zadaniach tłumaczenia maszynowego czy generowania języka naturalnego, cechuje się on również dotychczas niespotykaną możliwością paralelizacji obliczeń. Zjawisko to miało na tyle duży wpływ, że rynek kart graficznych od 2017 do 2023 rozwinął się o xxxx… (no, to trochę przepisać, rynek kart graficznych i tak się dobrze rozwijał, jeśli nie znajdę nic co 1:1 pokazuje tą zależność to wywalić to z pracy)

Transformer składa się z dwóch bloków: enkodera i dekodera. Budowa taka znana i typowa jest w sieciach 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.

Tak przygotowane sekwencje są następnie przekazywane do enkodera, gdzie w pierwszym kroku przetwarzane są przy użyciu mechanizmu Multi-Head Attention. W skrócie:

Mechanizm ten pozwala na kalkulację jak „ważne” są dla siebie nawzajem poszczególne sekwencje znaków; otrzymana finalnie wartość stanowi ważoną sumę wartości „value” z wagą, mierzoną jako zgodność „query” i „key”. Uwaga w takiej postaci wykonywana jest na zestawach danych - V, K i Q są matrycami odpowiednich wartości. Uwaga wielowątkowa 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 sieć 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 aby sekwencje nie mogły stosować mechanizmu uwagi na tokenach „z przyszłości”. Dla przykładu: słowo w środku zdania może się uczyć informacji jedynie o słowach je poprzedzających. Drugą zmianą jest wprowadzenie kolejnego bloku wielowątkowej uwagi, przez niektórych nazywanym cross-attention (karpathy). Tym sposobem sekwencje wejściowe bloku dekodera mogą „uważać” nie tylko na siebie, ale także na informacje z bloku enkodera, co jest kluczowe podczas efektywnego tłumaczenia maszynowego.

Kluczowe – osadzanie i kodowanie pozycyjne (dodać wzory i jakie dokładnie kodowanie zastosowali w attention is…)

Przed dokładnym opisem bloków należy jednak zwrócić uwagę na ich dane wejściowe – są to tokeny osadzone w przestrzeni wektorowej, w attention is all you need, 512 wymiarowej. Tokeny same w sobie natomiast są arbitralnym „pocięciem” wejściowej sekwencji zdań, z dodaniem równej wielkości wektorów z informacjami o pozycji słów w sekwencji (positional encodings). Jak zauważają autorzy, transformer nie wykorzystuje rekurencji, ani konwolucji, zatem nie byłby w stanie nauczyć się zależności pozycyjnych między słowami bez takiej modyfikacji danych wejściowych.

W przypadku tego modelu enkoder-dekoder, enkoder od początku otrzymuje całe zdanie w języku obcym, a dekoder działa autoregresywnie, wieloktronie używając swojego outputu jako inputu.

1.3 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.

1.4 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.”

1.4.1 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