Praktyczne wprowadzenie do LLM-ów

SPIS TREŚCI

Informacje organizacyjne

- Dzisiejsza prezentacja ma wprowadzić temat bardzo ogólnie, z praktycznego punktu widzenia.
- Będzie kilka informacji technicznych, najważniejszych pojęć, najbardziej znanych architektur, natomiast nie będzie to zaawansowana matematyka.
- Jeśli ktoś szukałby więcej informacji, w README repozytorium będą pojawiać się linki do dodatkowych materiałów.

Przypomnienie

- **model** - konstrukcja matematyczna, dostaje pewien input, przetwarza go z użyciem wielu funkcji i struktur i zwraca nam output o pewnych zdefiniowanych przez nas cechach
- **supervised machine learning** (uczenie nadzorowane) - model dostaje input zawierający pewne features, a także odpowiadające im etykiety (labels)
- **unsupervised machine learning** (uczenie nienadzorowane) - model nie dostaje etykiet jako input, sam musi znaleźć pewien pattern w danych (np. podział na clusters)
- **neural network** - model zawierający przynajmniej jedną warstwę ukrytą (hidden layer)
- **evaluation (metrics)** - sposób na ocenę jakości modelu i porównanie go z innymi
- **pre-trained model** - model, który został już wyuczony do jakiegoś zadania
- **fine-tuning** - kiedy chcemy dopasować pre-trained model do naszych potrzeb (np. jest nauczony na tekstach ze wszystkich kierunków studiów na UW, a my chcemy, aby pomógł nam w historii)

Natural Language Processing

- Łączy AI i lingwistykę, automatyzuje przetwarzanie języka i jego generowanie.
- Znane zadania:
  - Sentiment analysis - jakie emocje są wyrażone w tekście (np pozytywne/ negatywne/ neutralne)
  - Named Entity Recognition (NER) - wykrywanie rodzajów słów np. które słowa określają ludzi, które miejsca, które lokalizację
  - Optical character recognition (OCR) - ekstrakcja tekstu z danego obrazka
  - Part-of-speech tagging (POS) - rzeczownik, czasownik, przymiotnik
  - Topic segmentation/ recognition - identyfikacja głównego tematu dla danego segmentu tekstu
  - Text summarization
  - Translation
- Metryki - przykłady
  - Accuracy, Precision, Recall, F1 Score - gdy mamy etykiety
  - BLEU - Bilingual evaluation understudy - liczy pokrycie między wyjściowymi zdaniami i referencyjnym ground truth

Modele językowe

- Uczą się na bardzo dużych zbiorach danych.
- Przewidują kolejne tokeny/ sekwencje tokenów w tekście (bazując na prawdopodobieństwie)
  - Token - pewna niepodzielna część, na której model się uczy i którą przewiduje np. ‘pies’, pojedyncze znaki
- Celem jest wygenerować sensowny tekst (np autouzupełnianie wiadomości w telefonie)
- “Niebo jest…”
  - “niebieskie”
  - “pochmurne”
  - “pomarańczowe”
- Dawniejsze metody opierały się na statystyce, a nie głębokim uczeniu i transformerach
  - n-grams bazują na analizie częstości występowania pewnych ciągów słów - jakie jest prawdopodobieństwo, że po słowie ‘jedzą’ wystąpi ‘ryby’. Wady to brak ‘pamiętania’ długiego kontekstu i bardzo dużo potencjalnych kombinacji.
  - Bag of Words - tekst to zbiór słów, bez uwzględnienia kolejności, każde słowo ma przypisaną wagę. Np ‘Ala ma kota’ i ‘Kot ma Alę’ dla takiej metody będzie tym samym.
  - Typowe problemy:
    - brak zrozumienia dłuższego kontekstu
    - problemy z wieloznacznością i złożoną składnią (np. zamek i zamek)

Large language models

- Są trenowane na bardzo dużych zbiorach danych
- Mają zaawansowaną architekturę
- Są znacznie lepsze w rozumieniu kontekstu niż poprzednicy
- Mają miliardy parametrów
  - parametry = wagi, które zmieniają input w ouptut
- Ich proces uczenia jest bardzo długi
  - pretraining - model uczy się charakterystyki języka
  - fine-tuning - do konkretnego zadania, np prawo, medycyna
- Trzeba mieć duże zasoby, żeby móc taki model dotrenować

- Warto pamiętać, że istnieją też SLM - Small Language Models. Mają mniej parametrów niż LLMs, są trenowane na mniejszej bazie danych, bardziej sprecyzowanej pod dany task.

Transformers

- Skupia się na najważniejszych częściach inputu, nie “zapomina” ważnych części inputu, jak to miało miejsce w poprzednich modelach (vanishing gradient problem)
- Modele z tą architekturą znacznie lepiej rozumieją kontekst
- Encoder-decoder architektura
  - encoder: input -> vector (mathematical representation, podsumowanie najważniejszych rzeczy w inpucie)
  - decoder: vector -> output
- “What is the color of the sky?” -> color, sky, blue -> “The sky is blue”
- Attention: pomaga zdefiniować modelowi, które fragmenty zdań są najważniejsze względem siebie
- Self-attention: pozwala na znalezienie wielu rodzajów relacji między fragmentami tego samego inputu

- Dla chętnych: *Attention is All You Need* (Vaswani et al. 2017)

Prompt

- Polecenia/ zapytanie, które przekazujemy modelowi językowemu, aby uzyskać odpowiedź.
- Aby otrzymać sensowną odpowiedź, trzeba jasno określić, czego oczekujemy od modelu.
- Gdy model analizuje nasz prompt, stara się przewidzieć najbardziej prawdopodobną odpowiedź.
- Model stara się też stylem odpowiedzi dopasować do naszych oczekiwań.
    - “Kim był Albert Einstein?”
    - “Zachowując styl Szekspira, napisz dialog między Romeo a Julią”

Prompt - dobre rady

- Bądź precyzyjny
  - “Opisz budynek” -> “Opisz zalety mieszkania w kamienicy w porównaniu z nowoczesnym budownictwem”
- Podaj kontekst
  - “Wyobraź sobie, że jesteś nauczycielem historii. Wyjaśnij uczniom szkoły podstawowej jak wyglądało życie w Średniowieczu”
- Określ format odpowiedzi
  - “Odpowiedź podaj w punktach”

Przykłady modeli - Llama

- https://www.llama.com
- Meta AI
- https://github.com/meta-llama/llama-models
- Open access
- Dane: głównie po angielsku, 5% danych treningowych w 30 innych językach.

- “Quantized”
  - Celem jest mniejsze wymagania odnośnie mocy obliczeniowych i pamięci.
  - Na przykład poprzez zmniejszanie dokładności zapisu wag (np 8 a nie 32 bity)
    - Zamienienie przestrzeni z wartościami np z float32 na przestrzeń int8 (256 wartości)
  - Znacznie szybsze są wtedy obliczenia takie jak mnożenie macierzy.

Przykłady modeli - GPT

- OpenAI
- Multimodalny - teksty, obrazy, audio, video
- API płatne
- Dane: publicznie dostępne oraz które dostali od firm
- Trening: pretrenowanie na danych, aby przewidywał kolejny token, a potem dotrenowany aby dawał etyczne i społecznie poprawne odpowiedzi.
- https://openai.com/chatgpt/overview/
- Nie open source

Przykłady modeli - PLLuM

- Polski duży model językowy
- https://pllum.org.pl
- Bezpłatny
- Docelowo dostosowany do “polskich realiów”, asystent wspomagający pracę administracji publicznej
- Dane: “wysokiej jakości” dane polskojęzyczne
- https://pllum.clarin-pl.eu/pllum_8x7b
- lub na HuggingFace

Przykłady modeli - Deepseek

- Chiński model
- https://www.deepseek.com
- API, chat oraz lokalnie, jest GitHub

Co warto wiedzieć

- Do LLMów (nie lokalnie postawionych) NIE wrzucamy danych poufnych, prywatnych
- Halucynacje
- Bias (uprzedzenia, stereotypy)
- Unexpected output (dane treningowe)
- Bezpieczeństwo - overreliance, prywatność danych
- Od czego zależy output - różne parametry, dane treningowe, różne języki
- Twórcy starają się aby modele zachowywały się etycznie i zrozumiale
    - Reinforcement Learning from Human Feedback (RLHF)
    - Filtrowanie inputu i outputu, czy nie ma tam szkodliwych treści
    - Chronienie danych użytkownika
    - Testowane przeciwko atakom
    - Zwiększanie wyjaśnialności i przejrzystości decyzji modeli, aby zwiększać zaufanie od użytkowników

Co można modyfikować

- Użyty model - wiele benchmarków
- Temperatura
- Prompt
- Czy chcecie dostać prawdopodobieństwo tokenu
- Długość odpowiedzi modelu
- Oraz wiele innych parametrów, więcej informacji znajdziecie w dokumentacji np. dla OpenAI lub tutaj

Jak używać i polecane narzędzia

- Via web np chatgpt.com
- Via API
- Lokalnie
  - prywatność danych
  - dostępność offline
  - oszczędność kosztów
  - i nie jest to wcale takie trudne obecnie ;)
  - chociaż polecamy mniejsze modele do stawiania lokalnie
- Langchain
  - framework do budowania aplikacji z użyciem LLM
- Kodu:
  - colab, deepnote
  - lub oczywiście lokalnie

Przykładowe zastosowania

- Chatboty
- Automatyczne tłumaczenie
- Generowanie kodu
- Tworzenie podsumowań tekstu
- Tworzenie artykułów
- Bonus: inne generatywne modele
  - obrazy
  - filmy
- Jakieś przykłady narzędzi?

Wady i zalety LLMs

- Zalety
  - wszechstronne (teksty, kody, podsumowania)
  - optymalizowanie czasu przygotowania żmudnych materiałów
  - integracja z innymi narzędziami typu bazy danych
  - wsparcie dla wielu języków
- Wady
  - nie dla wszystkich języków działają tak samo dobrze
  - potrafią halucynować (asystenci, ale nie źródło pracy)
  - bardzo drogie trenowanie i potem również inferencja, zarówno pieniądze jak i energia
  - bias, dezinformacja, zagrożenie prywatności
  - overreliance na odpowiedziach modelu

Praca domowa

- Praca domowa na 4 punkty: 2pkt za prostszą wersję, +2 za trudniejszą.
- W folderze homeworks plik ‘homework1_llms.txt’.
- Kod w Pythonie.
- Prompty i działanie z modelami przez kod (nie w web).
- Cel: odszyfrować teksty zawarte w pliku.
- Wersja prostsza: używając API dowolnego modelu.
- Wersja trudniejsza: postawić lokalnie model i wtedy go użyć.

Praca domowa - jak wysyłać

- fork repo z GitHuba
- do folderu ‘homeworks/{imię_nazwisko}’ wrzucić swój kod i plik .txt lub .md z linkiem do nagrania oraz plik .txt z odszyfrowanymi tekstami
  - nagranie wersja prostsza: 2-3min
  - nagranie wersja trudniejsza 3-5min
  - kod: preferowany format to Jupyter Notebook z ładnymi komentarzami
- zgłosić PR ze zmianami
- na GitHuba NIE wrzucamy:
  - danych
  - modeli
  - obrazów dockera - proszę o wrzucenie samych plików konfiguracyjnych (np. dockerfile)
- jeśli czymś z powyższych chcielibyście się pochwalić/ opowiedzieć, odpowiednim miejscem na to jest nagranie :)