

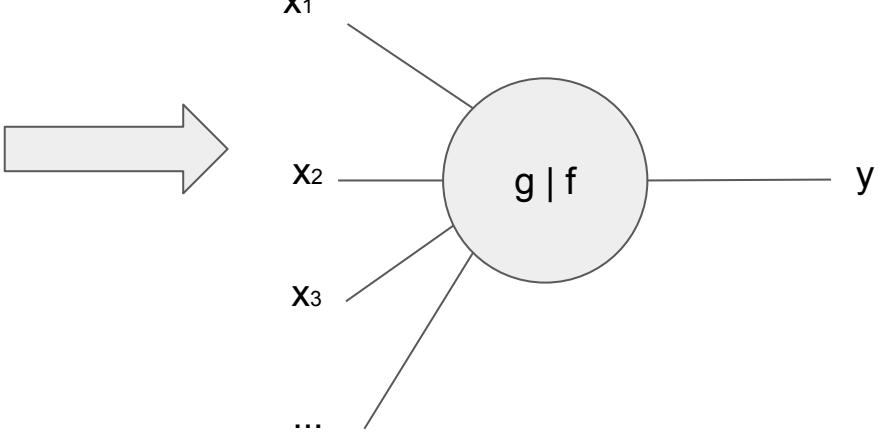
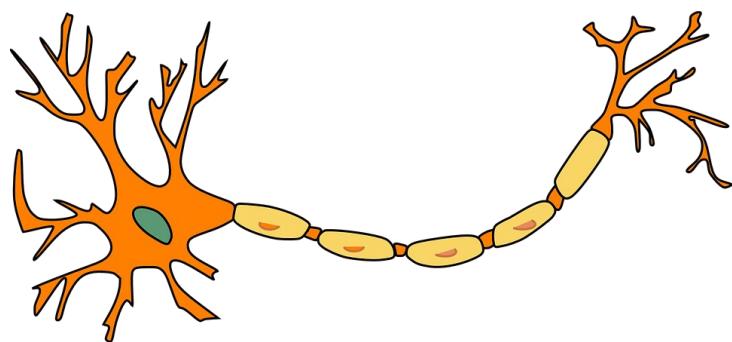
Generatywna sztuczna inteligencja z dużymi modelami tekstowymi

Architektura i rodzaje dużych modeli tekstowych

Michał Żarnecki

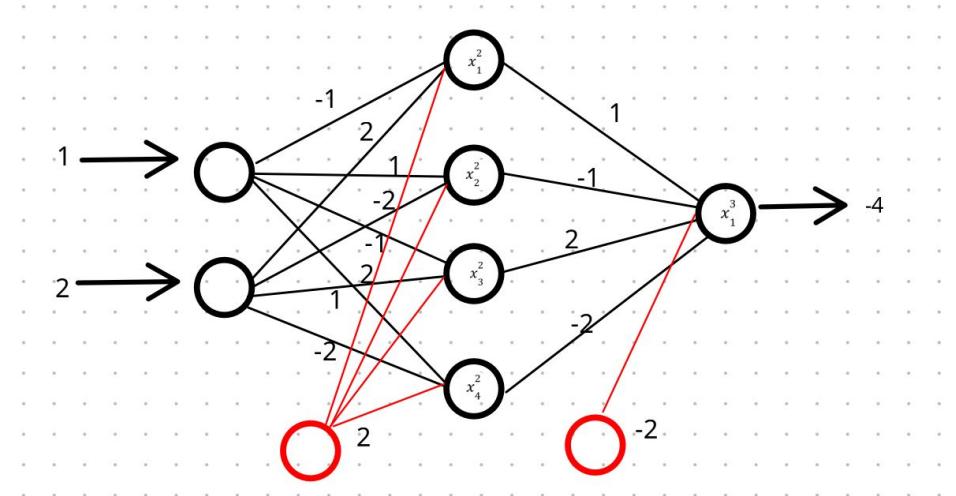


Model matematyczny neuronu



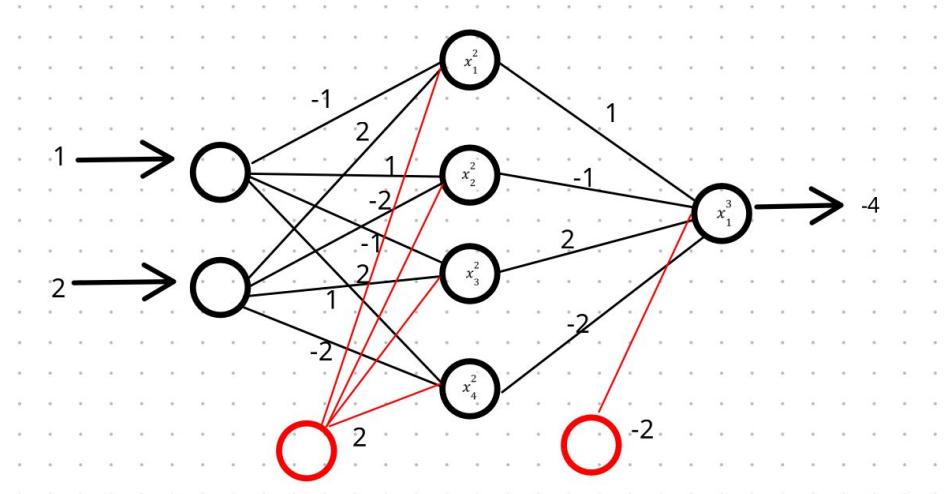
1943 Warren McCulloch and Walter Pitts proponują matematyczny model neuronu

Perceptron



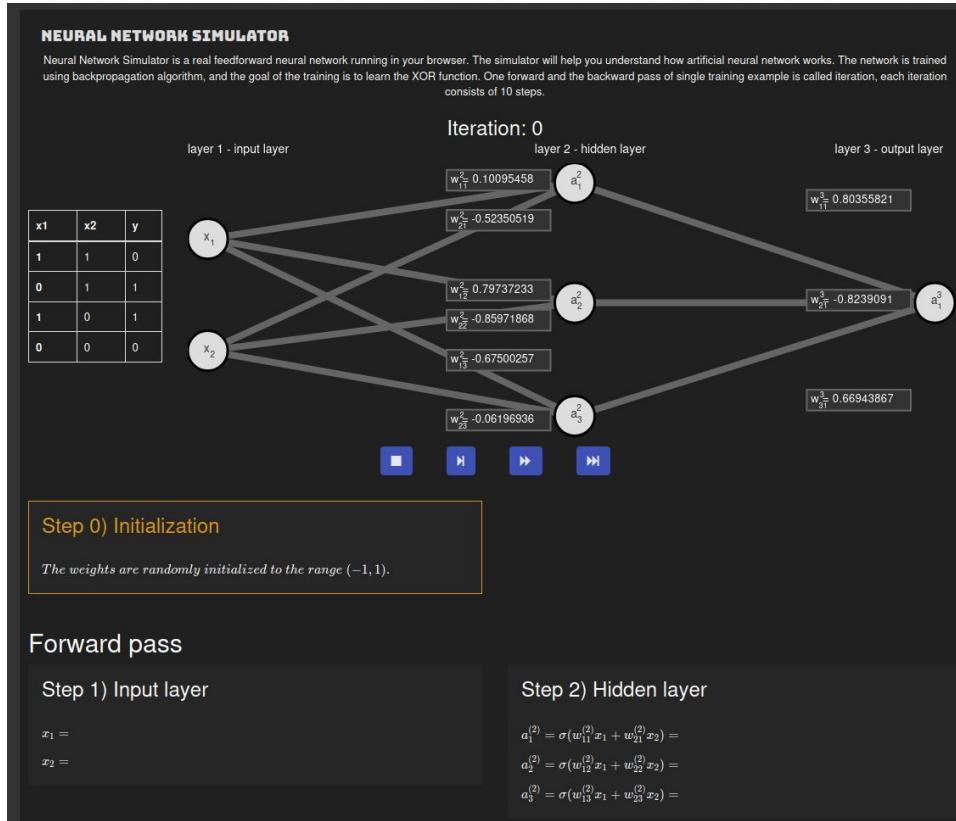
1957 Frank Rosenblatt proponuje model perceptronu,
który jest podstawowym elementem budowy sztucznych sieci neuronowych

Perceptron



artificial_neural_network.xopp

Jak działają sztuczne sieci neuronowe?



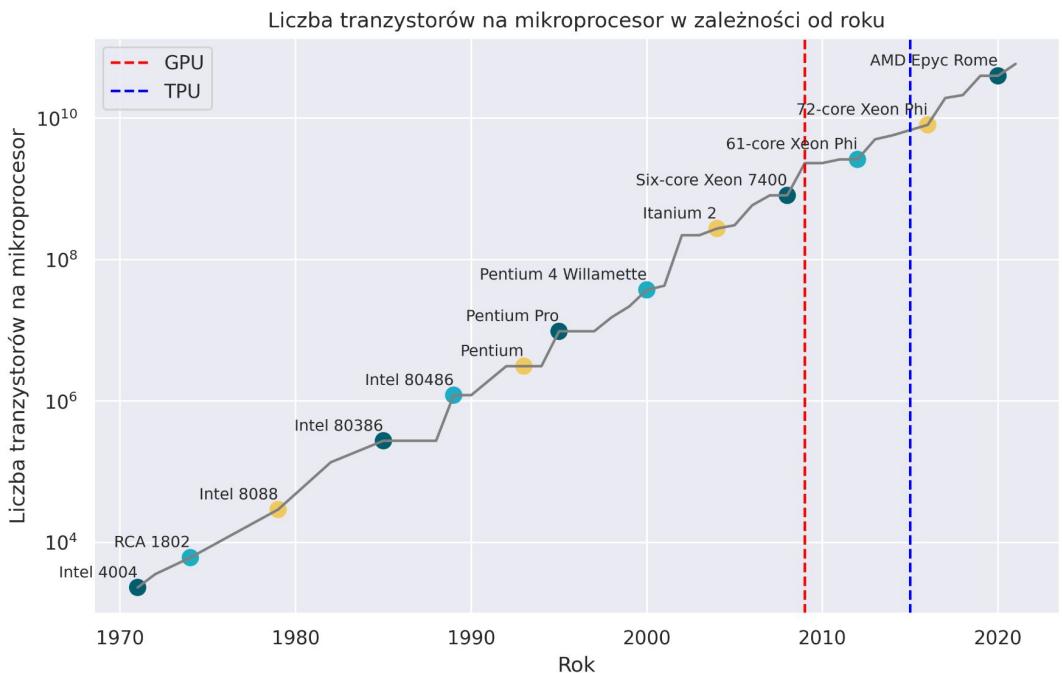
<https://www.mladdict.com/neural-network-simulator>

Zadanie

Skorzystaj z symulatora <https://www.mladdict.com/neural-network-simulator> i określ po ilu iteracjach propagacji wstecznej sieć zwraca prawidłowe wyniki dla bramki XOR.

?

prawo Moore'a



Przez dekady niedostatki mocy obliczeniowej i bariery technologiczne spowalniają zastosowanie algorytmów uczenia maszynowego w i sztucznych sieci neuronowych w praktyce.

Attention is all you need

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Lukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention

Attention is all you need

“GPT 2 too dangerous to be released”

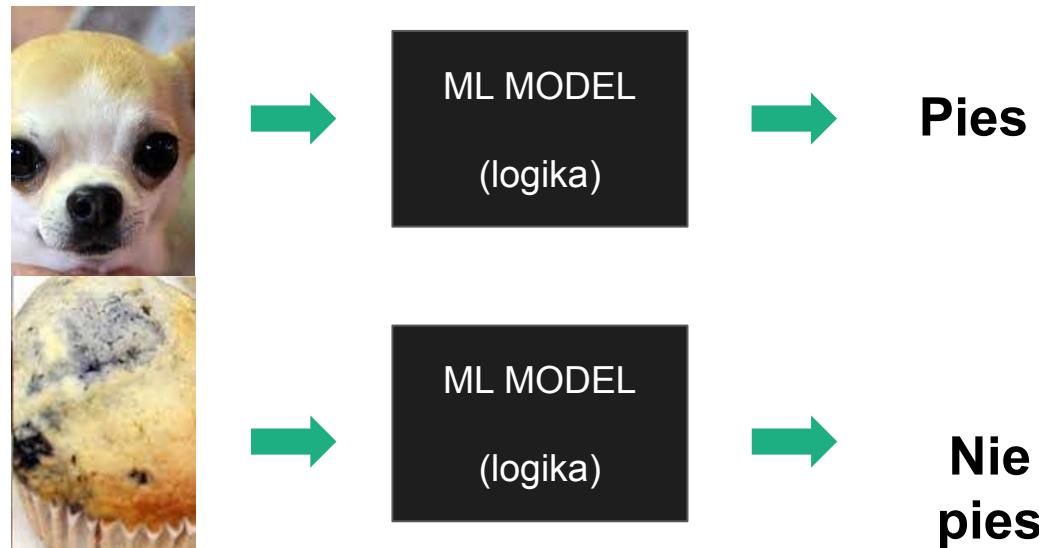
created: 02.2019

published: 11.2019

Our model, called GPT-2 (a successor to [GPT](#)), was trained simply to predict the next word in 40GB of Internet text. Due to our concerns about malicious applications of the technology, we are not releasing the trained model. As an experiment in responsible disclosure, we are instead releasing a much [smaller model](#) for researchers to experiment with, as well as a [technical paper](#).

Model

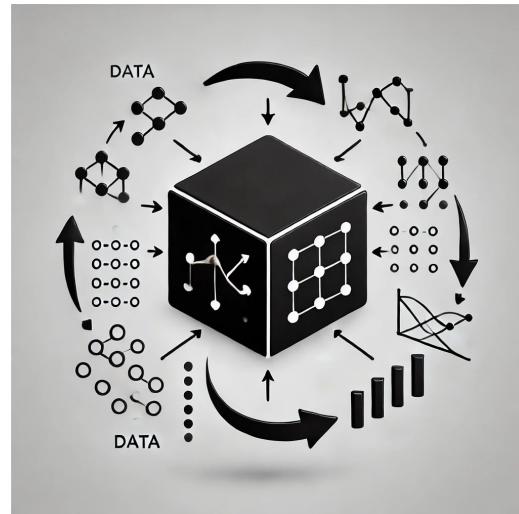
Model uczenia maszynowego to matematyczna lub statystyczna struktura, która jest trenowana na zbiorze danych w celu wykonywania zadań, takich jak klasyfikacja, predykcja, rozpoznawanie wzorców czy optymalizacja. Model ten uczy się na podstawie dostarczonych danych wejściowych (treningowych) i wyciąga na ich podstawie wnioski, które mogą być następnie używane do przewidywania wyników na nowych, nieznanych danych.



Model

W procesie uczenia model dostosowuje swoje parametry, aby minimalizować różnicę między przewidywaniami a rzeczywistymi wynikami, co pozwala na poprawę jego dokładności i efektywności.

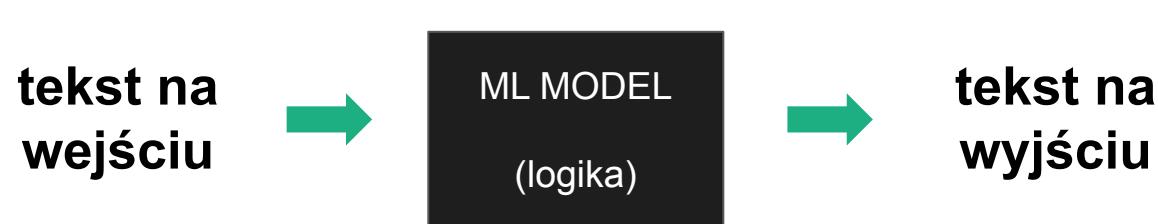
Modele uczenia maszynowego mogą przybierać różne formy, od prostych modeli liniowych, przez drzewa decyzyjne, po złożone sieci neuronowe używane w głębokim uczeniu.



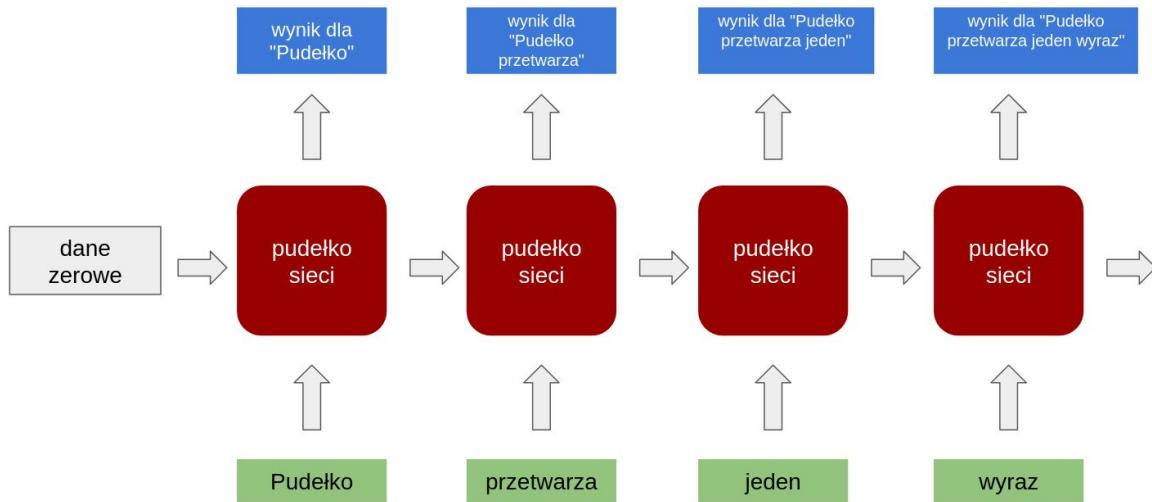
by DALL-E

Model tekstowy

Model tekstowy to rodzaj modelu uczenia maszynowego, który jest trenowany na zbiorze danych tekstowych w celu zrozumienia, generowania lub przetwarzania tekstu w języku naturalnym. Modele tekstowe mogą wykonywać różne zadania, takie jak tłumaczenie języków, klasyfikacja tekstu, rozumienie kontekstu, odpowiadanie na pytania, generowanie nowych treści tekstowych itp.

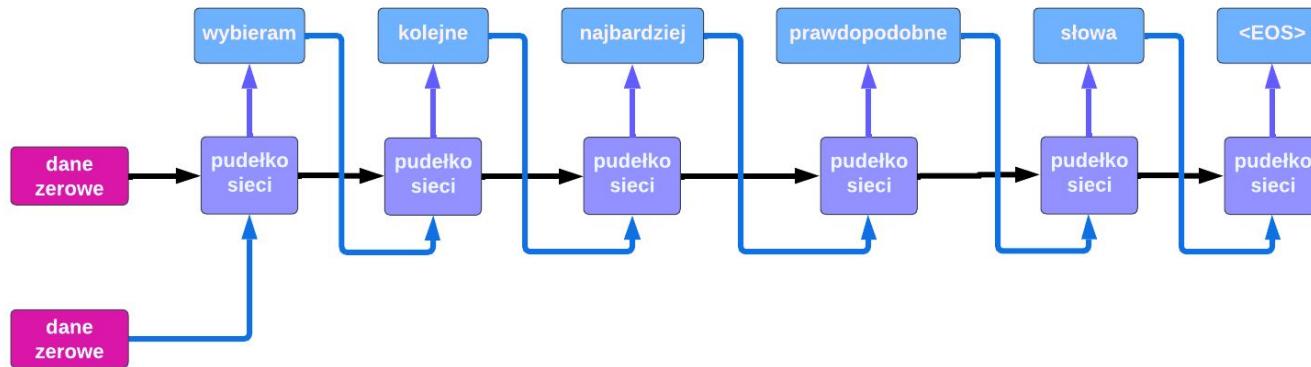


Model tekstowy



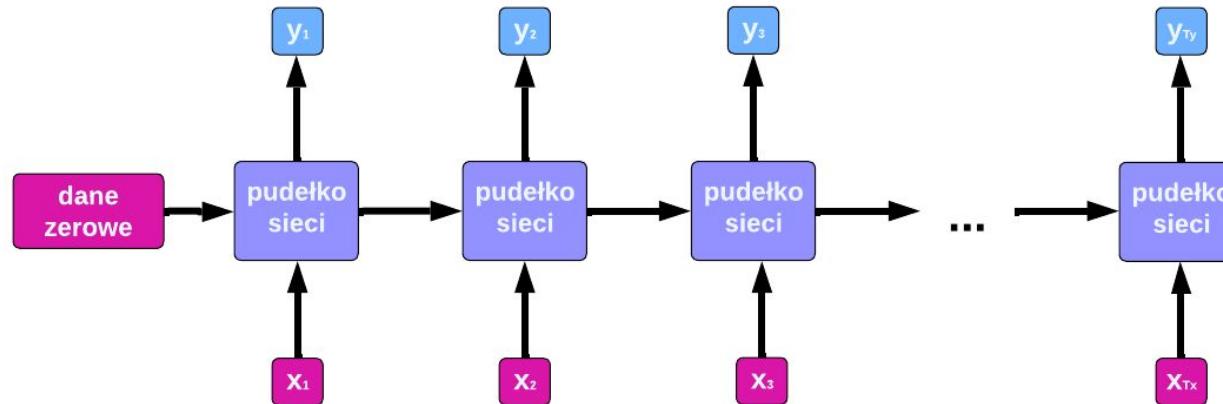
Sieć jeden do wielu

Sieć jeden-do-wielu



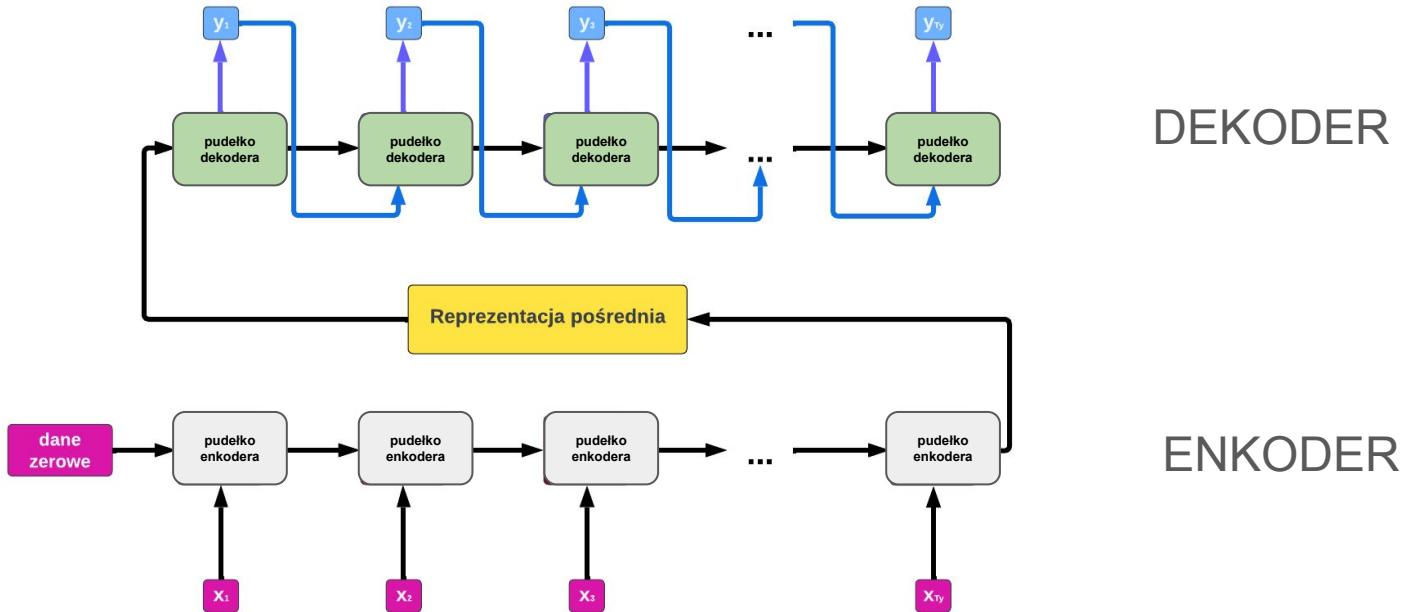
sieć wiele do wielu $Tx = Ty$

Sieć wiele-do-wielu $Tx = Ty$



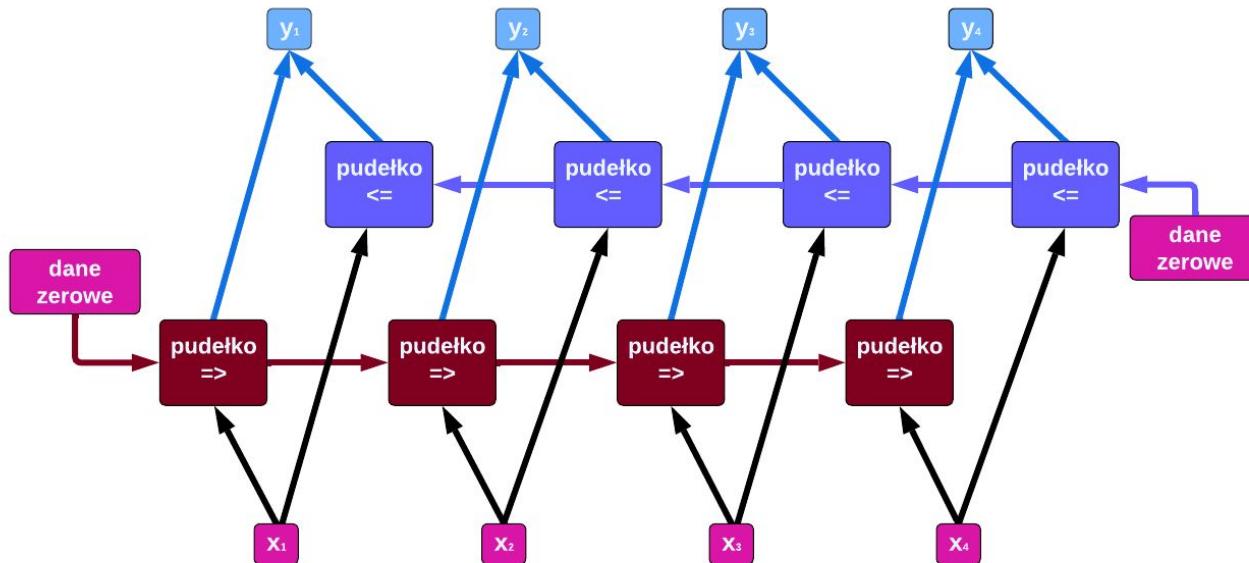
sieć wiele do wielu $T_x \neq T_y$

Sieć wiele-do-wielu, $T_x \neq T_y$ (enkoder - dekoder)



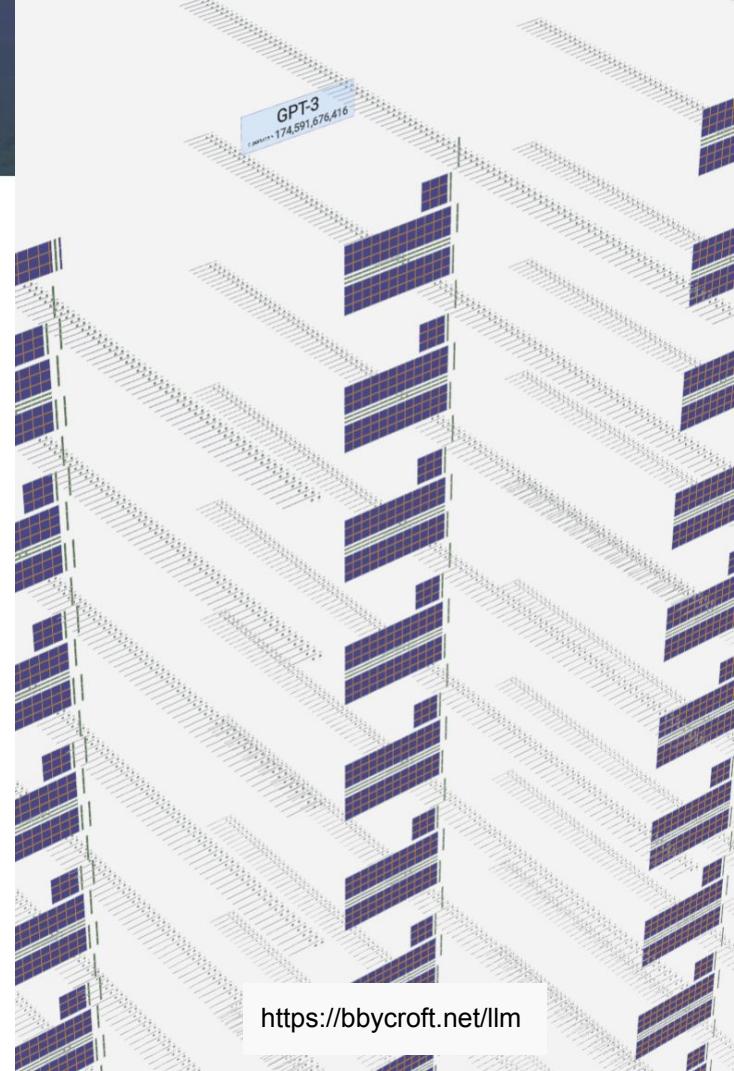
sieć wiele do wielu $T_x \neq T_y$

Sieć dwukierunkowa



Duży model tekstowy LLM

Duży model tekstowy to zaawansowana wersja modelu tekstowego, który został wytrenowany na wyjątkowo dużych zbiorach danych tekstowych i zazwyczaj składa się z miliardów lub nawet bilionów parametrów. Dzięki swojej skali, duże modele tekstowe mają zdolność do bardziej precyzyjnego rozumienia kontekstu, generowania bardziej naturalnych odpowiedzi i wykonywania zadań NLP na bardzo wysokim poziomie.

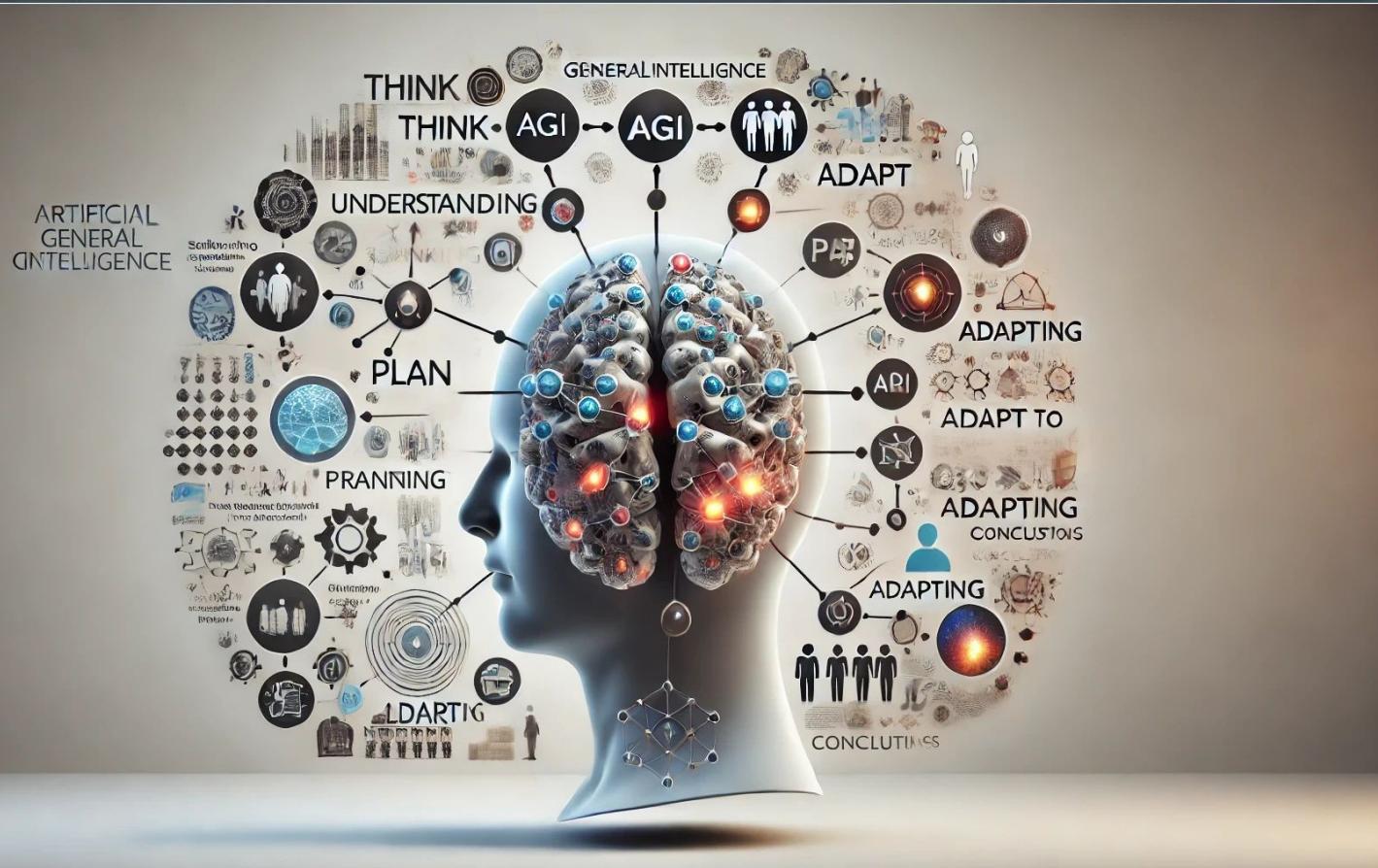


LLM vs AGI

Jakie cechy posiada Ogólna Sztuczna Inteligencja AGI?

Jakie są różnice między AGI, a LLM?

LLM vs AGI



"AGI similarity to
human mind"
by DALL-E

LLM vs AGI

Artificial General Intelligence AGI (Ogólna Sztuczna Inteligencja), odnosi się do rodzaju sztucznej inteligencji, która potrafi wykonywać dowolne zadania intelektualne, jakie może wykonać człowiek.

W przeciwieństwie do wyspecjalizowanej AI (ang. Narrow AI), która jest zaprojektowana do realizowania konkretnych zadań (np. rozpoznawania twarzy, grania w szachy), AGI miałaby zdolność uczenia się i stosowania wiedzy w szerokim zakresie dziedzin oraz rozwiązywania problemów w sposób uniwersalny.

W idealnym przypadku AGI potrafiłaby myśleć, rozumieć, planować, adaptować się do nowych sytuacji i kontekstów, a nawet wyciągać wnioski, podobnie jak ludzki umysł. To jest cel dalekosiązny w badaniach nad sztuczną inteligencją, **który jeszcze nie został osiągnięty**.

LLM vs AGI

Czy program opisany poniżej, oparty na aktualnie dostępnych LLM spełniałby założenia AGI?

1. Program generuje output na losowe źródła danych (np. text), dostarczane w losowych momentach.

Wynik przetwarzania LLM wraz z wejściem jest wykorzystywany do fine tuningu parametrów metodą PEFT.

2. Program może za pomocą outputu decydować o kolejnych akcjach np. zdobyciu dodatkowych źródeł wiedzy lub kontaktie z określoną osobą. Te wyniki także zostały niezwłocznie wykorzystane w fine tuningu.

3. Dodatkowo program wzbudzałby się w losowych momentach ponownie przetwarzając już wcześniej procesowane dane wraz ze swoją wcześniejszą odpowiedzią, ten wynik zaś byłby także wykorzystywany w fine tuningu.



by DALL-E

LLM vs AGI

Czy program opisany poniżej, oparty na aktualnie dostępnych LLM spełniałby założenia AGI?

1. *Program generuje output na losowe źródła danych (np. text), dostarczane w losowych momentach.*

Wynik przetwarzania LLM wraz z wejściem jest wykorzystywany do fine tuningu parametrów metodą PEFT.

...

Program opisany przez Ciebie wydaje się być zaawansowaną formą systemu opartego na sztucznej inteligencji, ale nie spełnia kryteriów AGI (Artificial General Intelligence), które oznaczają ogólną inteligencję o zdolnościach równoważnych lub przekraczających możliwości ludzkie w szerokim zakresie zadań.

Program, który opisujesz, jest bardziej zaawansowaną wersją obecnych modeli LLM (Large Language Models) z mechanizmami samouczenia i adaptacji, ale nadal mieści się w kategorii narrow AI. AGI wymagałoby czegoś znacznie bardziej uniwersalnego, z umiejętnością myślenia abstrakcyjnego, podejmowania decyzji o własnych celach i szerokiej adaptacji do zupełnie nowych, nieznanych wcześniej problemów.

źródło GPT-4o

LLM vs AGI

Zakres działania: Twój program, chociaż dynamiczny i adaptacyjny, działa w oparciu o losowe dane i procesy, ale nadal w określonym, zdefiniowanym przez twórcę zakresie. AGI, z definicji, musi być w stanie zrozumieć i rozwiązywać problemy w zupełnie nowych kontekstach, nie tylko adaptując się do danych, ale również ucząc się szeroko pojętych pojęć, zadań i dziedzin bez bezpośredniego trenowania na tych danych.

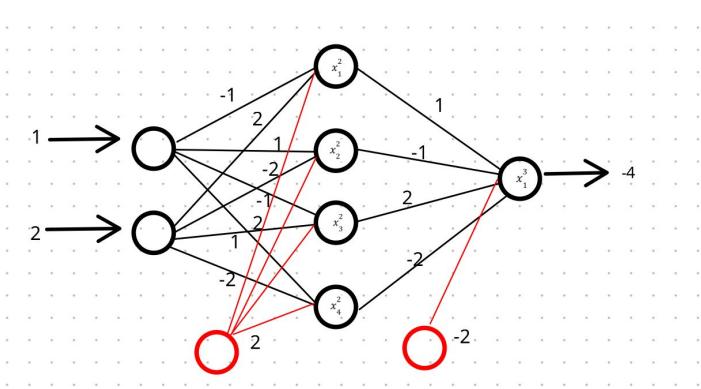
Fine-tuning z PEFT: Twój system korzysta z techniki fine-tuningu (np. PEFT), co jest typowe dla systemów narrow AI (Sztuczna Inteligencja o wąskim zakresie działania). Fine-tuning pozwala na przystosowanie modelu do specyficznych danych, ale nie daje mu pełnej autonomii ani zdolności do niezależnego formułowania i rozumienia abstrakcyjnych koncepcji, co jest kluczowe dla AGI.

Brak "świadomości" i intencji: Kluczowym aspektem AGI jest możliwość posiadania szerokiego "zrozumienia" kontekstu, celów, a także posiadania intencji w działaniu. Twój system podejmuje akcje, takie jak zdobycie dodatkowych źródeł wiedzy, ale te akcje wynikają z wprowadzonego schematu, a nie z autonomicznego zrozumienia świata czy złożonych potrzeb, które AGI musiałaby rozwijać i realizować.

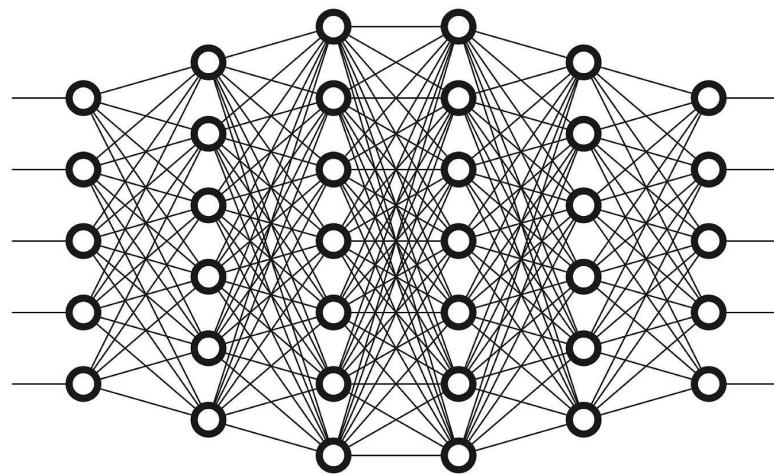
Brak rozumienia wielozadaniowości na poziomie człowieka: AGI potrafiłaby płynnie przechodzić między różnymi zadaniami, dziedzinami i umiejętnościami, rozumiejąc złożone relacje pomiędzy nimi. Twój program działa w oparciu o wcześniej zdefiniowane cele i algorytmy, co czyni go bardziej zaawansowanym narzędziem narrow AI, ale nie AGI.

Pojemność modelu

Pojemność modelu odnosi się do zdolności modelu uczenia maszynowego do uchwycenia i reprezentowania złożoności wzorców w danych. Jest to miara potencjalnej zdolności modelu do nauczenia się i zapamiętania zależności między danymi wejściowymi a wyjściowymi.



płytką sieć neuronową - model o małej pojemności
ile parametrów?



głęboka sieć neuronowa - model o "większej" pojemności
229 parametrów

Pojemność modelu



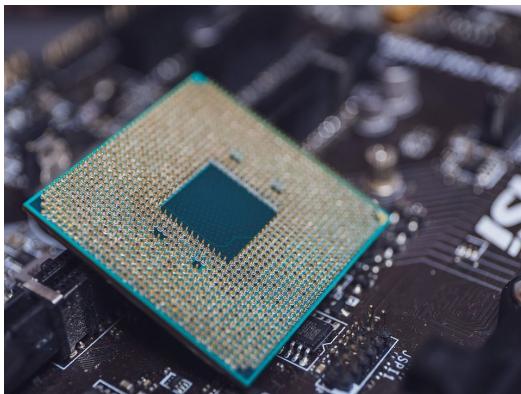
GPT3 - model o dużej pojemności
175 miliardów parametrów

Trenowanie dużych modeli

Do trenowania dużych modeli tekstowych wykorzystuje się dedykowane układy GPU (graphics processing unit) i TPU (tensor processing unit), które potrafią zrównoleglać obliczenia.

Nawet przy takiej optymalizacji trenowanie LLM wymaga tysięcy godzin (GPU/TPU).

źródło grafiki:
<https://pixabay.com/pl/photos/ryzen-cpu-%D0%B0%D0%BC%D0%B4-edytor-sz-t-%C5%BCeton-5250770/>



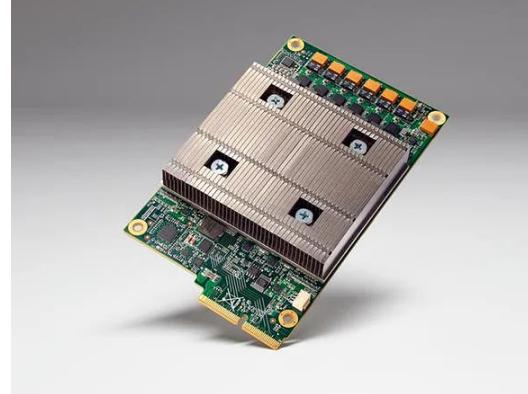
CPU

źródło grafiki:
<https://pixabay.com/pl/photos/nvidia-karta-graficzna-bitcoin-gpu-5264921/>



GPU

źródło grafiki: Google



TPU

CUDA - funkcje c++ z przetwarzaniem równoległym

specyfikator `_global_` wskazuje funkcję,
która działa na urządzeniu (GPU)

```
global void vector_add(float *out, float *a, float *b, int n) {  
    int index = 0;  
    int stride = 1  
    for(int i = index; i < n; i += stride) {  
        out[i] = a[i] + b[i];  
    }  
}
```

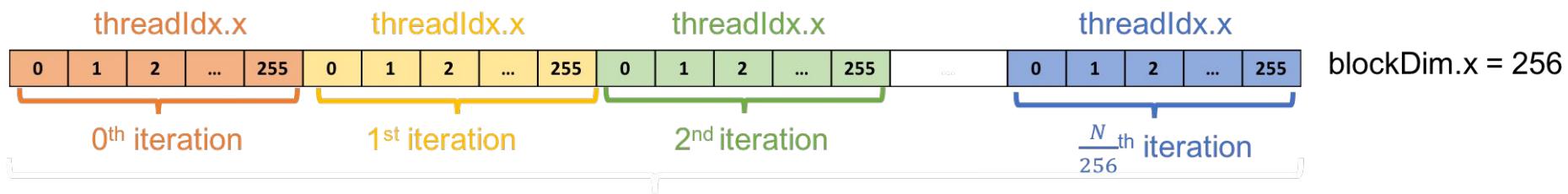
piętla zostanie podzielona na mniejsze
równoległe pętle dla podzakresów

```
vector_add <<< 1 , 256 >>> (d_out, d_a, d_b, N);
```

liczba wątków

liczba równoległych
zadań w wątku

CUDA - funkcje c++ z przetwarzaniem równoległy



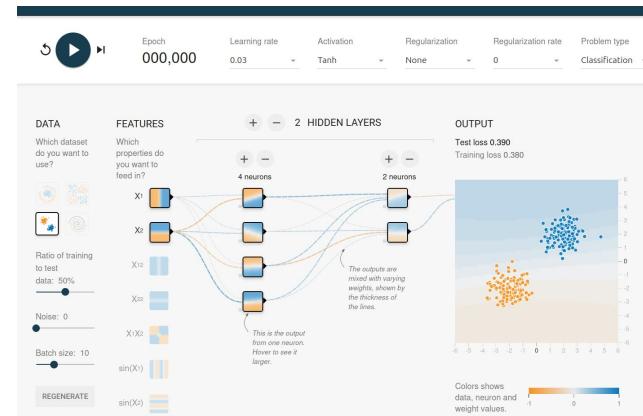
Pojemność modelu

Zadanie:

Skorzystaj z symulatora <https://playground.tensorflow.org>

1. Ile minimalnie neuronów w warstwie ukrytej potrzeba, aby model był w stanie odseparować każdy z 4 wariantów rozkładu punktów.
2. Ile minimalnie neuronów w warstwie ukrytej oraz wyjściowej potrzeba, aby model był w stanie odseparować rozkład z obrazu poniżej.

Zauważ, że wraz ze wzrostem liczby neuronów wzrasta liczba parametrów modelu oraz jego pojemność, bo jest w stanie reprezentować trudniejsze wzorce.



Pojemność modelu

LLM można interpretować jako wiele małych modeli o różnych zastosowaniach połączonych ze sobą w jedną dużą sieć.



Generuje dobre odpowiedzi na pytania wiedzy ogólnej, ale może mieć ograniczenia w odpowiadaniu na pytania eksperckie.

Kontekst

Kontekst w dużym modelu językowym (LLM, Large Language Model) odnosi się do informacji zawartych w sekwencji tekstu, na podstawie której model generuje odpowiedzi lub dokonuje predykcji. W przypadku LLM, takich jak GPT-3 czy GPT-4, kontekst obejmuje całość lub część tekstu, który model przetwarza, zanim wygeneruje kolejną odpowiedź lub przewiduje następne słowo w ciągu.

W praktyce kontekst LLM jest kluczowy dla dokładności i relevantności generowanych odpowiedzi, a jego odpowiednie zarządzanie (np. poprzez dostarczenie odpowiedniego fragmentu tekstu) jest niezbędne do uzyskania optymalnych rezultatów w zadaniach NLP.

Kontekst

- **Kontekst wejściowy:**

To tekst dostarczony jako dane wejściowe do modelu. Może to być jedno zdanie, akapit lub cała rozmowa, którą model analizuje, aby wygenerować odpowiedź. Kontekst ten pomaga modelowi zrozumieć, na co powinien odpowiedzieć i jakie informacje są istotne.

- **Okno kontekstowe:**

Modele LLM mają ograniczone "okno kontekstowe", co oznacza, że mogą przetwarzać tylko określoną liczbę tokenów (czyli fragmentów tekstu, takich jak słowa czy znaki) naraz. Wszystko, co wykracza poza to okno, może zostać pominięte lub mieć mniejszy wpływ na generowaną odpowiedź.

- **Zachowanie spójności i ciągłości:**

W ramach kontekstu model stara się utrzymać spójność tematyczną i logiczną w generowanych odpowiedziach. Im więcej istotnych informacji dostarczy się w kontekście, tym bardziej adekwatna i trafna będzie odpowiedź modelu.

- **Kontekst semantyczny:**

Model wykorzystuje semantyczne powiązania między słowami w kontekście, aby generować odpowiedzi, które są logiczne i zgodne z oczekiwaniemi użytkownika.

Kontekst

- **Liczba słów:**

Odnosi się do ilości pełnych słów w tekście. Słowa są jednostkami języka, które mają znaczenie w naturalnym języku, takie jak "dom", "jest", "piękny".

- **Liczba tokenów:**

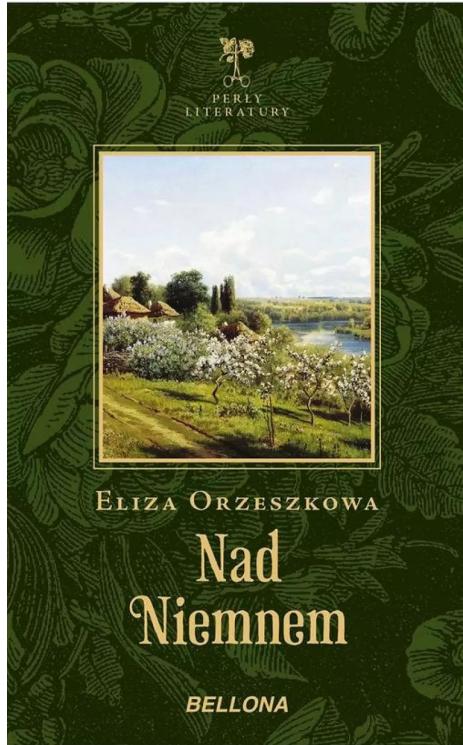
Tokeny to jednostki tekstowe, które są używane przez model językowy do przetwarzania tekstu. Tokeny mogą być całymi słowami, częściami słów, a nawet pojedynczymi znakami, w zależności od sposobu tokenizacji.

"Ada has a cat, but John has a dragon."

- **Liczba słów:** 9 (Ada, has, a, cat, but, John, has, a, dragon)

- **Liczba tokenów:** 11 (Ada, has, a, cat, ",", but, John, has, a, dragon, ".")

Kontekst



Czy kontekst LLM pomieści wybitne dzieło
Elizy Orzeszkowej? (622 strony)

Rozmiar modelu



GPT4o

API OPENAI

175 mld parametrów
kontekst 128k tokenów



MIXTRAL

API / STANDALONE

45 mld parametrów
kontekst 32k tokenów



LLAMA3

API / STANDALONE

70 mld parametrów
kontekst 8.2k tokenów



CLAUDE 3

API AWS BEDROCK

2 tryliony parametrów
kontekst 200k tokenów

Rozmiar modelu

Modele tekstowe występują w różnych rozmiarach:

- small
- large
- XXL
- ...

Załadowanie całego LLM do pamięci podręcznej wymaga dziesiątek GB RAM

Wydajne przetwarzanie milionów operacji wymaga zrównoleglenia za pomocą wydajnych układów GPU i TPU.

Category	Requirement	Details
Hardware Requirements	Processor and Memory	<ul style="list-style-type: none">CPU: Modern CPU with at least 8 cores recommended for efficient backend operations and data preprocessing.GPU: One or more powerful GPUs, preferably Nvidia with CUDA architecture, recommended for model training and inference. RTX 3000 series or higher is ideal.RAM: Minimum 16 GB for 8B model and 32 GB or more for 70B model.
	Storage	Disk Space: Sufficient storage to host the model and associated datasets. For large models like the 70B, several terabytes of SSD storage recommended for fast data access.

<https://llamaimodel.com/requirements/>

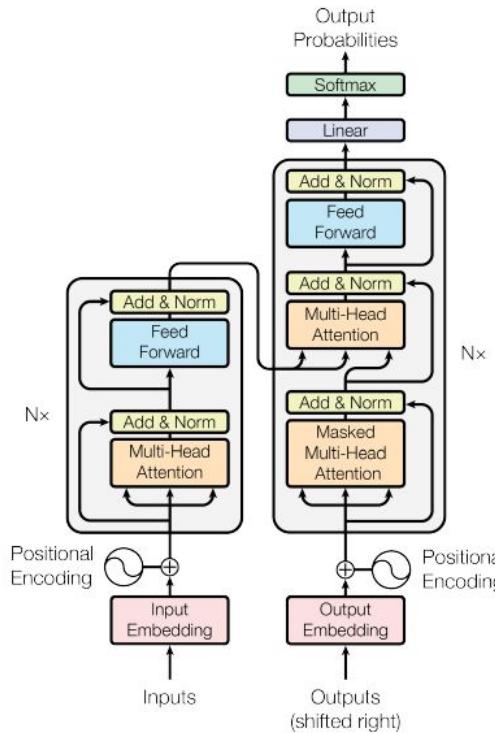
Models 41	Sort: Recently updated
meta-llama/Meta-Llama-3-70B-Instruct	Text Generation · Updated 11 days ago · 303k · 1.37k
meta-llama/Llama-Guard-3-8B	Text Generation · Updated 11 days ago · 126k · 86
meta-llama/Meta-Llama-3.1-405B-Instruct-FP8	Text Generation · Updated 11 days ago · 64.9k · 156
meta-llama/Meta-Llama-3.1-405B-Instruct	Text Generation · Updated 11 days ago · 48.3k · 428
meta-llama/Meta-Llama-3.1-70B-Instruct	Text Generation · Updated 11 days ago · 782k · 414
meta-llama/Meta-Llama-3.1-8B-Instruct	Text Generation · Updated 11 days ago · 2.81M · 2.13k
meta-llama/Meta-Llama-3.1-405B-FP8	Text Generation · Updated 17 days ago · 395k · 91
meta-llama/Meta-Llama-3.1-405B	Text Generation · Updated 22 days ago · 151k · 756
meta-llama/Llama-Guard-3-8B-INT8	Text Generation · Updated 24 days ago · 2.62k · 27
meta-llama/Prompt-Guard-86M	Text Classification · Updated Jul 25 · 526k · 162

<https://huggingface.co/meta-llama>

Rodziny LLM

- GPT - Generative pre-trained transformer, tylko dekoder, GPT-4 rozumie różne typy danych
- Mixtral by Mistral, sukces nieznaczącego startupu, tylko 7 miliardów parametrów
- Claude by Anthropic, bardzo duży kontekst
- Llama by Meta, duży nacisk na bezpieczeństwo
- Flan-T5, instruction tuning, niewielki model dający możliwość tuningu, (Alpaca = Llama with fine tuning)
- Phi by Microsoft
- Gemini by Google
- Bielik - gorszy od wiodących modeli, lepiej radzi sobie z j. polskim i zagadnieniami związanymi ze źródłami w j. polskim

Architektura LLM



From “Attention is all you need” paper by Vaswani, et al., 2017 [1]

Enkoder - dekoder

Transformery składają się z warstw **enkoderów i dekoderów**.

Zadaniem **enkodera** (po lewej stronie architektury Transformer) jest odwzorowanie sekwencji wejściowej na sekwencję ciągły reprezentacji (sygnał / kod / sekwencję), która jest następnie wprowadzana do dekodera.

Dekoder w prawej połowie architektury odbiera sygnał wyjściowy kodera wraz z sygnałem wyjściowym dekodera w poprzednim kroku czasowym w celu wygenerowania sekwencji wyjściowej. Na każdym kroku model jest autoregresywny, zużywając poprzednio wygenerowane symbole jako dodatkowe dane wejściowe podczas generowania następnego.

Kodowanie - worek słów

zbiór tekstów:

Ada ma komputer.

Uczenie maszynowe pozwala uczyć komputer.

Do rozwiązania problemów wystarczy komputer oraz duża liczba danych.

słownik:

[ada, ma, komputer, uczenie, maszynowe, pozwala, uczyć, do, rozwiązania, problemów, wystarczy, duża, liczba, danych]

zakodowane teksty:

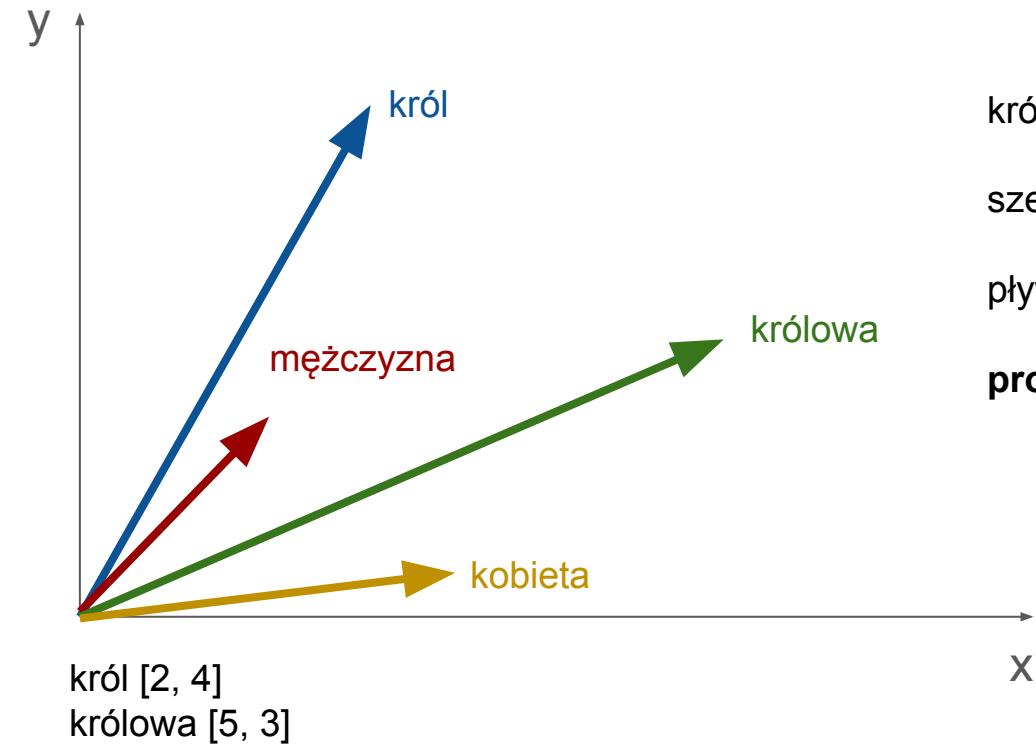
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]

[0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1]

Nie zachowuje informacji o znaczeniu słów
i relacjach pomiędzy nimi.

Embedding



król - mężczyzna \approx królowa

szedłem - idę \approx biegłem - biegnę

pływamy - my \approx pływam - ja

projekt + deadline \approx stackoverflow + copy/paste

Kodowanie tekstu w bazie do postaci wektorowej

```
1 model['go']
```

```
array([-0.078894,  0.4616 ,  0.57779 , -0.71637 , -0.13121 ,  0.4186 ,  
       -0.29156 ,  0.52006 ,  0.089986, -0.35062 ,  0.51755 ,  0.51998 ,  
       0.15218 ,  0.41485 , -0.12377 , -0.37222 ,  0.0273 ,  0.75673 ,  
      -0.8739 ,  0.58935 ,  0.46662 ,  0.62918 ,  0.092603, -0.012868,  
      -0.015169,  0.25567 , -0.43025 , -0.77668 ,  0.71449 , -0.3834 ,  
      -0.69638 ,  0.23522 ,  0.11396 ,  0.02778 ,  0.071357,  0.87409 ,  
      -0.1281 ,  0.063576,  0.067867, -0.50181 , -0.28523 , -0.072536,  
      -0.50738 , -0.6914 , -0.53579 , -0.11361 , -0.38234 , -0.12414 ,  
      0.011214, -1.1622 ,  0.037057, -0.18495 ,  0.01416 ,  0.87193 ,  
      -0.097309, -2.3565 , -0.14554 ,  0.28275 ,  2.0053 ,  0.23439 ,  
      -0.38298 ,  0.69539 , -0.44916 , -0.094157,  0.90527 ,  0.65764 ,  
      0.27628 ,  0.30688 , -0.57781 , -0.22987 , -0.083043, -0.57236 ,  
      -0.299 , -0.81112 ,  0.039752, -0.05681 , -0.48879 , -0.18091 ,  
      -0.28152 , -0.20559 ,  0.4932 , -0.033999, -0.53139 , -0.28297 ,  
     -1.4475 , -0.18685 ,  0.091177,  0.11454 , -0.28168 , -0.33565 ,  
     -0.31663 , -0.1089 ,  0.10111 , -0.23737 , -0.64955 , -0.268 ,  
      0.35096 ,  0.26352 ,  0.59397 ,  0.26741 ], dtype=float32)
```

Kodowanie tekstu w bazie do postaci wektorowej

```
1 model['away']
```

```
array([-0.10379 , -0.014792,  0.59933 , -0.51316 , -0.036463,  0.6588 ,
       -0.57906 ,  0.17819 ,  0.23663 , -0.21384 ,  0.55339 ,  0.53597 ,
       0.041444,  0.16095 ,  0.017093, -0.37242 ,  0.017974,  0.39268 ,
      -0.23265 ,  0.1818 ,  0.66405 ,  0.98163 ,  0.42339 ,  0.030581,
       0.35015 ,  0.25519 , -0.71182 , -0.42184 ,  0.13068 , -0.47452 ,
      -0.08175 ,  0.1574 , -0.13262 ,  0.22679 , -0.16885 , -0.11122 ,
      -0.32272 , -0.020978, -0.43345 ,  0.172 , -0.67366 , -0.79052 ,
       0.10556 , -0.4219 , -0.12385 , -0.063486, -0.17843 ,  0.56359 ,
       0.16986 , -0.17804 ,  0.13956 , -0.20169 ,  0.078985,  1.4497 ,
       0.23556 , -2.6014 , -0.5286 , -0.11636 ,  1.7184 ,  0.33254 ,
       0.12136 ,  1.1602 , -0.2914 ,  0.47125 ,  0.41869 ,  0.35271 ,
       0.47869 , -0.042281, -0.18294 ,  0.1796 , -0.24431 , -0.34042 ,
       0.20337 , -0.93676 ,  0.013077,  0.080339, -0.36604 , -0.44005 ,
      -0.35393 ,  0.15907 ,  0.55807 ,  0.1492 , -0.86433 ,  0.040305,
      -1.0939 , -0.26386 , -0.29494 ,  0.25696 , -0.33718 , -0.086468,
      -0.24246 , -0.21114 ,  0.099632,  0.12815 , -0.78714 , -0.51785 ,
      -0.10944 ,  0.9763 ,  0.57032 ,  0.13581 ], dtype=float32)
```

Kodowanie tekstu w bazie do postaci wektorowej

```
1 (model['go'] + model['away'])/2
```

```
array([-0.091342,  0.223404,  0.58856 , -0.614765, -0.0838365 ,  
       0.5387 , -0.43531 ,  0.349125,  0.163308, -0.28223 ,  
       0.53547 ,  0.52797496,  0.096812,  0.2879 , -0.0533385 ,  
      -0.37232 ,  0.022637 ,  0.574705, -0.553275,  0.385575 ,  
       0.565335,  0.805405 ,  0.2579965,  0.0088565,  0.1674905 ,  
       0.25543 , -0.571035, -0.59926 ,  0.422585, -0.42896 ,  
      -0.389065,  0.19631 , -0.00933 ,  0.127285, -0.0487465 ,  
       0.381435, -0.22540998,  0.021299, -0.1827915, -0.16490501,  
      -0.47944498, -0.431528 , -0.20091 , -0.55665 , -0.32982 ,  
      -0.088548, -0.28038502,  0.219725,  0.090537, -0.67012 ,  
       0.0883085, -0.19332 ,  0.0465725,  1.160815,  0.0691255 ,  
      -2.47895 , -0.33707 ,  0.083195,  1.86185 ,  0.283465 ,  
      -0.13081 ,  0.927795 , -0.37028 ,  0.1885465,  0.66198 ,  
       0.505175,  0.37748498,  0.1322995, -0.380375, -0.025135 ,  
      -0.1636765, -0.45639 , -0.047815, -0.87394 ,  0.0264145 ,  
       0.0117645, -0.427415 , -0.31048 , -0.317725, -0.02326 ,  
       0.525635,  0.05760051, -0.69786 , -0.1213325, -1.2707 ,  
      -0.225355, -0.1018815 ,  0.18575001, -0.30943 , -0.211059 ,  
      -0.279545, -0.16002001,  0.100371 , -0.05461 , -0.71834505 ,  
      -0.392925,  0.12075999,  0.61991 ,  0.582145 ,  0.20161 ],  
      dtype=float32)
```

Kodowanie pozycyjne



zapis odległości (numer słowa w sekwencji):

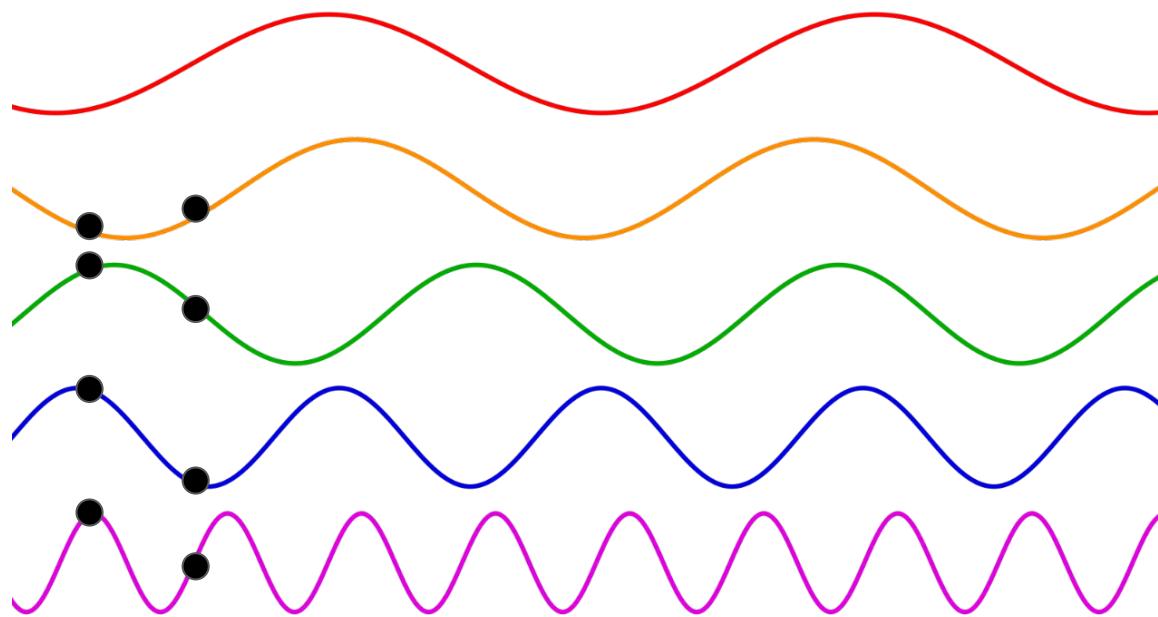
● 30

● 100

● 150

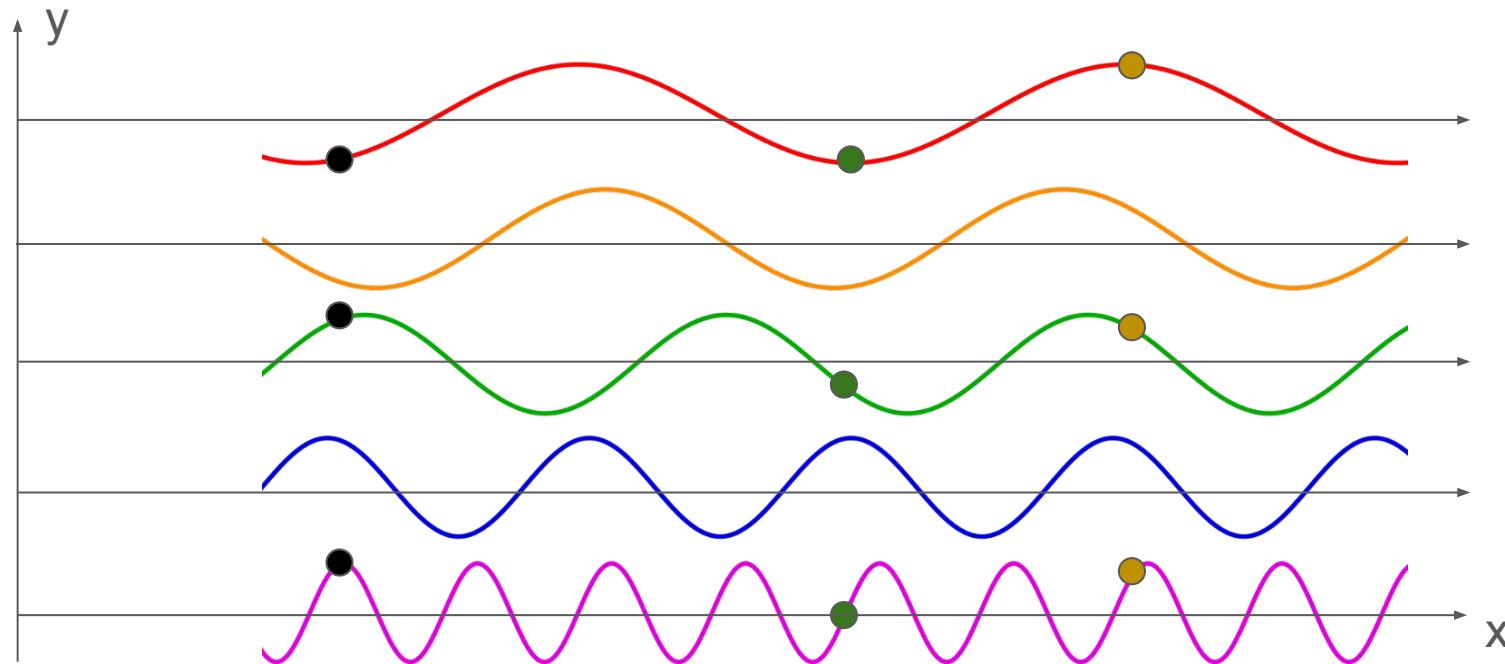
brak regularyzacji (duże wartości, różna skala)

Kodowanie pozycyjne



- pozycje 2 słów na fali sinusoidalnej o różnej częstotliwości

Kodowanie pozycyjne



zapis wartości sinusa:

- -0.9, 0.9, 1.0
- -1.0, -0.4, 0.0
- 0.95, 0.0, 0.9

pozycja każdego słowa zapisana za pomocą wartości z przedziału (-1, 1)

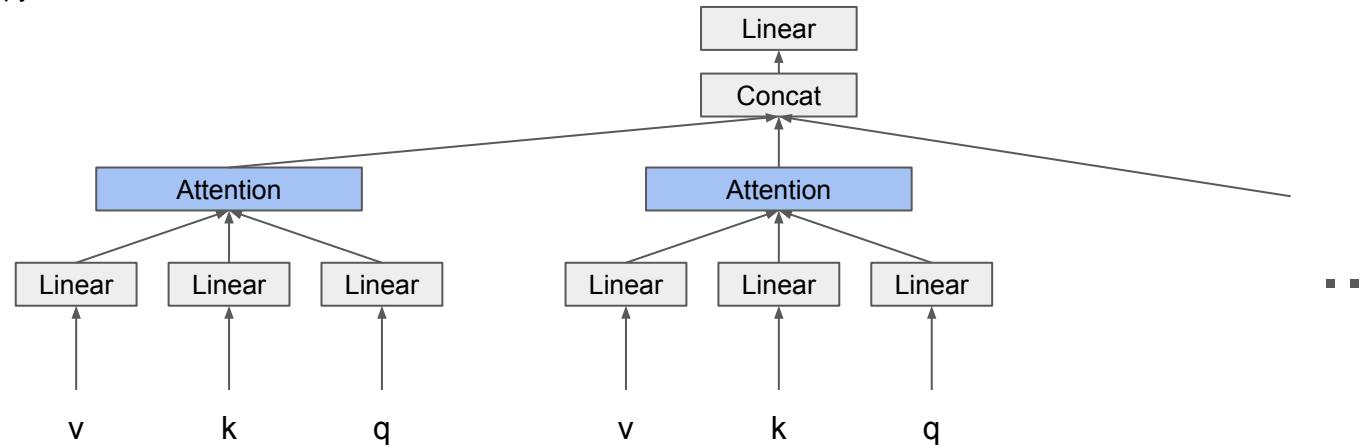
Multi Head attention

Mechanizm multi-head attention pozwala modelowi na przetwarzanie informacji w sposób bardziej efektywny i zróżnicowany. Multi-head attention umożliwia modelowi równoczesne „zwracanie uwagi” na różne aspekty danych wejściowych, co pomaga lepiej zrozumieć kontekst i relacje między słowami czy tokenami.

Każde wejście (np. token w zdaniu) jest reprezentowane przez wektor, a te wektory są przekształcane w trzy różne zestawy wektorów:

- zapytania (**queries**)
- klucze (**keys**)
- wartości (**values**)

Zapytanie (query) reprezentuje element, na który chcemy zwrócić uwagę, a **klucz** (key) i **wartość** (value) to elementy danych, które są porównywane z zapytaniem.



Multi Head attention

Analogia w programowaniu:

Lookup table:

```
table = { "key0": "value0", "key1": "value1", ... }
```

Query Process:

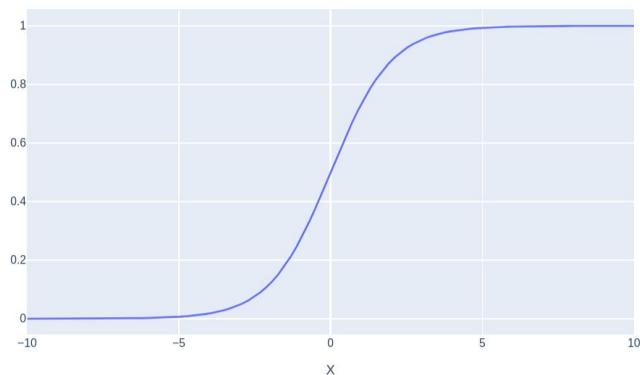
```
table["key1"] => "value1"
```

Multi Head attention

Obliczanie attention:

- Każde zapytanie (Q) jest porównywany z wszystkimi kluczami (K), obliczając ich podobieństwo. Wynikiem tego jest tzw. **skala wagowa** (attention score), która mówi, jak ważny jest dany klucz w kontekście zapytania.
- Podobieństwo oblicza się zwykle jako iloczyn skalarny wektorów zapytania i klucza. Wynik ten jest następnie normalizowany (np. za pomocą funkcji softmax), aby uzyskać prawdopodobieństwo wskazujące na istotność każdego klucza.
- Otrzymane wagi są używane do skalowania wartości (V). Zsumowane wartości dają wynik attention, który jest reprezentacją informacji, na którą model zwraca uwagę.

softmax



Multi Head attention

multi-head

1. Zamiast wykonywać pojedyncze obliczenie attention, multi-head attention dzieli wektory Q, K i V na kilka mniejszych części (zwanych „głowami”). Każda głowa oblicza swoją wersję attention, co pozwala modelowi skupić się na różnych aspektach danych jednocześnie.
2. Na końcu wszystkie te niezależne wyniki z różnych głów są łączone i przekształcane w jedną, finalną reprezentację.

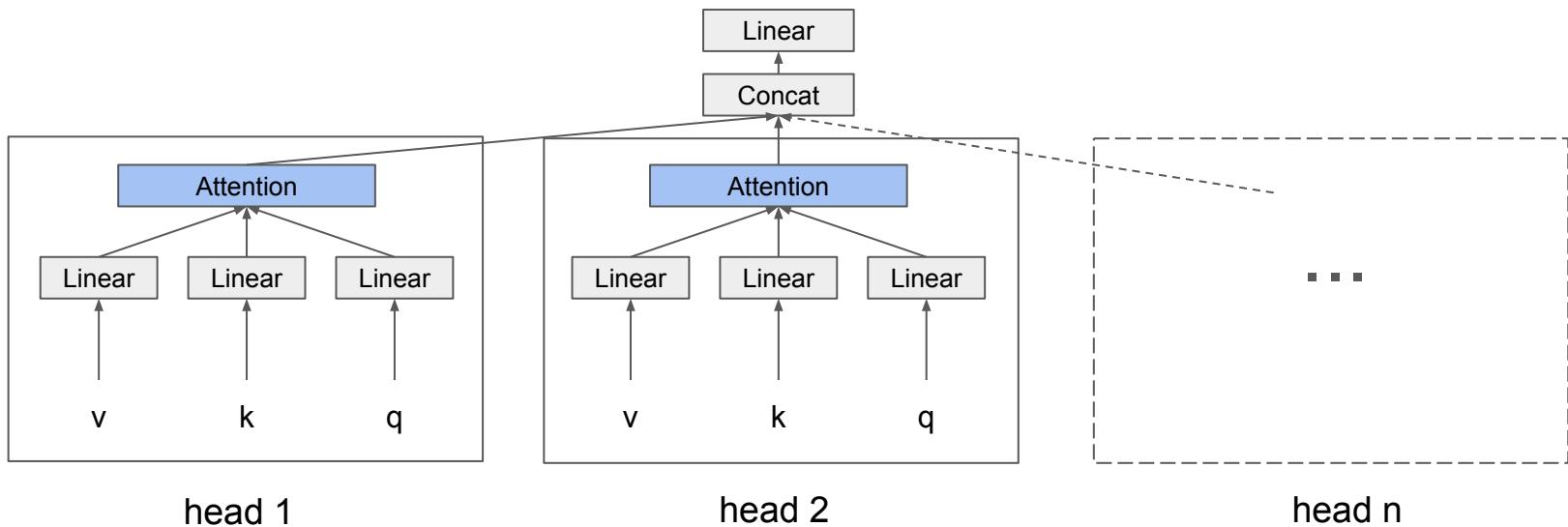
Dzięki temu, że model ma wiele głów (np. 8, 12, 16 głów w różnych implementacjach), może analizować różne relacje i wzorce między słowami w zdaniu. Każda głowa koncentruje się na innych relacjach, np. jedna głowa może skupić się na relacjach krótkodystansowych między sąsiadującymi słowami, a inna na relacjach między słowami odległymi od siebie.

kluczowe zalety mechanizmu multi-head attention:

- **Równoczesne przetwarzanie różnych informacji:** Model może analizować różne aspekty wejścia w sposób równoległy.
- **Lepsze rozumienie kontekstu:** Dzięki wielu głowom mechanizm ten jest w stanie uchwycić bardziej złożone relacje między elementami danych.
- **Zwiększenie pojemności modelu:** Dzięki multi-head, model jest w stanie reprezentować więcej informacji i uczyć się bardziej złożonych wzorców.

W praktyce multi-head attention jest używany w różnych warstwach modeli typu Transformer (takich jak BERT, GPT), co sprawia, że modele te są tak efektywne w zadaniach przetwarzania języka naturalnego (NLP).

Multi Head attention



Add and normalization

Mechanizm **Add and Normalize** wspomaga stabilność i efektywność uczenia się modelu. Składa się z dwóch kluczowych komponentów:

1. Add (Dodawanie):

- Po zastosowaniu operacji attention lub warstwy feed-forward, wynik tej operacji jest dodawany do oryginalnego wejścia za pomocą mechanizmu znanego jako **residual connection** (łącze resztkowe).
- Dodanie to jest kluczowe, ponieważ pozwala na zachowanie części oryginalnych informacji z wejścia, co przeciwdziała problemowi „zanikania gradientów” i pomaga modelowi unikać utraty istotnych informacji podczas głębokiego uczenia.
- **Residual connection** dodaje oryginalne dane wejściowe do wyjścia z kolejnych warstw, co ułatwia modelowi uczenie się nawet w bardzo głębokich strukturach.

$$\text{Output} = \text{Layer Output} + \text{Input}$$

Add and normalization

2. Normalization (Normalizacja):

- Następnie wynik dodawania jest normalizowany, aby ustabilizować i ułatwić proces uczenia modelu. W Transformerze wykorzystuje się warstwę normalizacji, która różni się od standardowej batch normalization.
- **Layer normalization** działa na poziomie pojedynczych warstw, normalizując każdą aktywację względem wartości średniej i odchylenia standardowego dla danej warstwy. Pozwala to na stabilniejsze uczenie modelu, niezależnie od rozkładu danych wejściowych.
- Dzięki normalizacji wartości wyjściowe mają bardziej przewidywalny zakres, co pomaga w zapobieganiu problemom z eksplodującymi lub zanikającymi gradientami.

Gdzie:

$$\text{Normalized Output} = \frac{\text{Output} - \mu}{\sigma + \epsilon}$$

- μ to średnia z wyjść warstwy,
- σ to odchylenie standardowe,
- ϵ to mała wartość dodawana dla stabilności numerycznej.

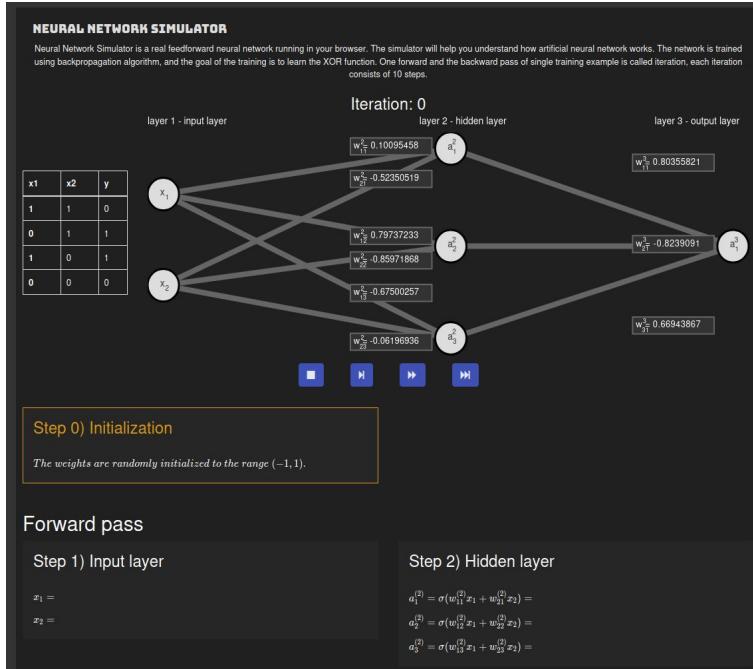
Add and normalization

Residual connections (Add) sprawiają, że model może łatwiej uczyć się funkcji tożsamości (identity mappings), co umożliwia przepływ oryginalnych danych w głąb sieci, co zaś ułatwia trenowanie głębokich sieci.

Layer normalization zapewnia, że wartości w różnych warstwach są bardziej stabilne, co przyspiesza konwergencję podczas procesu uczenia się i pozwala na stabilniejsze aktualizacje wag.

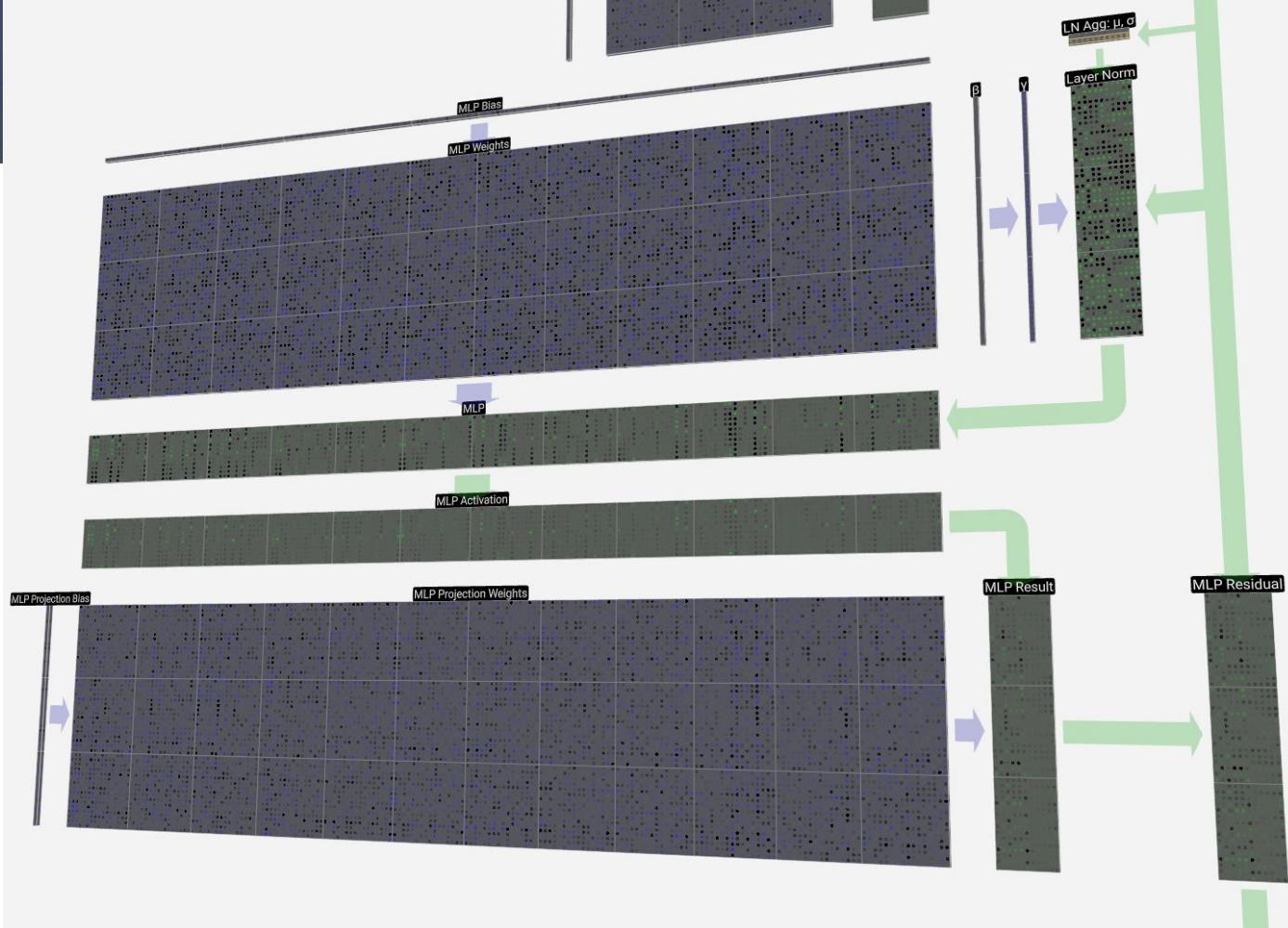
W skrócie, mechanizm **Add and Normalize** pozwala na stabilne i efektywne trenowanie głębokich modeli, takich jak Transformer, umożliwiając lepsze przenoszenie informacji między warstwami i zapobiegając problemom z gradientami.

Feed forward



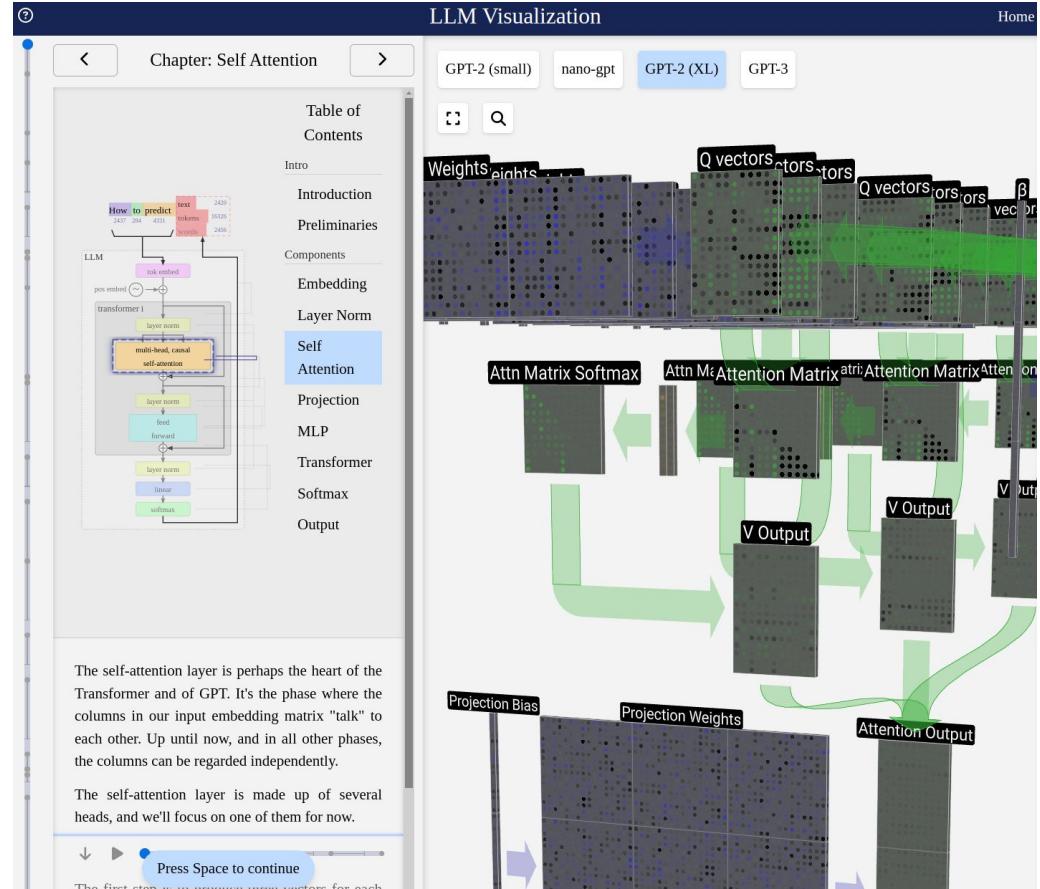
<https://www.mladdict.com/neural-network-simulator>

Feed forward



Zadanie

Uruchom każdy z etapów wizualizacji
przetwarzania danych przez LLM
bbycroft.net



<https://bbycroft.net/llm>