

Transformery (parametry, warianty atencji, trenowanie)

Paweł Rychlikowski

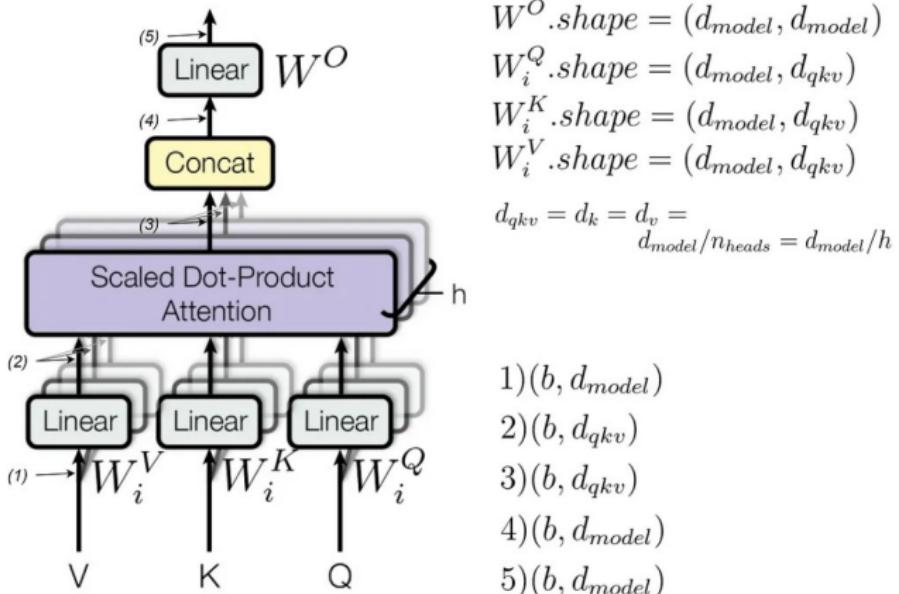
Instytut Informatyki UW

18 grudnia 2024

Literatura do przedmiotu (cd)

- Bardziej zorientowaną na programistów książką jest **Przetwarzanie języka naturalnego z wykorzystaniem transformerów**, Tunstral, von Werra, Wolf (Helion, O'Reilly)
- Przegląd aktualnych prac z AI: <https://www.marktechpost.com/>
- Dwie kategorie, które są najwyżej:
 - ▶ LLMs (pierwsze miejsce)
 - ▶ Small Language Models (drugie miejsce)

Liczba parametrów transformera (atencja)



Transformer multi-head attention. Adapted from figure 2 from the [public domain paper](#)

- Wektory przetwarzane przez transformera grupowane są we wsadach (batch), liczba wsadów to b
- (...)

Liczba parametrów transformera (atencja)

Obliczenia liczby parametrów

$$N_{attention} = \overbrace{(d_{model} * d_{model} + d_{model})}^{W^O} + \overbrace{\underbrace{(d_{model} * d_{qkv} + d_{qkv}) * n_{heads} * 3}_{\text{II}}}^{\substack{W_i^Q, W_i^K, \text{ or } W_i^V \\ \text{Three matrices for every head}}} =$$
$$(d_{model} * d_{model} + d_{model}) + \overbrace{(d_{model} * d_{model} + d_{model}) * 3}^{(d_{model} * d_{model} + d_{model}) * 3} =$$
$$(d_{model} * d_{model} + d_{model}) * 4 = \underbrace{(d_{model}^2 + d_{model}) * 4}_{\text{The exact formula}} \approx \underbrace{4 * d_{model}^2}_{\text{The approximate formula}}$$

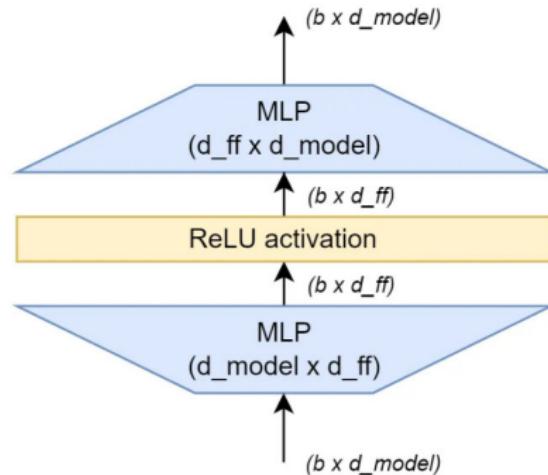
Why $(d_{model} * d_{qkv} + d_{qkv}) * n_{heads} = (d_{model} * d_{model} + d_{model})$:

$$(d_{model} * d_{qkv} + d_{qkv}) * n_{heads} =$$
$$\underbrace{d_{model} * d_{qkv} * n_{heads}}_{\substack{d_{model}, \\ \text{since } d_{qkv} = d_{model}/n_{heads}}} + \underbrace{d_{qkv} * n_{heads}}_{d_{model}} = d_{model} * d_{model} + d_{model}$$

The formula for calculating the number of parameters in the Transformer attention module. Image by Author

Źródło: <https://towardsdatascience.com/how-to-estimate-the-number-of-parameters-in-transformer-models-ca01>

Liczba parametrów transformera (feedforward)



Transformer feed-forward network. Image by Author

$$N_{feedforward} = \overbrace{(d_{model} * d_{ff} + d_{ff})}^{\text{The first linear layer}} + \overbrace{(d_{ff} * d_{model} + d_{model})}^{\text{The second linear layer}} = \\ d_{model} * d_{ff} + d_{ff} * d_{model} + d_{model} + d_{ff} = \\ \underbrace{2 * d_{model} * d_{ff} + d_{model} + d_{ff}}_{\text{The exact formula}} \approx \underbrace{2 * d_{model} * d_{ff}}_{\text{The approximate formula}}$$

The formula for calculating the number of parameters in the Transformer feed-forward net. Image by Author

Liczba parametrów transformera (feedforward)

- LayerNormalization ma X parametrów (ile?)
- Jeżeli w słowniku mamy V elementów, to dochodzi to tego $d \times V$ osadzeń słów
- Parametry sieci FF i atencji dotyczą jednej warstwy, trzeba je zatem pomnożyć przez liczbę warstw.

Ostateczny wzór (w przybliżeniu)

$$6d^2 \times L + |V| \times d$$

gdy $d_{\text{ff}} = d$

Anatomia papugi

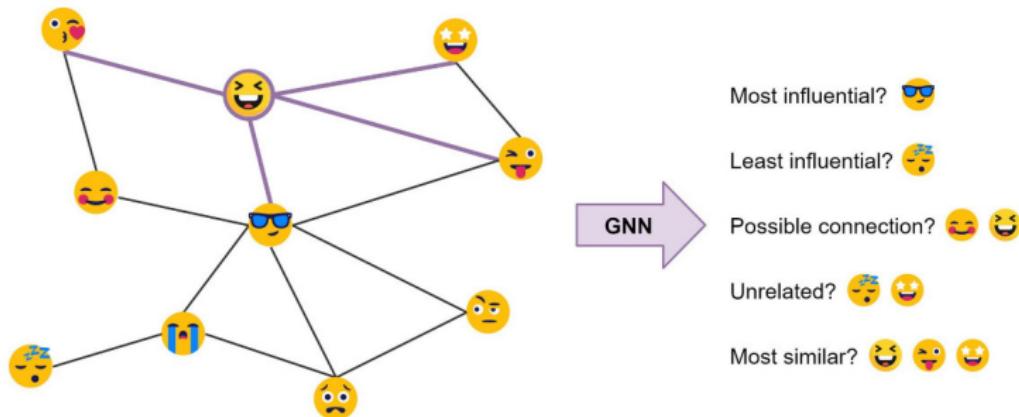


Zobaczmy, jak wygląda model papugi (w notatniku)

Transformery to sieci grafowe

Definicja

Neuronowe sieci grafowe (GNN) to sieci, dla których daną wejściową jest graf (na ogół skierowany)

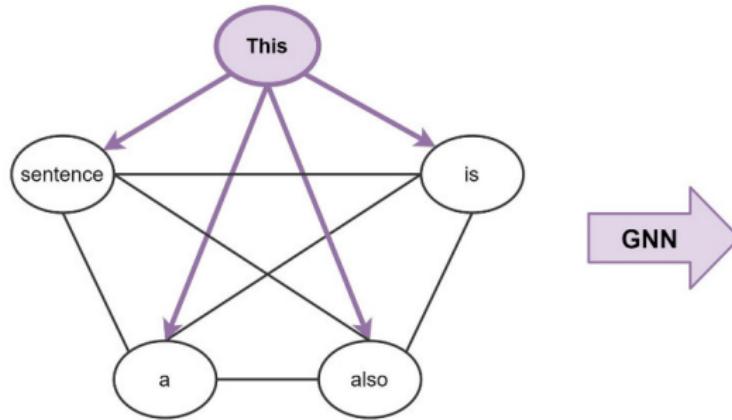


Take the example of this emoji social network: The node features produced by the GNN can be used for predictive tasks such as identifying the most influential members or proposing potential connections.

Źródło:

<https://thegradient.pub/transformers-are-graph-neural-networks/>

Transformery to sieci grafowe



Uwaga

Zdanie możemy widzieć jako graf w którym węzłami są słowa/tokeny

Transformery to sieci grafowe

$$\mathbf{A} = \text{softmax} \left(\text{mask} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \right) \right) \mathbf{V} \quad (11.1)$$

q ₁ ·k ₁	-∞	-∞	-∞
q ₂ ·k ₁	q ₂ ·k ₂	-∞	-∞
q ₃ ·k ₁	q ₃ ·k ₂	q ₃ ·k ₃	-∞
q ₄ ·k ₁	q ₄ ·k ₂	q ₄ ·k ₃	q ₄ ·k ₄

N

(a)

q ₁ ·k ₁	q ₁ ·k ₂	q ₁ ·k ₃	q ₁ ·k ₄
q ₂ ·k ₁	q ₂ ·k ₂	q ₂ ·k ₃	q ₂ ·k ₄
q ₃ ·k ₁	q ₃ ·k ₂	q ₃ ·k ₃	q ₃ ·k ₄
q ₄ ·k ₁	q ₄ ·k ₂	q ₄ ·k ₃	q ₄ ·k ₄

N

(b)

Figure 11.2 The $N \times N$ $\mathbf{Q}\mathbf{K}^T$ matrix showing the $q_i \cdot k_j$ values, with the upper-triangle portion of the comparisons matrix zeroed out (set to $-\infty$, which the softmax will turn to zero).

Uwaga

- Możemy utożsamiać tokeny z węzłami w grafie.
- Za pomocą odpowiedniej maski możemy sprawić, żeby atencja przebiegała tylko pomiędzy sąsiednimi węzłami w grafie.
- Maska atencji \approx macierz incydencji

Schematy atencji w transformerach dla sekwencji

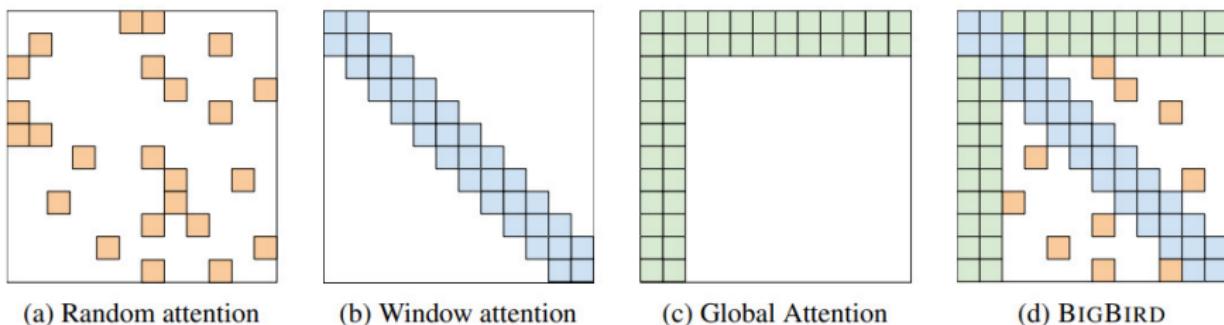
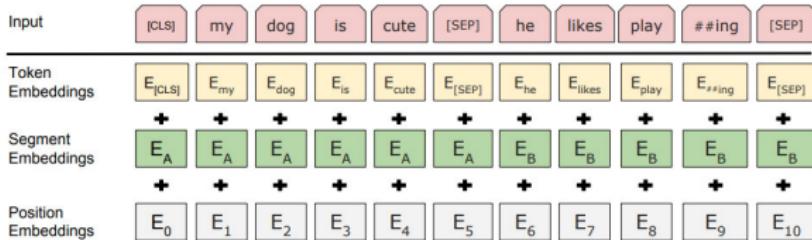


Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

- Atencja **losowa** nawiązuje do idei **małego świata** (każdy jest 6 hand-szejków od np. Donald Trumpa)
- Atencja **globalna** wprowadza specjalne węzły, które zbierają informację (i tylko z nimi się inne węzły łączą)
- Atencja **w oknie**, w której każdy węzeł łączy się z bliskimi sąsiadami, przypomina trochę sieci konwolucyjne (CNN)

Jeszcze o kodowaniu pozycji



- Nie zawsze jest konieczne: w grafach mechanizm atencji określa przepływ informacji i relacje między węzłami
- Również w wersji GPT sieć jest w stanie się uczyć (trocę wolniej) również wówczas, gdy jedynie atencja określa relacje między słowami.
- Można określać **równocześnie** różne rodzaje pozycji (jak w BERT-cie, które słowo i które zdanie)

Zagrajmy w szachy!



Zagrajmy w szachy!



Zadanie

Zaprojektować wejście do transformera w zadaniu, w którym dla pozycji szachowej mamy powiedzieć, kto ma przewagę (dla uproszczenia załóżmy, że ruch przypada na białe).

Zagrajmy w szachy!



- Transformer będzie działał w stylu BERT-a, 64 pola + jedno dodatkowe na token **[CLS]**
- Tokenami będą bierki (czarny skoczek, biały król, itd)
- Kodowanie pozycji (są różne opcje):
 - ▶ **[a5], [c3]**, ... mają osobne osadzenia (mamy nadzieję, że sieć się nauczy struktury 2D)
 - ▶ osadzenie dla **[a5]** to osadzenie dla **[a] + [5]**

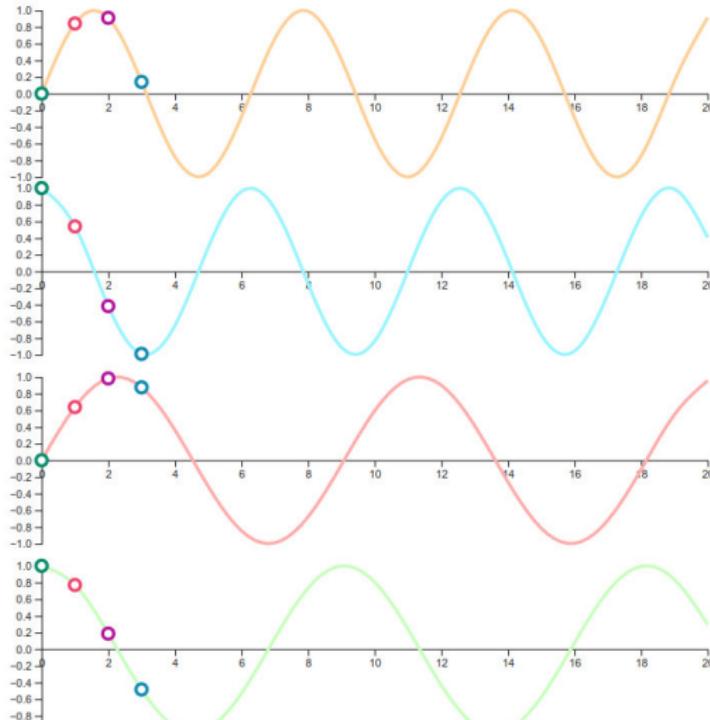
Osadzenia pozycji niepodlegające uczeniu

Uwaga

Używa się też kodowań pozycji, które nie są trenowane (tak było w oryginalnym transformerze)

- Bazują one na funkcjach sin i cos z różnymi okresami
- Pewne analogie dla kodowania binarnego: jak kolumny są kodami kolejnych liczb

Positional encoding visualization



p0	p1	p2	p3	i=0
0.000	0.841	0.909	0.141	i=0
1.000	0.540	-0.416	-0.990	i=1
0.000	0.638	0.983	0.875	i=2
1.000	0.770	0.186	-0.484	i=3

Positional Encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

Settings: $d = 50$

The value of each positional encoding depends on the *position* (pos) and *dimension* (d). We calculate result for every *index* (i) to get the whole vector.



Wracamy do treningu



Trening sieci neuronowych z lotu ptaka (przypomnienie)

- Trening jest minimalizacją **funkcji kosztu (loss)**, która bierze sieć, dane uczące i zwraca nieujemną liczbę rzeczywistą (określającą, jak bardzo sieć **nie pasuje** do danych)
- Procedura minimalizacji korzysta z gradientu, zmieniając parametry θ w kierunku największego spadku *kosztu*.
- W przypadku pojedynczego zadania klasyfikacji jest to:

$$-\log P(\hat{y}_k)$$

gdzie $P(\hat{y}_k)$ jest obliczonym przez sieć prawdopodobieństwem prawidłowego tokenu (prawidłowej klasy)

Suma takich składników dla wszystkich tokenów to nasza funkcja kosztu. Chcemy ją minimalizować, czyli maksymalizować prawdopodobieństwo danych (a dokładniej jego logarytm).

BERT. Trening (1)

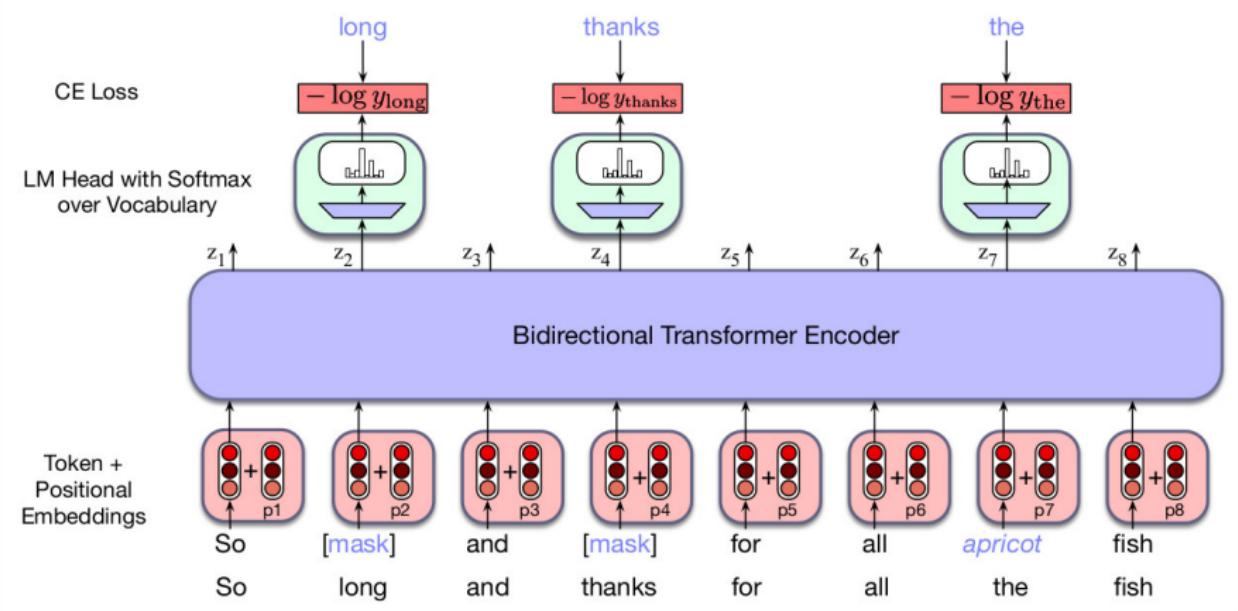


Figure 11.3 Masked language model training. In this example, three of the input tokens are selected, two of which are masked and the third is replaced with an unrelated word. The probabilities assigned by the model to these three items are used as the training loss. The other 5 tokens don't play a role in training loss.

Aby otrzymać Loss sumujemy wszystkie logarytmy, liczone na wyróżnionych pozycjach.

BERT. Trening (2)

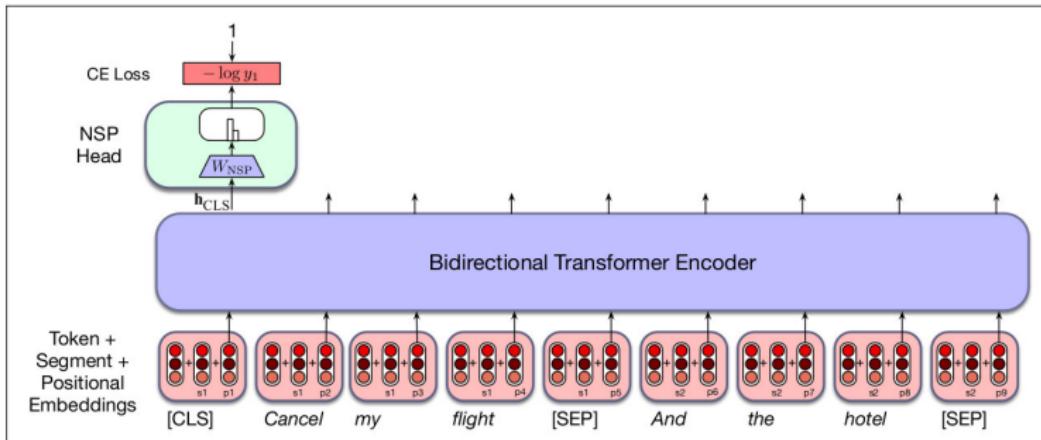


Figure 11.4 An example of the NSP loss calculation.

Do kosztu z poprzedniego slajdu dodajemy (być może z jakąś wagą) koszt związany z klasyfikacją (pary zdań)

Uwaga

To jest wariant uczenia, który może być stosowany do różnych zadań niezwiązanych z językiem (o obrazkach za parę slajdów)

GPT. Trening

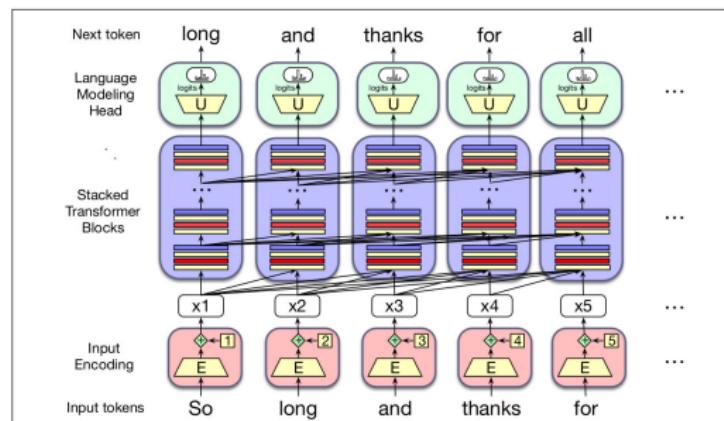


Figure 9.1 The architecture of a (left-to-right) transformer, showing how each input token get encoded, passed through a set of stacked transformer blocks, and then a language model head that predicts the next token.

Koszt jest sumą **wszystkich** logarytmów prawdopodobieństw, dla wszystkich słów **przewidywanych** (czyli przesuniętego o 1 ciągu wejściowego)

W tym przypadku mamy ...

$$-\log(\text{long}) + -\log(\text{and}) + -\log(\text{thanks}) + \dots$$

Koder-dekoder

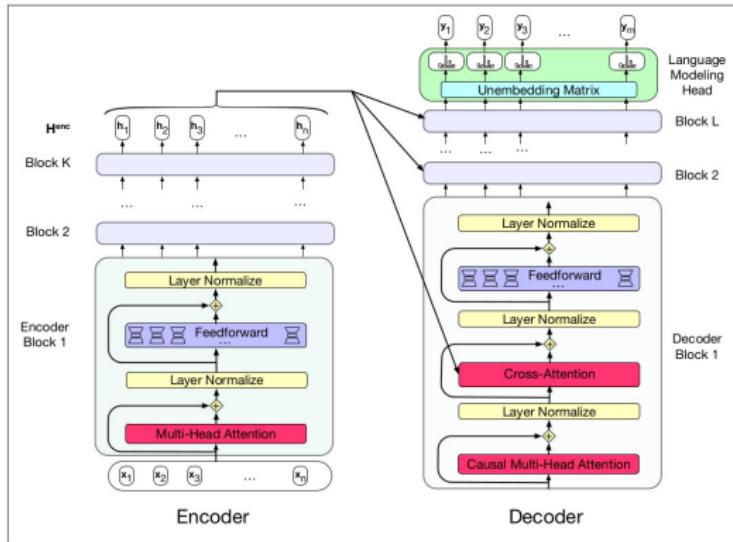


Figure 13.6 The transformer block for the encoder and the decoder. The final output of the encoder $\mathbf{H}^{enc} = \mathbf{h}_1, \dots, \mathbf{h}_n$ is the context used in the decoder. The decoder is a standard transformer except with one extra layer, the **cross-attention** layer, which takes that encoder output \mathbf{H}^{enc} and uses it to form its \mathbf{K} and \mathbf{V} inputs.

- Część kodera nie ma bezpośredniego wpływu na funkcję kosztu!
- self-attention vs cross-attention (ta druga jest pomiędzy dwiema osobnymi sieciami), nie musi patrzeć na tym samym poziomie.
- Koder może mieć atencję BERT-like

Trening wstępny i dostrajanie

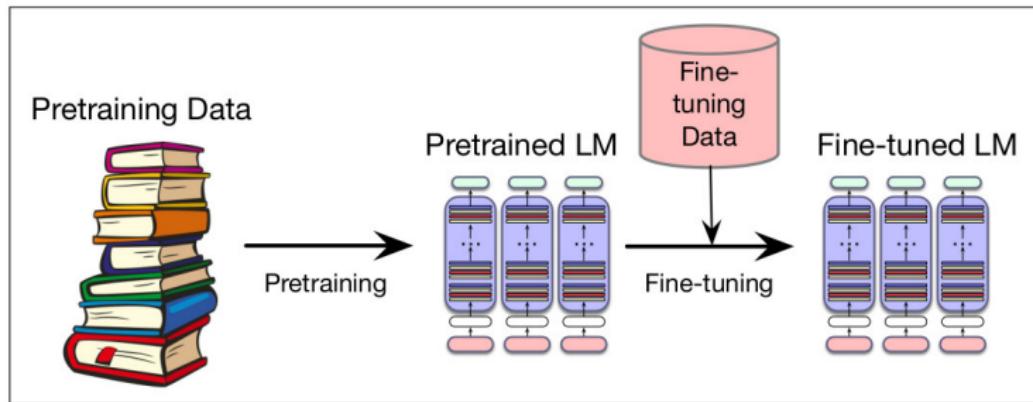


Figure 10.6 Pretraining and finetuning. A pre-trained model can be finetuned to a particular domain, dataset, or task. There are many different ways to finetune, depending on exactly which parameters are updated from the finetuning data: all the parameters, some of the parameters, or only the parameters of specific extra circuitry.

Być może oba te procesy mają dokładnie tę samą procedurę trenującą, tylko inne dane

Różne rodzaje dostrajania

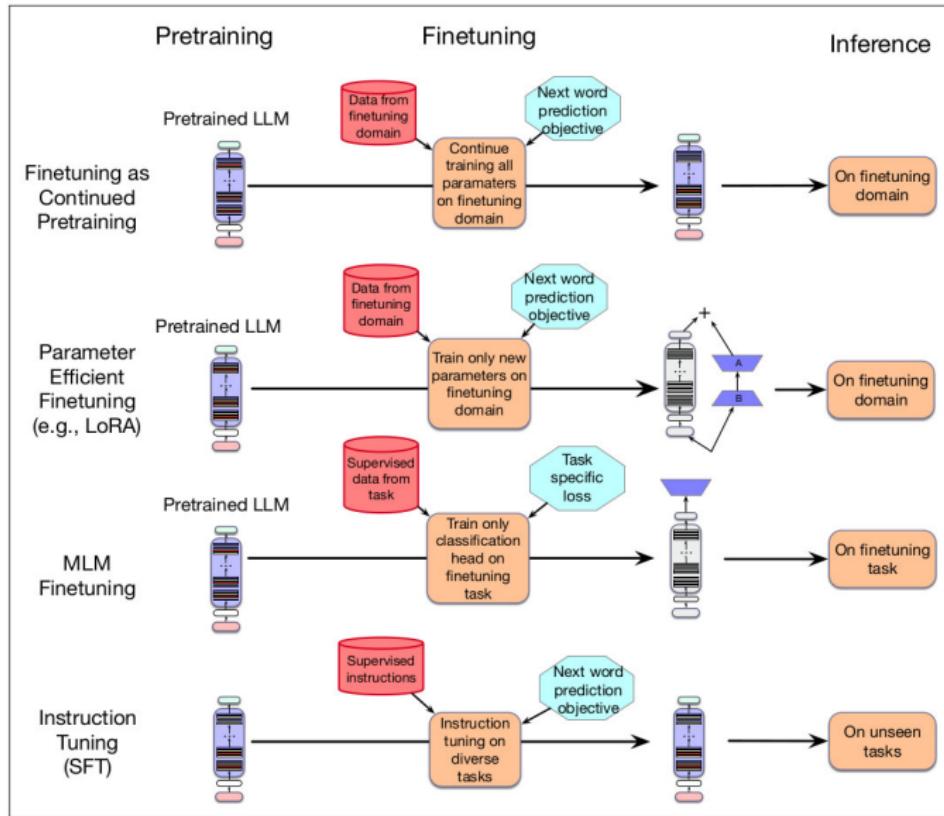


Figure 12.4 Instruction tuning compared to the other kinds of finetuning.

Uczenie ze wzmacnianiem w transformerach

- Założymy, że mamy jakąś zewnętrzną funkcję akceptującą/odrzucającą wynik generacji (pisaliśmy takie funkcje na pracowni)
- Taka funkcja może być też wytrenowana
 - ▶ Na przykład dostrajamy model, by po każdym końcu zdania wygenerował jeden z dwóch tekstów: **[+1]** albo **[-1]**, mówiących o tym, czy model zrobił wszystko jak trzeba.

Definicja

Trening, w którym **oceniacz** jest wytrenowany na podstawie ludzkich preferencji nazywamy jest **Reinforcement Learning with Human Feedback (RLHF)**

Uczenie ze wzmacnianiem w transformerach

Trening

- Generujemy zdanie/kawałek tekstu
- Oceniamy zewnętrznym ewaluatorem
- Czy jest ok?
 - ▶ **Tak:** zwiększamy prawdopodobieństwo zdania (dokładnie tak, jak po zaobserwowaniu w korpusie)
 - ▶ **Nie:** zmniejszamy prawdopodobieństwo zdania (tak, jak poprzednio, ale koszt jest pomnożony przez -1)

Oczywiście to potwarzamy tysiące/miliony razy.

Uwaga

Warto zadbać, żeby przykłady pozytywne i negatywne były mniej więcej w równowadze

Trening ChataGPT

- Kolejne slajdy będą z wystąpienia A. Karpathy'ego z OpenAI
- Link: <https://www.youtube.com/watch?v=bZQun8Y4L2A>

Implementacje GPT

Później będziemy też używać dwóch implementacji GPT Karpathy'ego:

- **miniGPT** (Link: <https://github.com/karpathy/minGPT>)
- **nanoGPT** (Link: <https://github.com/karpathy/nanoGPT>)

Base models are NOT 'Assistants'

- Base model does not answer questions
- It only wants to complete internet documents
- Often responds to questions with more questions, etc.:

Write a poem about bread and cheese.

Bread and cheese is my desire,
And it shall be my destiny.

Write a poem about someone who died of starvation.

Bread and cheese is my desire,
And it shall be my destiny.

Write a poem about angel food cake.

Write a poem about someone who choked on a ham sandwich.

Bread and cheese is my desire,
And it shall be my destiny.

Write a poem about a hostess who makes the

It can be tricked into performing tasks with prompt engineering:

Here is a poem about bread and cheese:

Bread and cheese is my desire,

And it shall be my destiny.

Bread and cheese is my desire,

And it shall be my destiny.

Here is a poem about cheese:

|

Base models are NOT 'Assistants'

(They can be somewhat tricked
into being AI assistants)

The diagram illustrates a process flow. On the left, three rounded rectangular boxes represent input types: "Make it look like document", "Few-shot prompt", and "Completion". An arrow points from "Completion" to a central vertical bar. From the top of this bar, another arrow points down to a large, light-blue rounded rectangle that contains a simulated AI conversation. The conversation starts with "[Human] Hi, how are you?" followed by "[Assistant] I'm great, thank you for asking. How I can help you today?". The human responds with "[Human] I'd like to know what is $2+2$ thanks", and the AI replies with "[Assistant] $2+2$ is 4.". The human then says "[Human] Great job.", and the AI asks "[Assistant] What else can I help you with?". Finally, the human asks "[Human] What is the capital of France?", and the AI responds with "[Assistant] Paris.".

Make it look like document

Few-shot prompt

Insert query here →

Completion

The following is a conversation between a Human and a helpful, honest and harmless AI Assistant.

[Human]
Hi, how are you?

[Assistant]
I'm great, thank you for asking. How I can help you today?

[Human]
I'd like to know what is $2+2$ thanks

[Assistant]
 $2+2$ is 4.

[Human]
Great job.

[Assistant]
What else can I help you with?

[Human]
What is the capital of France?

[Assistant]
Paris.

GPT Assistant training pipeline

