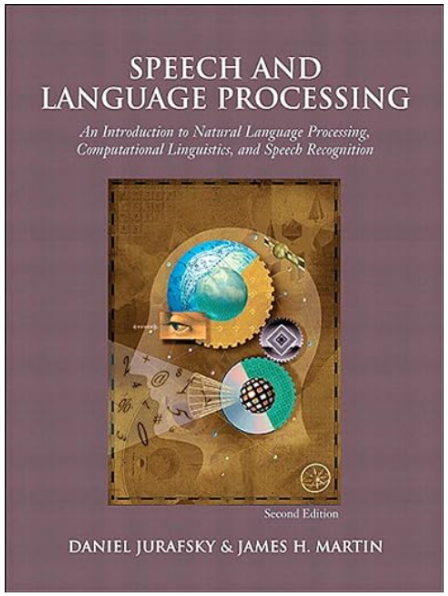
Transformery (ciąg dalszy)

Paweł Rychlikowski

Instytut Informatyki UWr

18 grudnia 2024

Literatura do przedmiotu



- Najbardziej wszechstronną książką o NLP jest Speech and Language Processing D.Jurafsky i H.Martin (hura!)
- ... ale jej ostatnie (drugie) wydanie to rok 2011 (łeee)
- ... ale autorzy pracują nad kolejnym (hura!)
- ... ale na pytanie, kiedy skończą odpowiadają na oficjalnej stronie Don't ask. (łeee)
- ... ale dopóki nie skończą, na stronie są pdf-y z aktualnymi wersjami rozdziałów (z nich brana jest część obrazków do tego wykładu)

Treść książki

Link: https://web.stanford.edu/ jurafsky/slp3/

Wiedza wstępna

- R4: Logistic Regression
- R7: Neural networks

Ważna część naszego wykładu

- R6: Vector Semantics and Embeddings
- R9: Transformers
- R10: Large Language Models
- R11: Masked Language Models
- R12: Model Alignment, Prompting, and In-Context Learning

Treść książki

Treści uzupełniające

- R8: RNNs and LSTMs
- R13: Machine Translation
- R14: Question Answering, Information Retrieval, and RAG
- R15: Chatbots and Dialogue Systems
- R16: Automatic Speech Recognition and Text-to-Speech
- R17: Sequence Labeling for Parts of Speech and Named Entities

Treść książki

Cmentarzyk

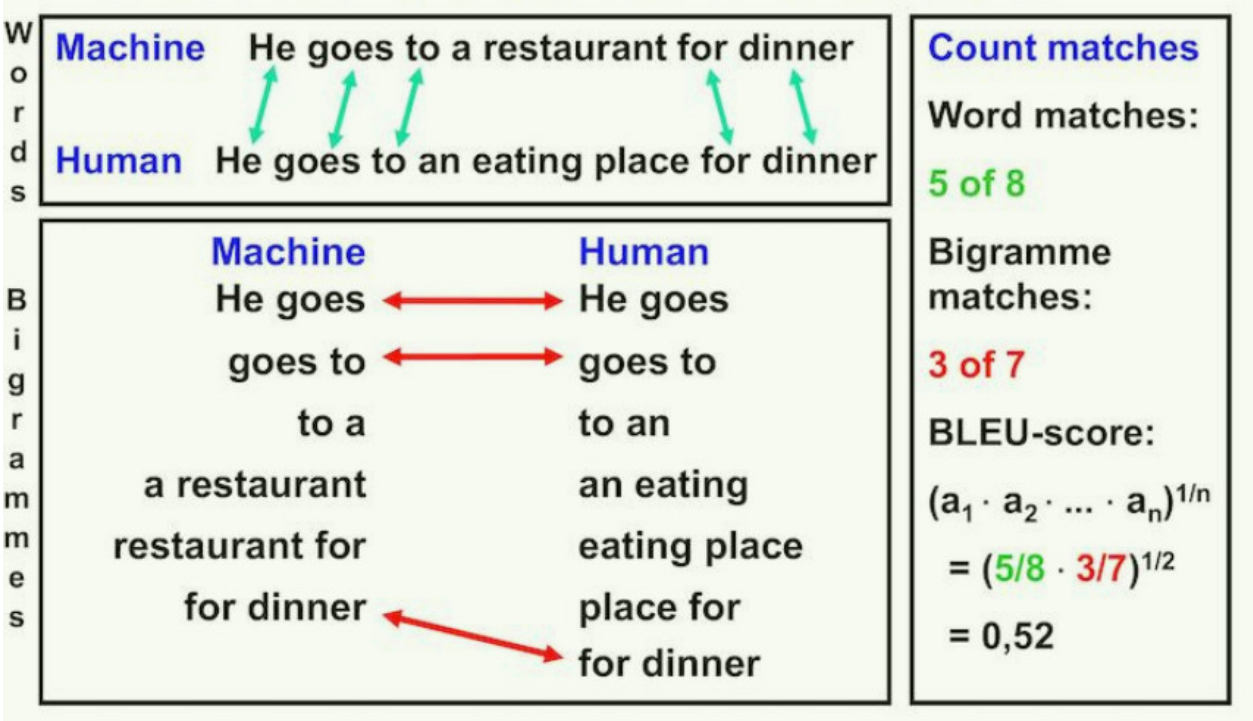
Rozdziały, które nie będą włączone do książki (jedynie w sieci)

- A: Hidden Markov Models
- B: Spelling Correction and the Noisy Channel
- C: Statistical Constituency Parsing
- D: Context-Free Grammars
- E: Combinatory Categorial Grammar
- F: Logical Representations of Sentence Meaning
- G: Word Senses and WordNet
- H: Phonetics

Metryki oceniania tłumaczenia

- Metryka BLEU (z zeszłego wykładu) ma sporo krytyków (że to nie o to chodzi w tłumaczeniu, że pomijamy synonimy, że ...)
- Mimo to jest ciągle stosowana. Ale może warto wiedzieć o nowszych metrykach...

BLEU. Przypomnienie



Problem

Restaurant wydaje się tutaj lepsze niż na przykład elephant, a metryka tego w żaden sposób nie uwzględnia.

Dlaczego BLEU jest ciągle stosowana?

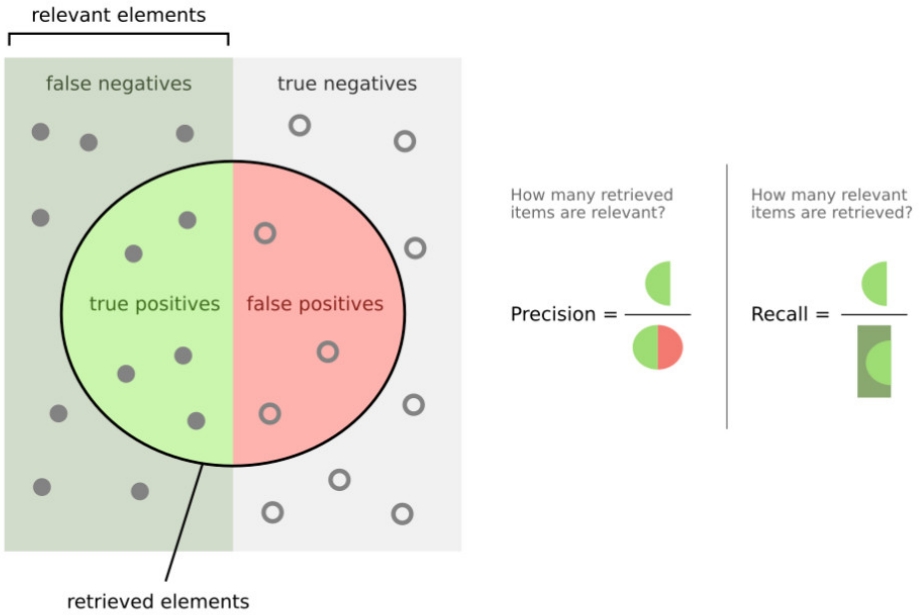
- Ludzie się przyzwyczaili, a ponadto ci, z którymi chcemy się porównać też używali BLEU (więc nie trzeba nic przeliczać, wystarczy wstawić ich liczby do naszych tabelek)
- Overfiting jest niewielki, bo używamy metryki tylko do ewaluacji, nie do treningu!
- Nie potrzebujemy idealnej metryki, tylko taką, która lepszym systemom tłumaczącym daje lepsze wyniki (a czy to będzie 90/100, czy 50/100 to nie ma większego znaczenia)

Uwaga

Ale warto zastanawiać się nad miarami semantycznego podobieństwa zdań, bo to zadanie samo w sobie jest użyteczne.

BERT-Score

Najpierw rysunkowe przypomnienie o miarach **precision** (dokładność, swoistość) oraz **recall** (zupełność, pokrycie, czułość)



Jak chcemy mieć jedną liczbę opisującą jakość systemu to bierzemy ich średnią harmoniczną i nazywamy to F_1

$$F_1 = \frac{2PR}{P+R}$$

BERT-Score

$$R_{\text{BERT}} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\tilde{x}_j \in \tilde{x}} x_i \cdot \tilde{x}_j \qquad P_{\text{BERT}} = \frac{1}{|\tilde{x}|}\sum_{\tilde{x}_j \in \tilde{x}} \max_{x_i \in x} x_i \cdot \tilde{x}_j \tag{13.21}$$

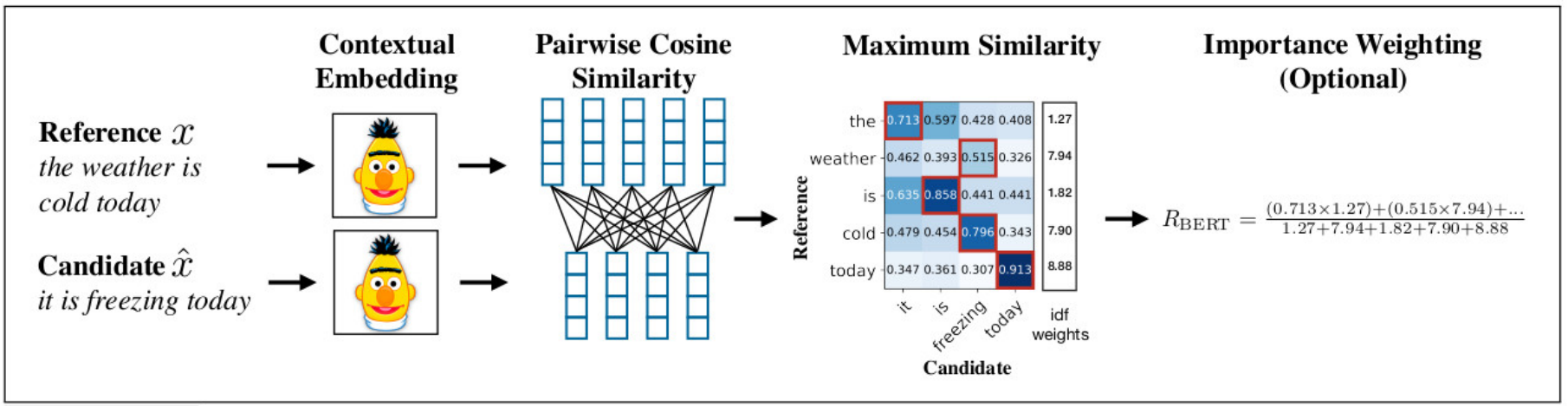


Figure 13.11 The computation of BERTSCORE recall from reference x and candidate $\hat{x}$, from Figure 1 in Zhang et al. (2020). This version shows an extended version of the metric in which tokens are also weighted by their idf values.

Wracamy do mechanizmu uwagi i transformerów

Transformery. Mechanizm uwagi, przypomnienie

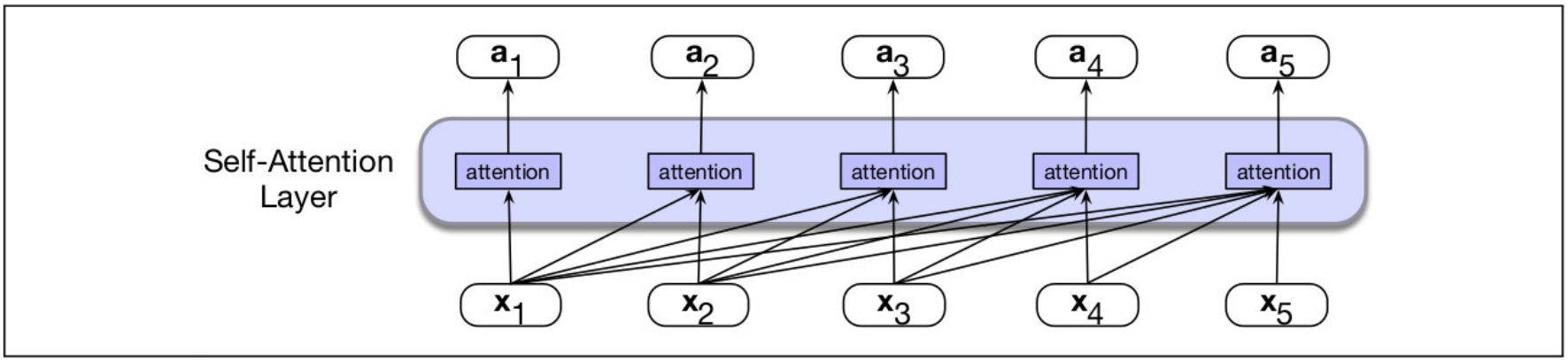


Figure 9.3 Information flow in causal self-attention. When processing each input $\mathbf{x}_i$, the model attends to all the inputs up to, and including $\mathbf{x}_i$.

- **Wejście:** osadzenia kontekstowe x_i
- **Wyjście:** wektory a_i (osadzenia z domieszką, czyli średnia ważona wektorów $x_j, j \leq i$)

Transformery. Mechanizm uwagi, przypomnienie

$$\begin{aligned}
\mathbf{q}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{Q}};\quad \mathbf{k}_j &= \mathbf{x}_j\mathbf{W}^{\mathbf{K}};\quad \mathbf{v}_j = \mathbf{x}_j\mathbf{W}^{\mathbf{V}} \\
\text{score}(\mathbf{x}_i,\mathbf{x}_j) &= \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}} \\
\alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{x}_i,\mathbf{x}_j))\ \ \forall j \le i \\
\mathbf{a}_i &= \sum_{j \le i} \alpha_{ij}\mathbf{v}_j
\end{aligned}$$

Omówimy sobie w kolejnych slajdach to dokładniej.

Transformery. Mechanizm uwagi

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^Q; \quad \mathbf{k}_j = \mathbf{x}_j \mathbf{W}^K; \quad \mathbf{v}_j = \mathbf{x}_j \mathbf{W}^V$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \ \forall j \leq i$$

$$\mathbf{a}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j$$

- Ogólny schemat wzoru: patrzymy z perspektywy pozycji i (ale obliczenia powtarzamy dla każdego i)
- Uwzględniamy *score* z osadzeniami **wcześniejszych** wektorów (i bieżącego) (w wersji GPT)

Transformery. Mechanizm uwagi

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{V}}$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \ \forall j \leq i$$

$$\mathbf{a}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j$$

- Używamy trzech macierzy: W^Q, W^V i W^K
- Zwróćmy uwagę na inne indeksy przy macierzach:
  - q_i – zapytanie zadawane przez wektor, który chce znaleźć pasujące mu osadzenia
  - k_j – odpowiedź na zapytanie, dawana przez inne wektory (w obrębie maski atencji)
  - v_j – wartość domieszowywana z odpowiednią wagą
- Uwzględniamy *score* z osadzeniami **wcześniejszych** wektorów (i bieżącego)

Softmax. Przypomnienie

$$\text{softmax}(\mathbf{z}_i) = \frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^{d} \exp(\mathbf{z}_j)} \quad 1 \leq i \leq d \tag{7.9}$$

Thus for example given a vector

$$\mathbf{z} = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1], \tag{7.10}$$

the softmax function will normalize it to a probability distribution (shown rounded):

$$\text{softmax}(\mathbf{z}) = [0.055, 0.090, 0.0067, 0.10, 0.74, 0.010] \tag{7.11}$$

Uwaga

Używany w sieciach neuronowych w sytuacji, gdy chcemy zamienić wartości zwracane przez sieć na rozkład p-stwa.

Transformery. Attention score

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{V}}$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \quad \forall j \leq i$$

$$\mathbf{a}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j$$

- Dopasowanie zapytania do q_i do klucza k_j to iloczyn skalarny.
- Jego wartości zmieniamy na wagi za pomocą funkcji softmax

Uwaga

Zwróćmy uwagę na czas $O(dN^2)$ wykonywania obliczeń

Transformery. Attention score

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{V}}$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \quad \forall j \leq i$$

$$\mathbf{a}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j$$

- Dopasowanie zapytania do q_i do klucza k_j to iloczyn skalarny.
- Jego wartości zmieniamy na wagi za pomocą funkcji softmax

Uwaga

Zwróćmy uwagę na czas $O(dN^2)$ wykonywania obliczeń (ale sporo pomogą nam tu karty graficzne)

Transformery. Mechanizm uwagi (jednogłowicowy)

Podsumowanie

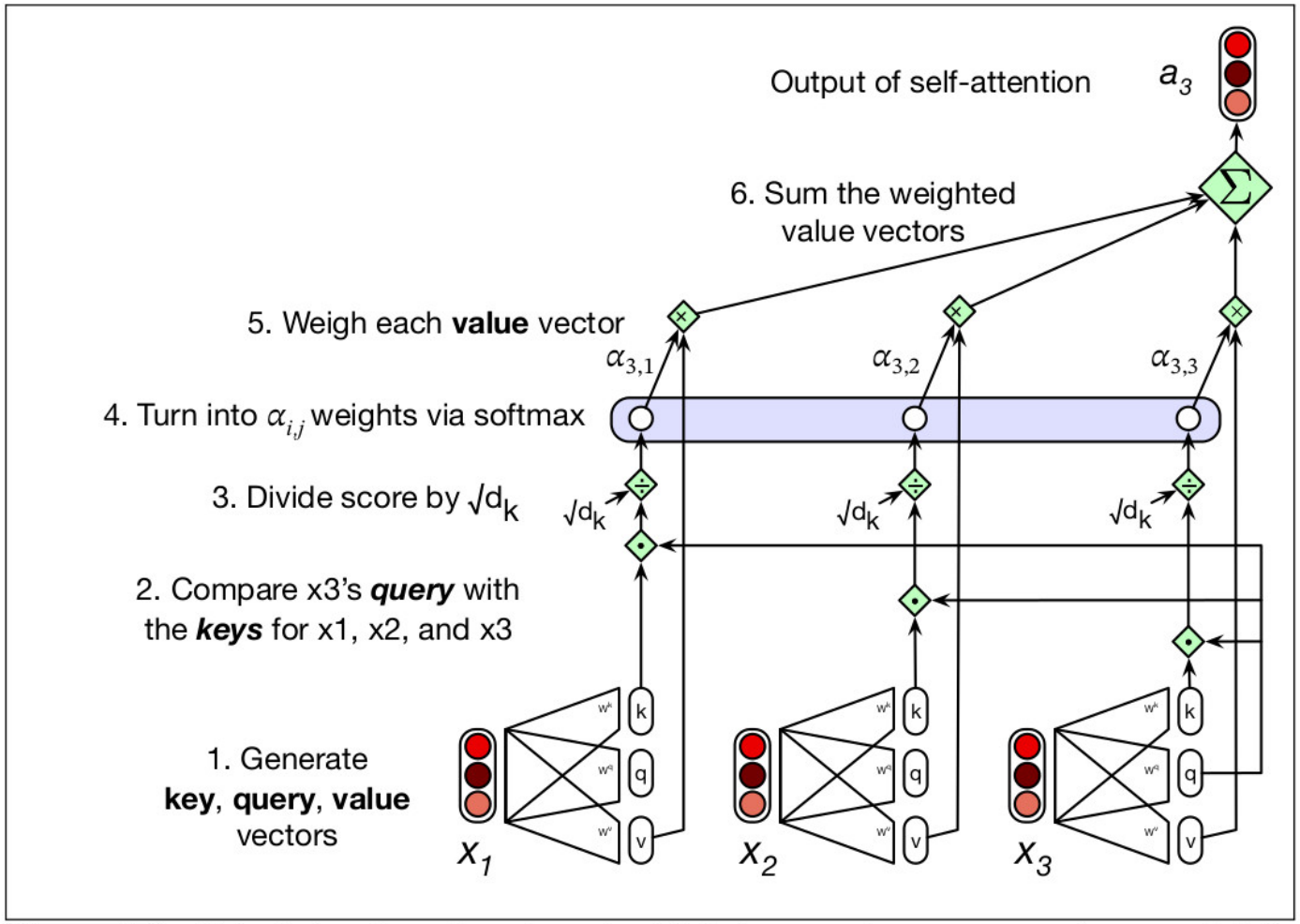


Figure 9.4 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Transformery. Mechanizm uwagi (wielogłowicowy)

- Myślimy o tym, że wektory osadzeń zawierają różne informacje (treściowe, gramatyczne, stylistyczne, ...?) – od kilkuset do kilku(nastu) tysięcy liczb.
- Możemy zatem tworzyć wiele sensownych W^Q, W^V i W^K
- Każdy z takich zestawów nazwiemy **głowicą** (głową, head)

Transformery. Mechanizm uwagi (wielogłowicowy)

$$
\begin{aligned}
\mathbf{q}_i^c = \mathbf{x}_i \mathbf{W}^{\mathbf{Qc}};\ \ \mathbf{k}_j^c = \mathbf{x}_j \mathbf{W}^{\mathbf{Kc}};\ \ \mathbf{v}_j^c &= \mathbf{x}_j \mathbf{W}^{\mathbf{Vc}};\quad \forall\, c \quad 1 \le c \le h \\
\text{score}^c(\mathbf{x}_i, \mathbf{x}_j) &= \frac{\mathbf{q}_i^c \cdot \mathbf{k}_j^c}{\sqrt{d_k}} \\
\alpha_{ij}^c &= \text{softmax}(\text{score}^c(\mathbf{x}_i, \mathbf{x}_j))\ \ \forall j \le i \\
\mathbf{head}_i^c &= \sum_{j \le i} \alpha_{ij}^c \mathbf{v}_j^c \\
\mathbf{a}_i &= (\mathbf{head}^1 \oplus \mathbf{head}^2 \ldots \oplus \mathbf{head}^h)\mathbf{W}^O \\
\text{MultiHeadAttention}(\mathbf{x}_i, [\mathbf{x}_1, \cdots, \mathbf{x}_N]) &= \mathbf{a}_i
\end{aligned}
$$

h jest liczbą głowic, W^O jest macierzą dostosowującą wymiary (można tak projektować sieć, by nie była ona konieczna), plus w kółeczku to konkatenacja.

Transformery. Schemat atencji wielogłowowej

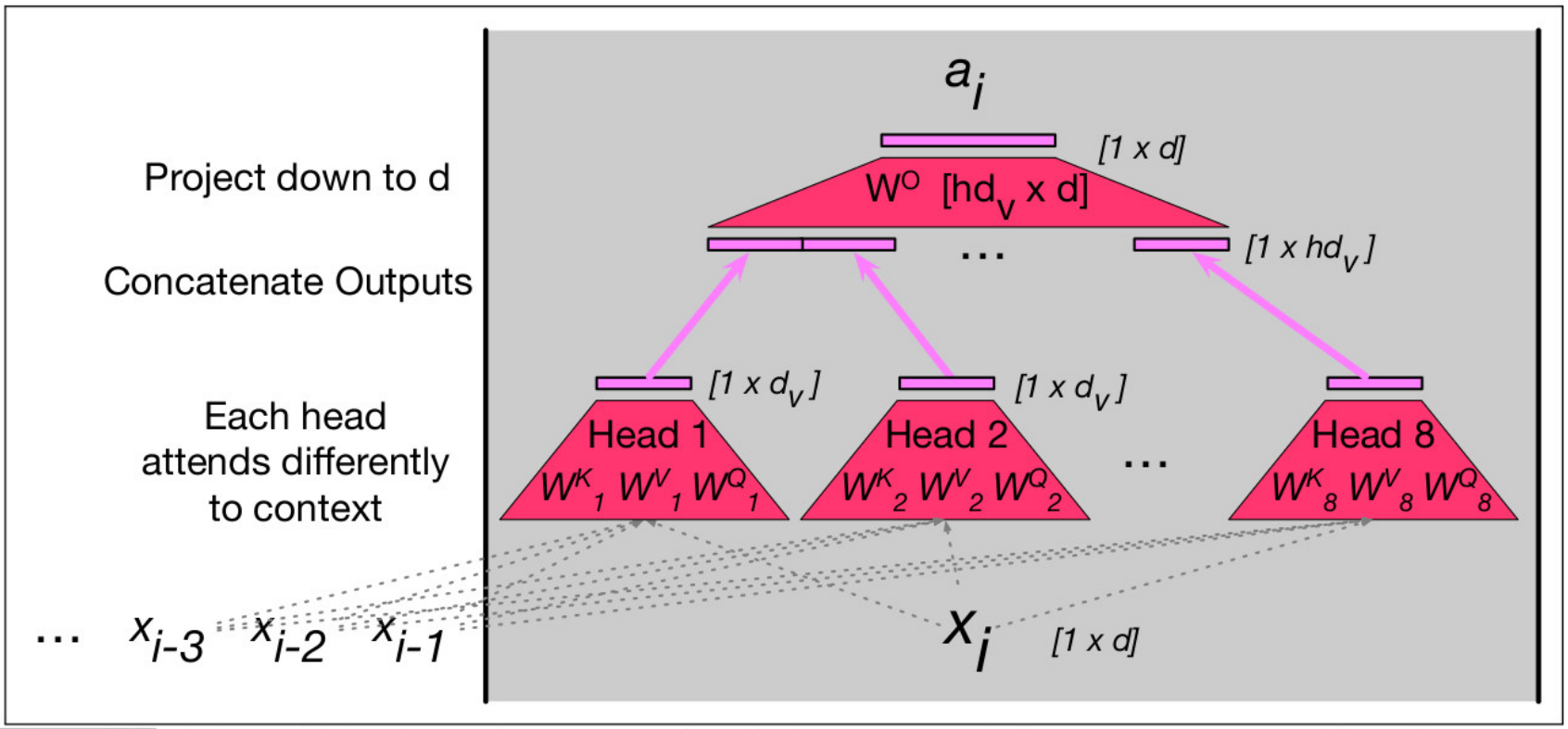


Figure 9.5 The multi-head attention computation for input $\mathbf{x}_i$, producing output $\mathbf{a}_i$. A multi-head attention layer has h heads, each with its own key, query and value weight matrices. The outputs from each of the heads are concatenated and then projected down to d, thus producing an output of the same size as the input.

Jeszcze o aspektach zanurzeń (Q, K, V)

Pytanie

Czy na pewno potrzebujemy wszystkich trzech macierzy?

- Mamy macierze odpowiedzialne za aspekty: W^Q, W^V i W^K
- Przyjmijmy, że któraś głowica ma taki kaprys: chcę popatrzeć na słowo dwie pozycje w lewo ode mnie i wziąć jego semantykę.
- W mechanizmie atencji będzie to wyglądać tak:
  - W^V: wyjęcie z osadzenie semantyki słowa
  - W^K: wyjęcie z osadzenia pozycji słowa
  - W^Q: wyjęcie z osadzenia pozycji słowa i zmniejszenie jej o 2

Transformery. Gdzie jesteśmy?

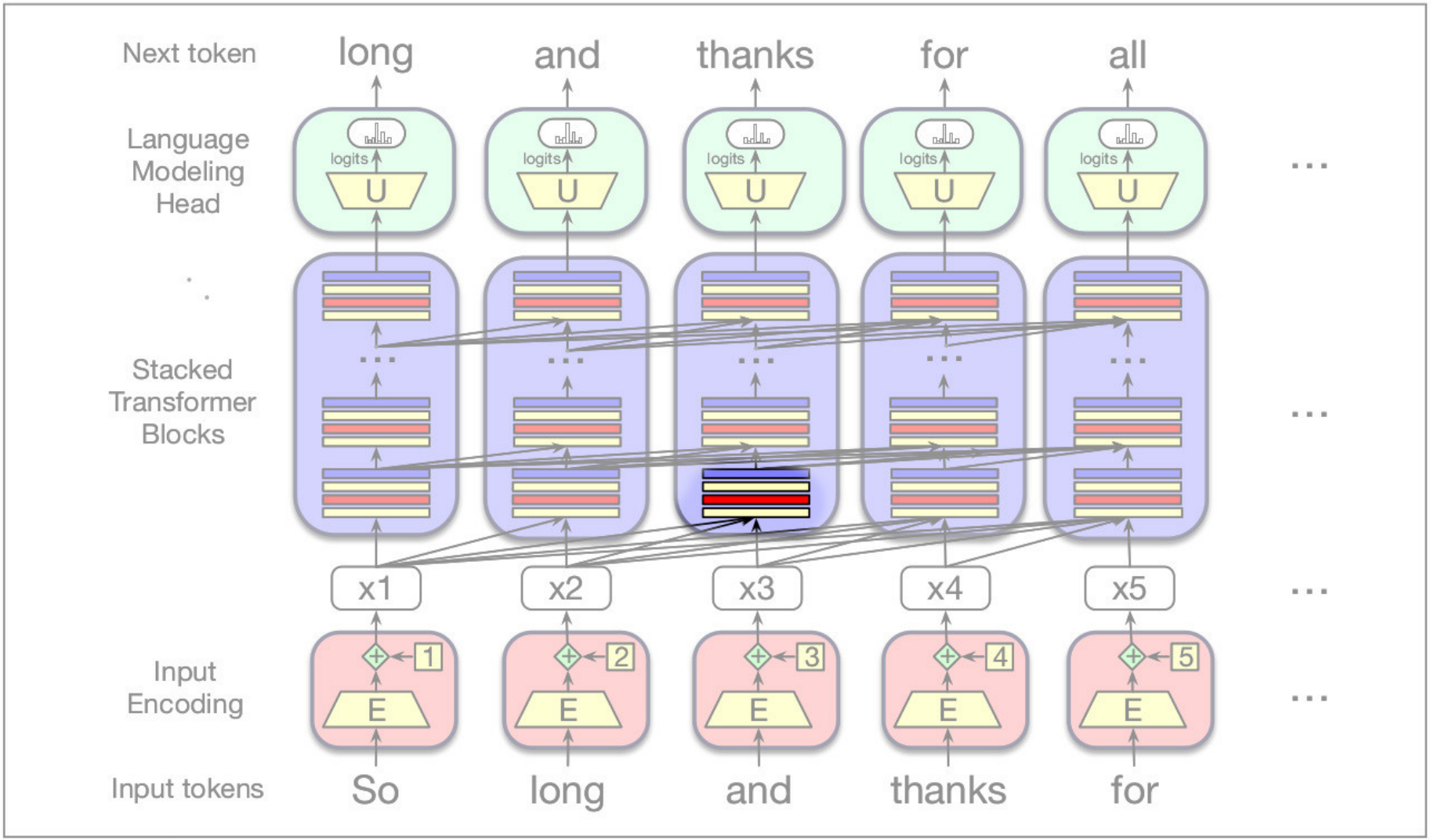


Figure 9.1 The architecture of a (left-to-right) transformer, showing how each input token get encoded, passed through a set of stacked transformer blocks, and then a language model head that predicts the next token.

Transformery

- Attention is **all** you need? Really?
- Samo inteligentne uśrednianie to za mało – potrzebujemy klasycznej sieci neuronowej pomiędzy kolejnymi uśrednianiami (i paru drobiazgów do tego)
- Od teraz mówimy o operacjach na **pojedynczym** osadzeniu (co daje możliwość wykonywania równoległego)

Wzór na sieć neuronową, przekształcającą osadzenie:

$$\mathrm{FFN}(\mathbf{x}_i) = \mathrm{ReLU}(\mathbf{x}_i\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

$(\mathrm{ReLU}(x) = \max(x, 0))$

Transformery. Schemat przepływów

- Myślimy o tym, że przez sieć płyną osadzenia, czasem się trochę mieszając z sąsiadami.
- To zachowanie jest ok, nie chcemy go tracić, tylko lekko zmodyfikować (dodając pewne nowe rzeczy)

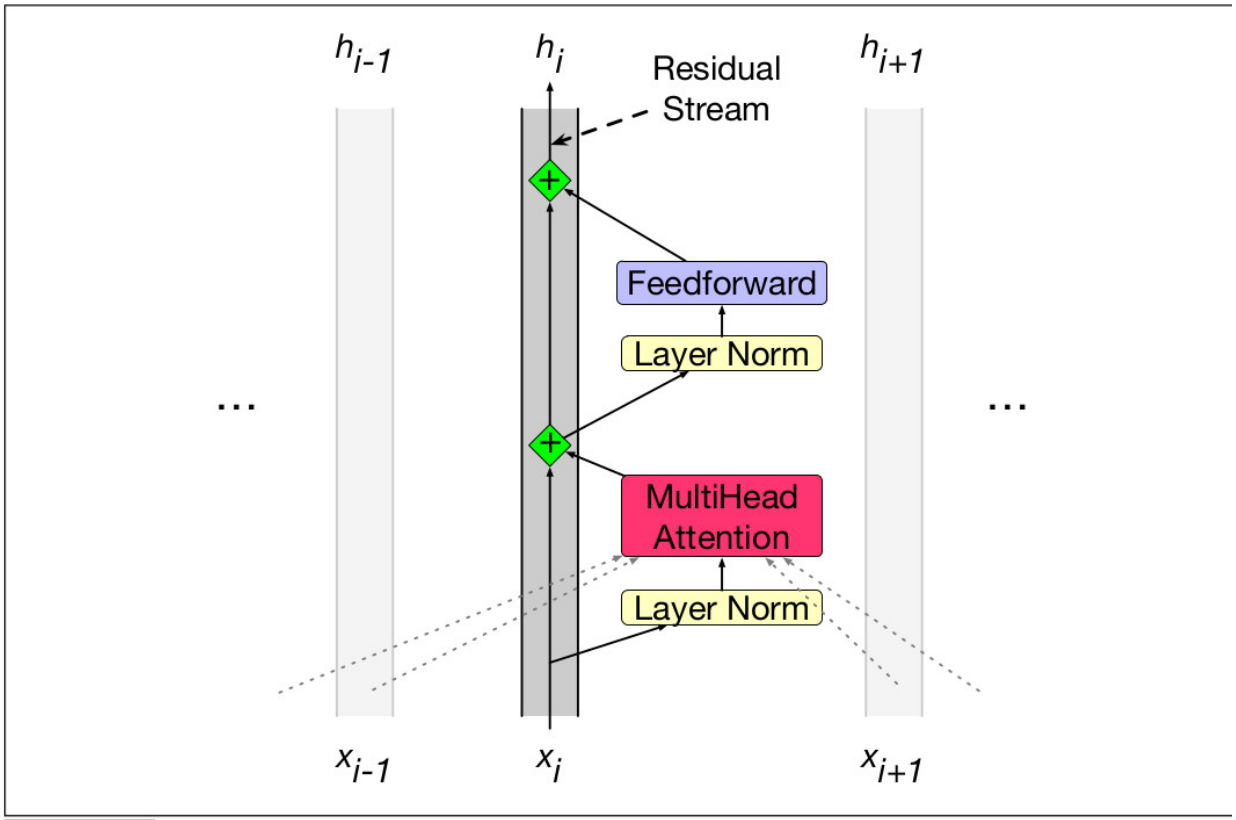


Figure 9.6 The architecture of a transformer block showing the **residual stream**. This figure shows the **prenorm** version of the architecture, in which the layer norms happen before the attention and feedforward layers rather than after.

Transformery. Normalizacja warstwy

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i \tag{9.21}$$

$$\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i-\mu)^2} \tag{9.22}$$

Given these values, the vector components are normalized by subtracting the mean from each and dividing by the standard deviation. The result of this computation is a new vector with zero mean and a standard deviation of one.

$$\hat{\mathbf{x}} = \frac{(\mathbf{x}-\mu)}{\sigma} \tag{9.23}$$

Finally, in the standard implementation of layer normalization, two learnable parameters, γ and β, representing gain and offset values, are introduced.

$$\text{LayerNorm}(\mathbf{x}) = \gamma\frac{(\mathbf{x}-\mu)}{\sigma} + \beta \tag{9.24}$$

Nie wyszła im ta nazwa! Normalizujemy pojedyncze osadzenie, nie warstwę.

Transformery. Definicja bloku

Putting it all together The function computed by a transformer block can be expressed by breaking it down with one equation for each component computation, using $\mathbf{t}$ (of shape $[1 \times d]$) to stand for transformer and superscripts to demarcate each computation inside the block:

$$\mathbf{t}_i^1 = \text{LayerNorm}(\mathbf{x}_i) \tag{9.25}$$
$$\mathbf{t}_i^2 = \text{MultiHeadAttention}(\mathbf{t}_i^1, [\mathbf{x}_1^1, \cdots, \mathbf{x}_N^1]) \tag{9.26}$$
$$\mathbf{t}_i^3 = \mathbf{t}_i^2 + \mathbf{x}_i \tag{9.27}$$
$$\mathbf{t}_i^4 = \text{LayerNorm}(\mathbf{t}_i^3) \tag{9.28}$$
$$\mathbf{t}_i^5 = \text{FFN}(\mathbf{t}_i^4) \tag{9.29}$$
$$\mathbf{h}_i = \mathbf{t}_i^5 + \mathbf{t}_i^3 \tag{9.30}$$

- Te obliczenia są powtarzane kilka/kilkadziesiąt razy (dla różnych parametrów, choć były i warianty zachowujące te same parametry)

Gdzie jesteśmy?

Na poprzednim slajdzie były **pełne obliczenia** wyznaczające kontekstowe osadzenia. Zostało nam jeszcze:

- Omówić wykorzystanie tych osadzeń do obliczania prawdopodobieństw.
- Omówić kwestie równoległości obliczeń i wsadów.
- Omówić trening transformerów (nieco dokładniej, niż od tej pory)

Głowa językowa (language modeling head)

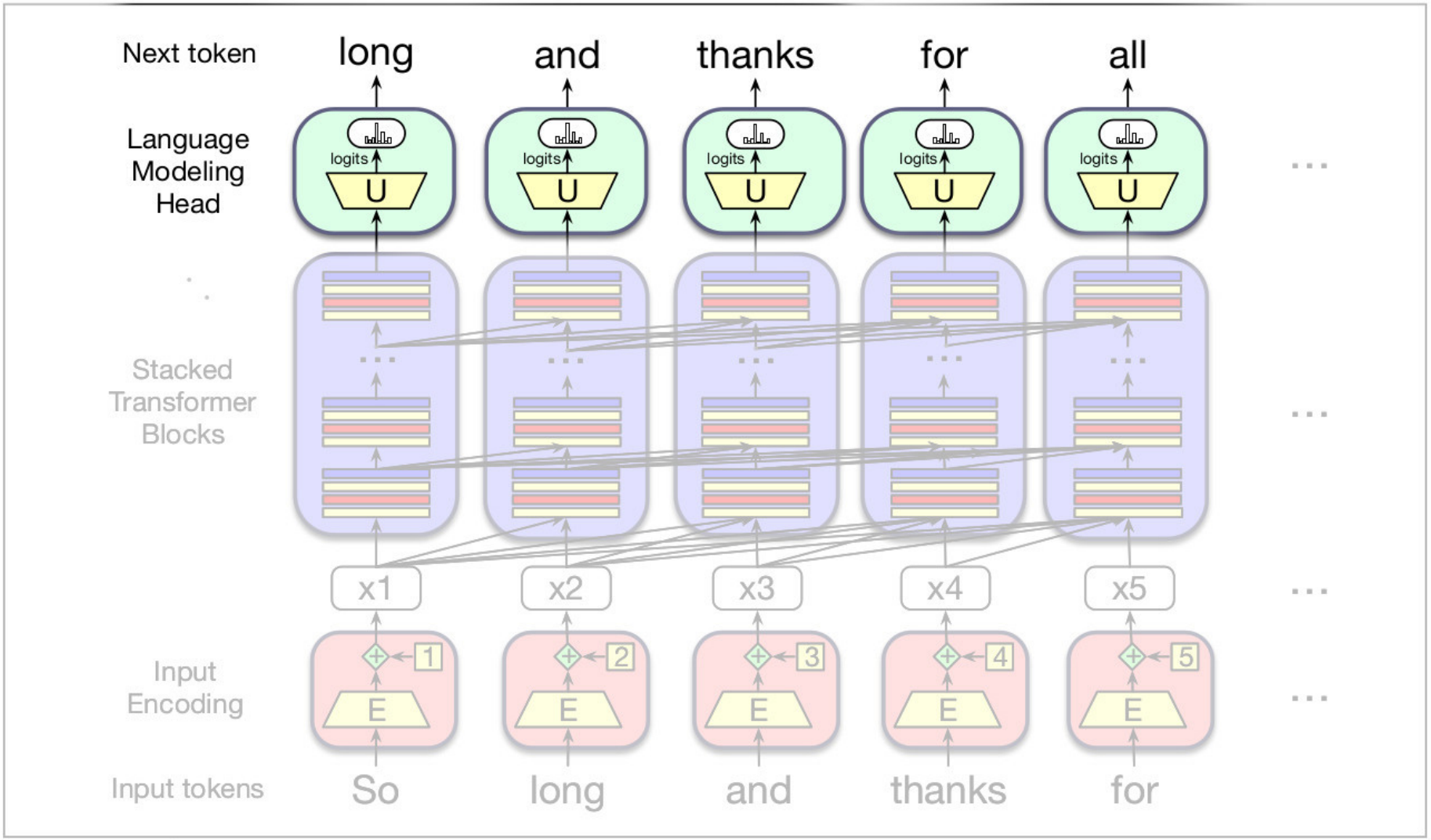


Figure 9.1 The architecture of a (left-to-right) transformer, showing how each input token get encoded, passed through a set of stacked transformer blocks, and then a language model head that predicts the next token.

Głowa językowa (language modeling head)

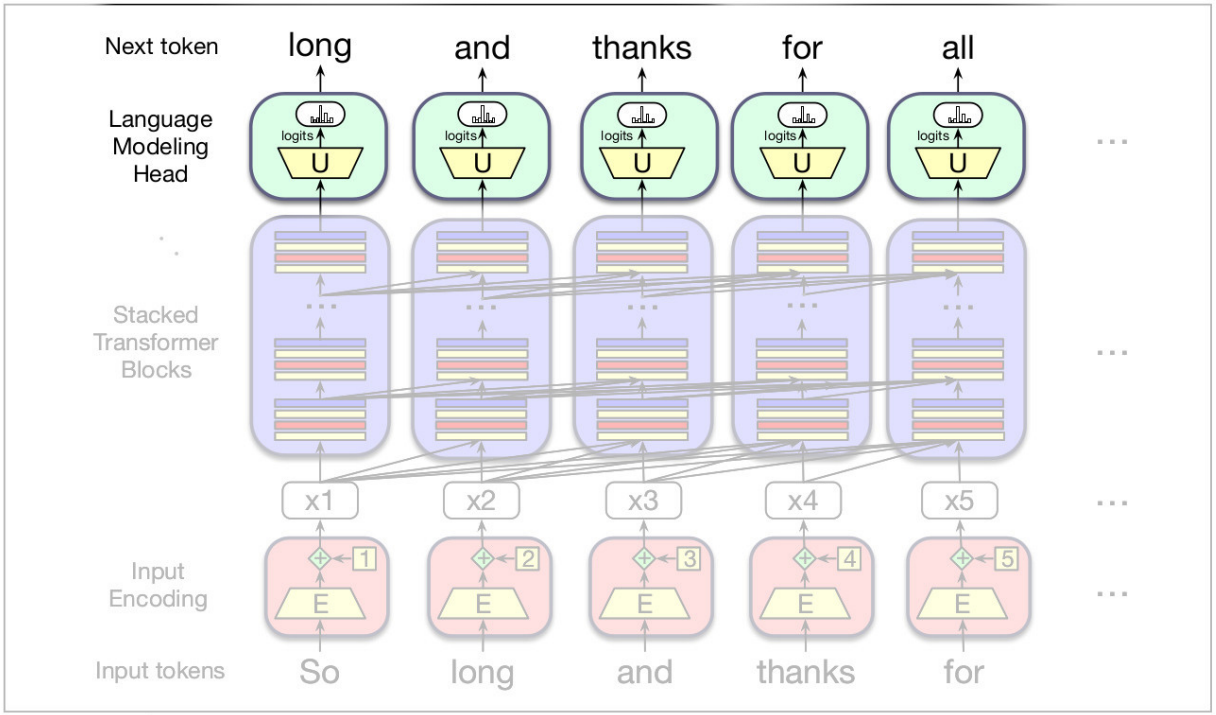


Figure 9.1 The architecture of a (left-to-right) transformer, showing how each input token get encoded, passed through a set of stacked transformer blocks, and then a language model head that predicts the next token.

- Osadzenia mnożymy przez macierz U, o wymiarach $d \times V$ (czyli otrzymamy punkty dla każdego tokenu w słowniku)
- Na to nakładamy operację softmax

Głowa językowa (language modeling head)

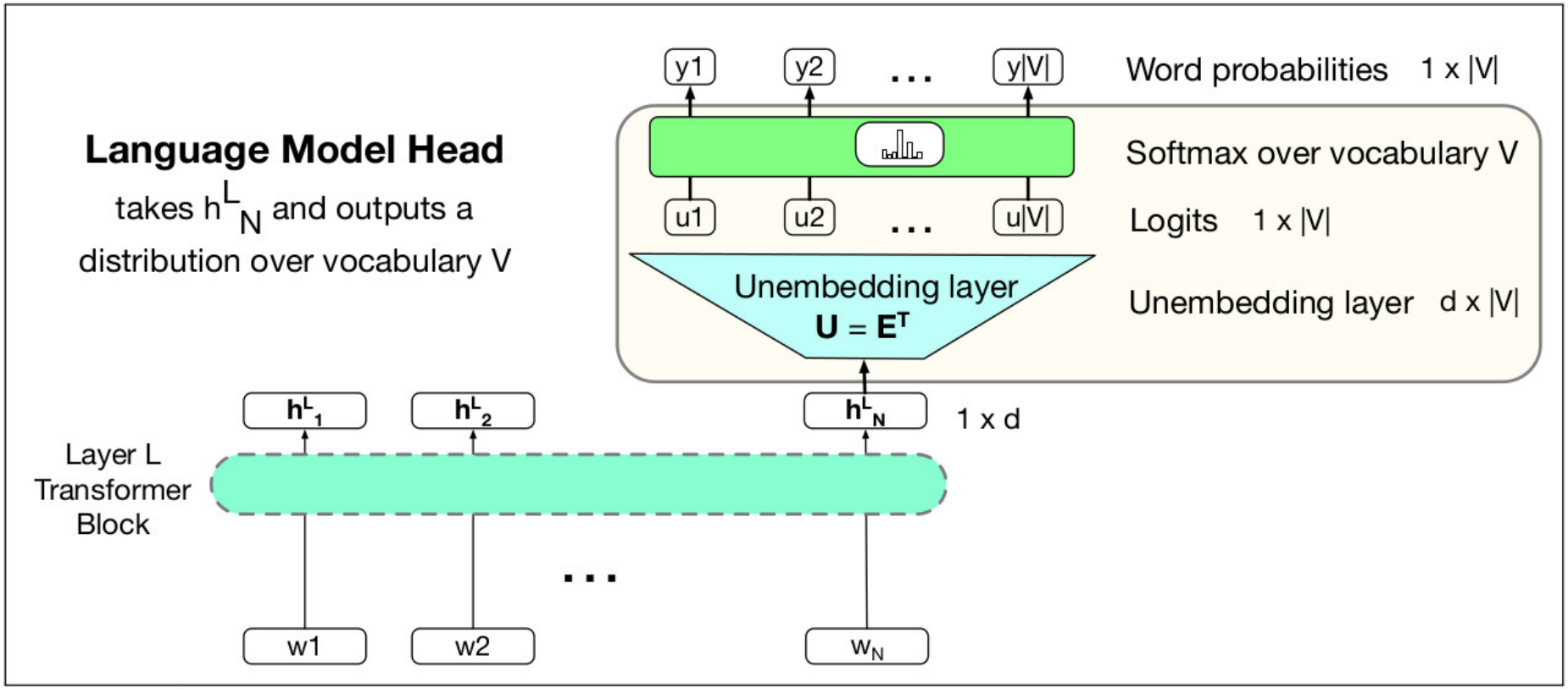
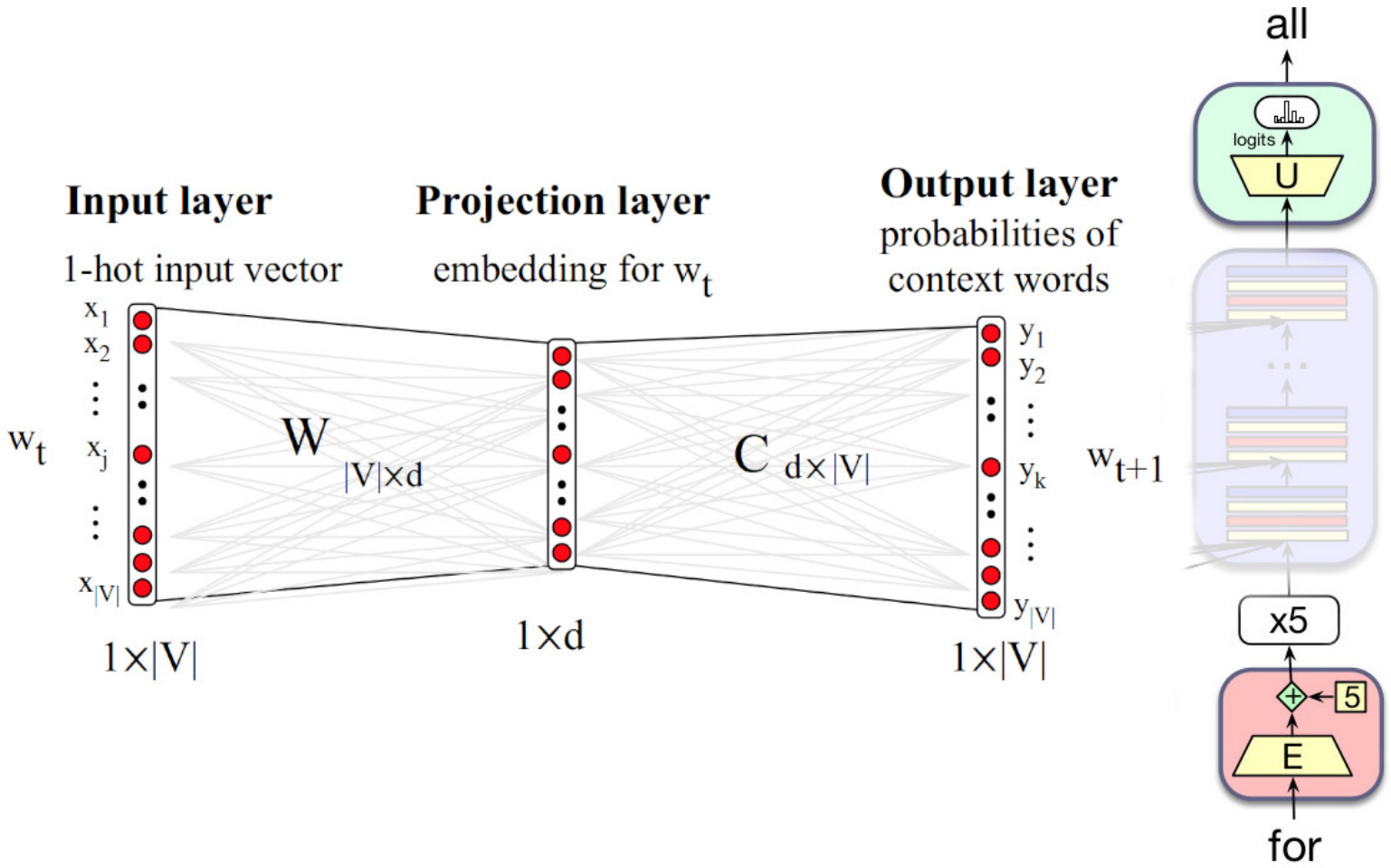


Figure 9.14 The language modeling head: the circuit at the top of a transformer that maps from the output embedding for token N from the last transformer layer ($\mathbf{h}^L_N$) to a probability distribution over words in the vocabulary V.

- E^T występuje dlatego, żeby macierz U jest zorientowana kolumnowo (a nie wierszowo, jak tradycyjne osadzenia)
- **Logity** – to słowo oznacza wartości sieci przekazywane softmaxowi.

Transformer vs word2vec



Transformer dokłada *trochę* obliczeń do bigramowego word2vec-a (ale w rezydualnych potokach)

Atencja w wersji macierzowej

- Chcemy obsłużyć wszystkie $x_i, i \in \{1, \ldots, n\}$ naraz, za pomocą pojedynczych mnożeń macierzy
- Umieścimy je w tym celu w macierzy X, jako n wierszy.

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}}; \ \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}}; \ \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}} \tag{9.31}$$

Given these matrices we can compute all the requisite query-key comparisons simultaneously by multiplying $\mathbf{Q}$ and $\mathbf{K}^\intercal$ in a single matrix multiplication. The product is of shape $N \times N$, visualized in Fig. 9.8.

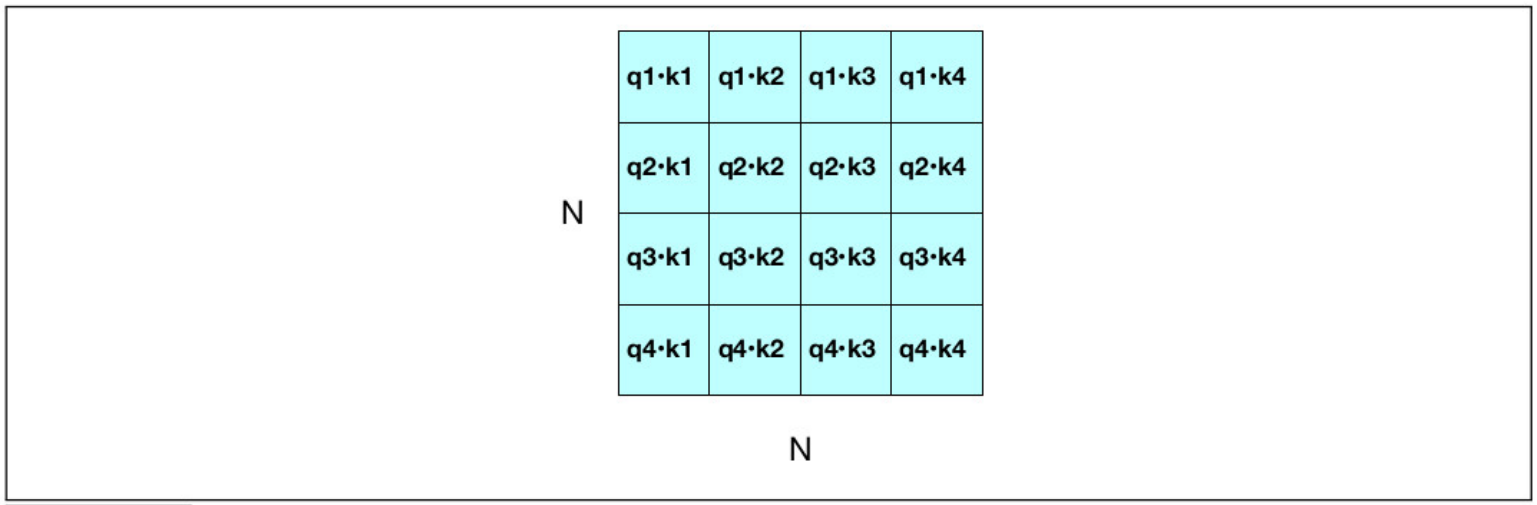


Figure 9.8 The $N \times N$ $\mathbf{Q}\mathbf{K}^\intercal$ matrix showing how it computes all $q_i \cdot k_j$ comparisons in a single matrix multiple.

Atencja w wersji macierzowej z maską

- Poniżej zobaczymy pełen wzór na atencję, w wersji macierzowej
- Operator mask jest identycznością dla BERT-a, a dla GPT wstawia $-\infty$ w miejscach niedozwolonej atencji (czyli patrzącej w przód)

$$\mathbf{A} = \text{softmax}\left(\text{mask}\left(\frac{\mathbf{QK}^\intercal}{\sqrt{d_k}}\right)\right)\mathbf{V} \tag{11.1}$$

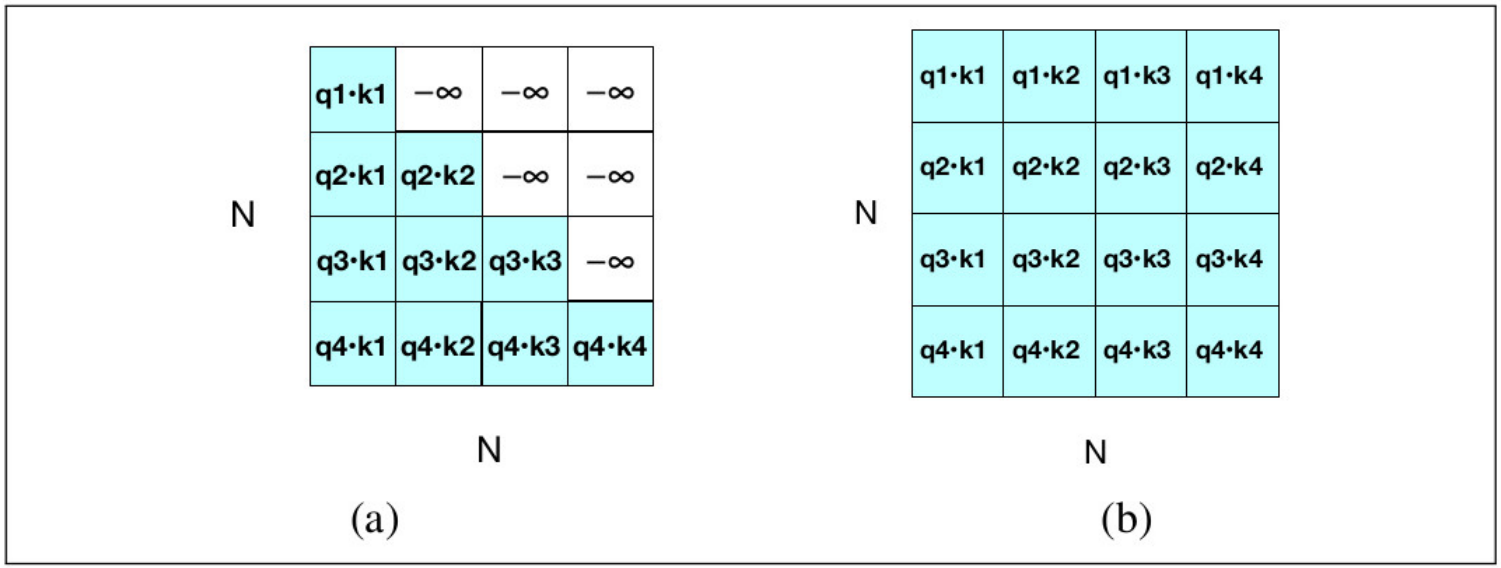


Figure 11.2 The $N \times N$ $\mathbf{QK}^\intercal$ matrix showing the $q_i \cdot k_j$ values, with the upper-triangle portion of the comparisons matrix zeroed out (set to $-\infty$, which the softmax will turn to zero).

Trening sieci neuronowych z lotu ptaka

- Sieć neuronowa jest różniczkowalną funkcją $f_\theta : \mathcal{R}^n \to \mathcal{R}^m$, gdzie θ to parametry
- **Trening sieci** to dostosowanie parametrów θ do zestawu danych uczących, czyli

$$\{(x_i, y_i)\}_{i=1}^N, \text{ gdzie } x_i \in \mathcal{R}^n, y_i \in \mathcal{R}^m$$

- Trening jest minimalizacją **funkcji kosztu (loss)**, która bierze sieć, dane uczące i zwraca nieujemną liczbę rzeczywistą (określającą, jak bardzo funkcja **nie pasuje** do danych)

Procedura minimalizacji korzysta z gradientu, zmieniając parametry θ w kierunku największego spadku *kosztu*.

Entropia krzyżowa (cross entropy)

- Czym w ogólnym przypadku jest entropia krzyżowa zajmiemy się na ćwiczeniach.
- 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 entropii dla wszystkich tokenów to nasza funkcja kosztu. Chcemy ją minimalizować, czyli maksymalizować prawdopodobieństwo danych (a dokładniej jego logarytm).