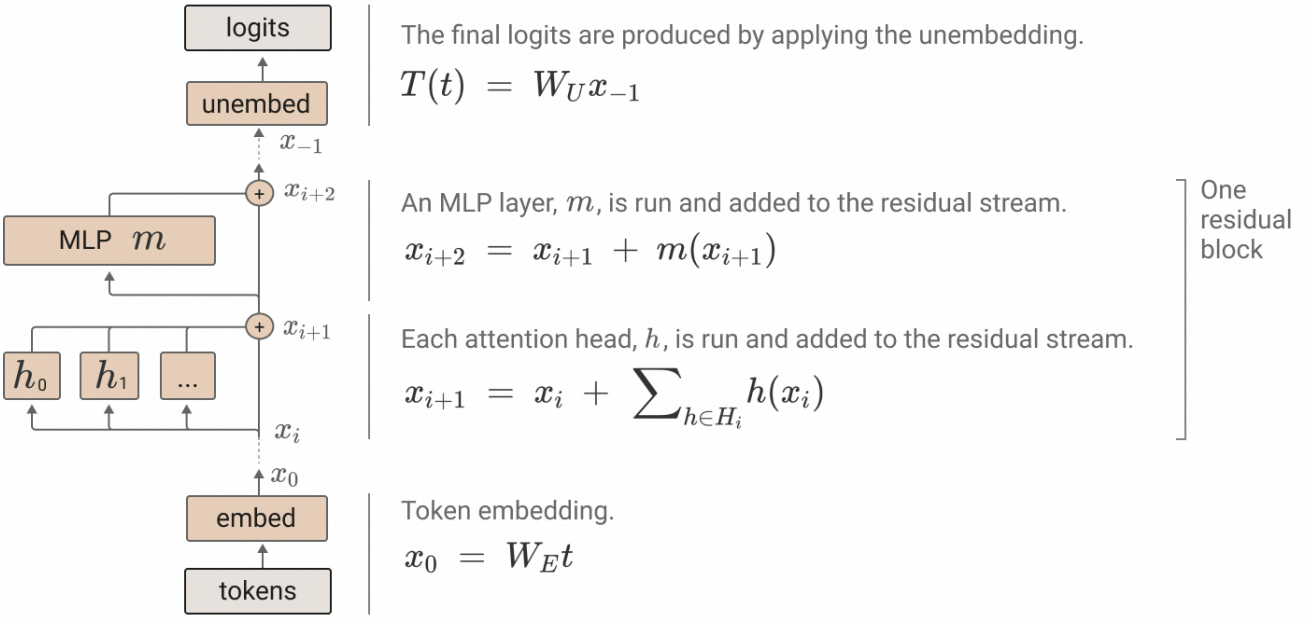
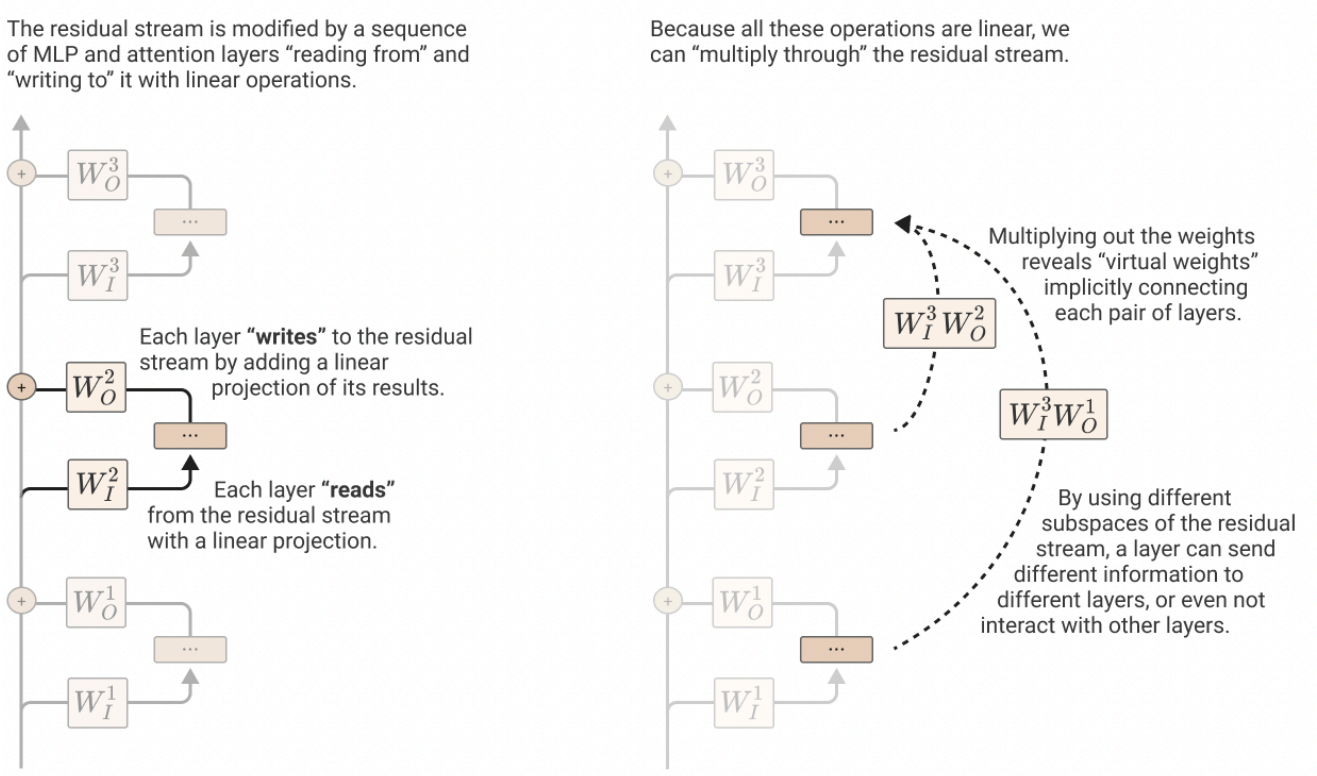
Architektura



[source: Anthropic]

1. Wymiarowość kanału rezydualnego jest bardzo wysoka
    - głowice atencji pracują w stosunkowo niskich wymiarach
    - Główny kanał rezydualny to wielokrotność; w dużych modelach to dziesiątki tysięcy
2. Transformer ma *liniową* strukturę
    - główny kanał **rezydualny** przenosi informację od embeddingu do de-embeddingu
    - Wyjścia poszczególnych bloków, atencji czy MLP, są **dodawane** do niego
        - Każdy blok wczytuje informację z kanału by zaraz potem do niego wczytać, nie omijając innych bloków
    - Ta architektura jest **bardziej liniowa** nawet niż ResNet, który wykonuje nieliniową transformację
    - Mogą być różnice w architekturze

Przetwarzanie



[source: Anthropic]

1. W ten sposób wartości przetwarzania wpływają bezpośrednio **nie tylko** na na następny poziom, ale także na wszystkie późniejsze
    - Trochę jak w architekturach typu densenet
2. Sieć MLP na drugim poziomie przetwarza sumę $W_I^3 \otimes W_)^2$ razem z $W_I^3 \otimes W_O^1$
   - oczywiście razem z kanałem rezydualnym
   - To jakby **wirtualne** wagi
   - Te informacje są przekazywane na bardzo duże odległości — w typowych sieciach informacje szybko *zanikają*
3. Poszczególne warstwy, MLP czy atencje mogą
    - Przepisywać informacje z jednego kanału do innego
    - dopisywać istotne informacje do kanałów (podprzestrzeni) strumienia rezydualnego
    - Mogą je później usuwać
4. Głowice atencyjne przetwarzają informacje równolegle i niezależnie

$$[W_O^{h_1}, \ldots, W_O^{h_n}] \cdot [r^{h_1}, \ldots, r^{h_n}]^T = \sum_i W_O{}^{h_i} r^{h_i}$$

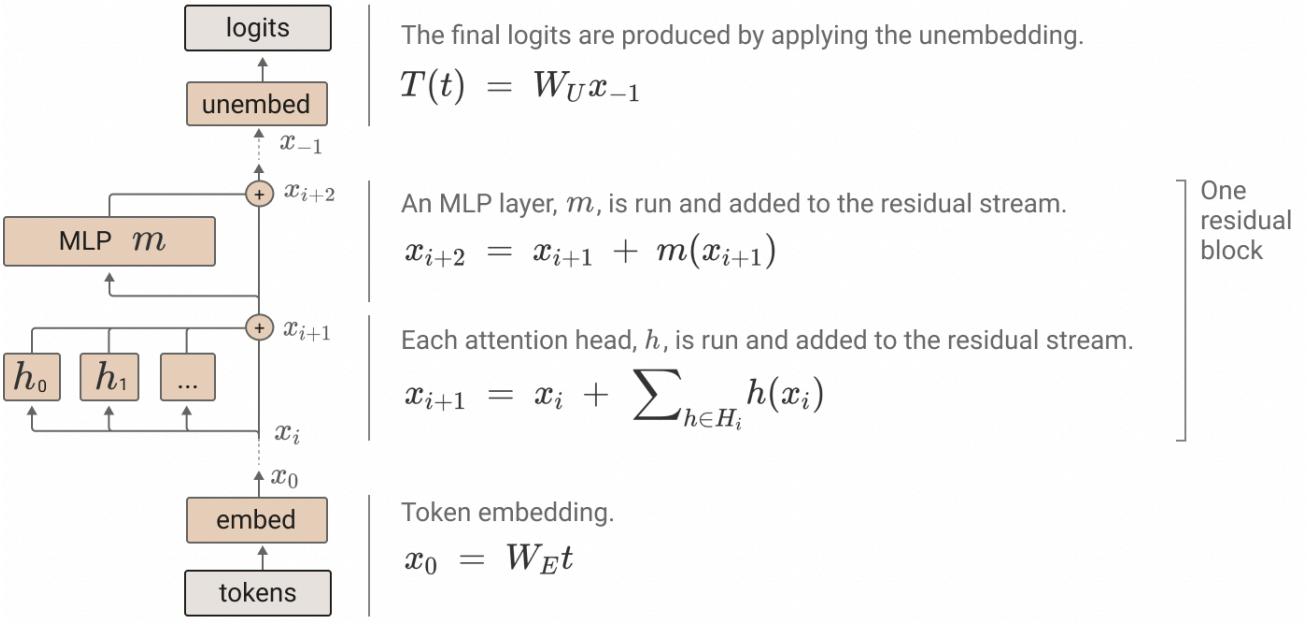
    - wyjście z głowy atencji dla każdego tokena to $h_i = W_O r_i$ gdzie
        - $r_i = \sum_j A_{i,j} v_j$ jako kombinacja liniowa macierzy atencji A i wektora
        - składnika *value* tokena $v_j = \sum_k W_V x_k$
    - te operacje mogą być efektywnie zaimplementowane wykorzystując tensory
5. głowice agencyjne zapisują informacje z kanałów jednego tokena do tych dla innego (czy innych)
    - Macierz A w steruje które tokeny j wpływają na informacje dotyczące tokenu i

    - Macierz A, poza software na końcu, jest w pełni liniowa
- macierze W_O i W_V decydują jakie informacje są przekazywane

Czy możemy mieć transformer bez atencji i MLP?

1. Warstwy embedding i de-embedding dla tokenów t są **także** uczone
    - Embedding jest liniowy
    - $T(t) = W_U W_E t$, gdzie t to one-hot reprezentacja symbolu
2. Bez atencji i MLP transformer może się nauczyć bigramów

Transformer jednowarstwowy



1. Model ensemble: złożenie modeli
    - bi-gramowego w kanale rezydualnym
    - Wielu modeli relacji na dłuższych odcinkach
2. Relacje typu $A \dots X Y$
    - Bi-gram następstwa Y po X
    - atencja aktualnego X do któregoś wcześniejszego A
    - Model zdobywa te informacje już bez MLP
3. Przetwarzanie dla warstwy bez MLP

$$\begin{aligned} T = & Id \otimes W_U \\ & \cdot (Id + \sum_{h \in H_i} A^h \otimes W_{OV}^h) \\ & \cdot Id \otimes W_E \end{aligned}$$

4. Pierwszy i ostatni składnik można połączyć (rachunek tensorowy) w

$$Id \otimes W_U W_E,$$

oraz całość w sumę

$$T = Id \otimes W_U W_E + \sum_{h \in H_i} A^h \otimes (W_U W_{OV}^h W_E),$$

gdzie $A^h = softmax(t \cdot W_U^T W_{QK}^h W_E \cdot t)$ jest *wzorcem atencji* $A[i,j]$ opisującym istotność tokenu i dla tokenu j

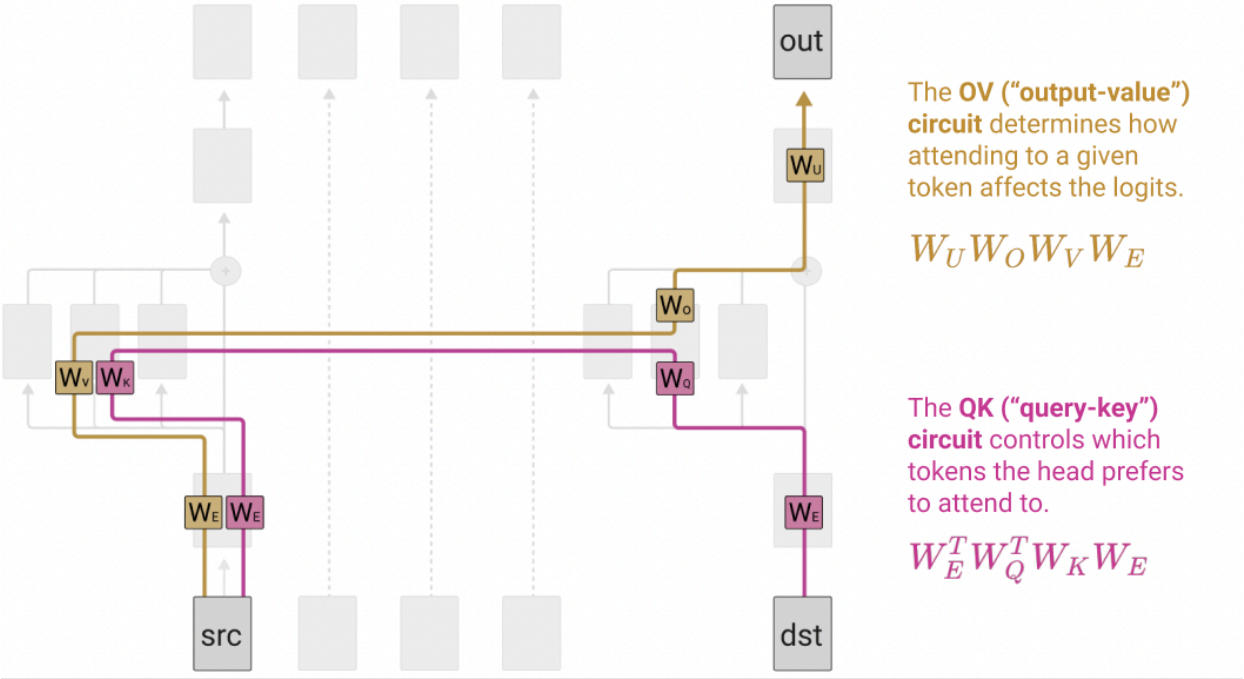
- A może opisywać własność kopiowania informacji z kanału i do j
- Transformer raczej nie przepisuje informacji sam do siebie, i.e. $A[i,i] = 0$

5. W tej sumie są dwa składniki
   - Pierwszy to bezpośrednie przetwarzanie kanału rezydualnego
     - Przetwarzanie transformera bez warstw ukrytych
   - Drugi to przetwarzanie informacji z głowic atencji

Obwody

#notes/circuit

1. Takie przetwarzanie tworzy obwód (circuit)



2. Dla każdej głowicy atencyjnej mamy

$$A^h \otimes (W_E W_{OV}^h W_E) = softmax(t \cdot W_E^T W_{QK}^h W_E \cdot t) \otimes (W_U W_{OV}^h W_E)$$

Wraz z $A^h = softmax(t \cdot W_E^T W_{QK}^h W_E \cdot t)$ mamy

- $W_E^T W_{QK}^h W_E$ to obwód query—key QK opisujący na ile *query* potrzebuje danego *key* sterując tym na ile dane tokeny są potrzebne
- $W_U W_{OV}^h W_E$ to obwód output—value OV opisujący na ile token wpłynie na wyjście

3. Obwody pozwalają na łatwiejsze interpretacje przetwarzania
4. wygodna interpretacja dla wykorzystania **lottery ticket hypothesis** Lottery ticket hypothesis: finding sparse, trainable neural networks, Frankle, Carbin ICLR'2019

5. Każda losowo zainicjowana wysoko wymiarowa sieć może zawierać w sobie szereg podsieć, które rozwiązują zadany problem lub podproblem
6. (Lub silniejsza wersja) każda nieprzeuczona wysoko wymiarowa sieć neuronowa będzie *zwykle* zawierać podsieć (obwód) rozwiązującą problem w sposób przybliżony, nawet bez douczania
7. Obwód QK:

   $$C_{QK}^h = W_E^T W_{QK}^h W_E = W_E^T (W_Q^T W_K) W_E$$

   , wyszukuje wcześniejszy istotny token ważny dla aktualnego

   $$[source] \overleftarrow{.....} [active]$$

8. Obwód OV

   $$C_{OV}^h = W_E W_{OV} W_E = W_E (W_O W_V) W_E$$

   przewiduje kolejny

   $$[active] \overrightarrow{} [next]$$

9. To tworzy razem

   $$[source] \overleftarrow{.....} [active] \overrightarrow{} [next]$$

   zwany ***skip-trigramem***

Kopiowanie

1. Głowice atencyjne przeznaczają dużo swojej aktywności na kopiowanie informacji z kanałów jednych to tokenów do kanałów innych
2. Zachowanie skip-gramów
3. Dla ustalonego tokenu *source*, obwód QK zwiększa (wyszukuje) prawdopodobieństwo tokenu aktualnego wybierając listę tych z najwyższym prawdopodobieństwem, np.

   $$[perfect] \overleftarrow{} [are, look, provides, is]$$

4. Obwód OV zwiększa prawdopodobieństwo następnego wyjściowego tokeny, jeśli poprzedni był aktualny, np.

   $$[looks] \overrightarrow{} [super, perfect]$$

   i stąd całe generowane wyrażenie

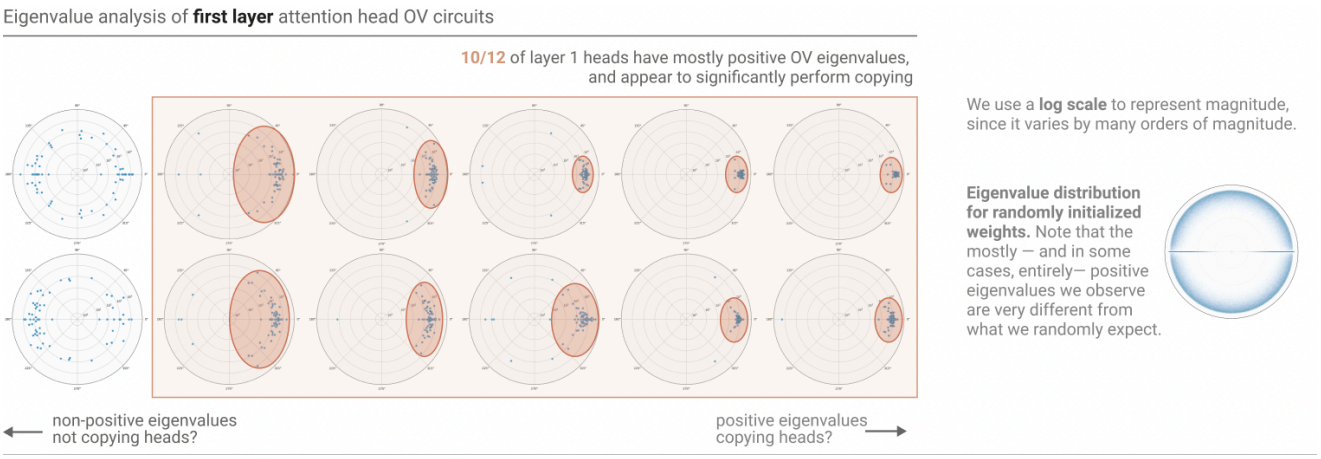
   $$perfect \dots is \ super, \text{ czy } \ super \dots is \ super$$

   - Tu widać przykład gdy początkowy token jest kopiowany do wyjściowego, bardzo częste zjawisko

5. Takim zjawiskiem może być dopełnianie, gdy token aktualny jest literą $Peter \dots [P] \overrightarrow{} [eter]$
6. Przykłady z przetwarzania Pythona
    1. Przewidywanie, że `else/elif/except` są najbardziej prawdopodobne dla kontekstu `\n\t\t\t ... \n\t\t`, gdzie aktualny fragment jest o jeden raz *mniej* wcięty
    2. Dla definicji metody w klasie `def … (` najbardziej prawdopodobne będzie `self`
    3. W `open ... ”,”` przewiduje parametr trybu otwarcia `[r / rb / w / wb]`
    4. `[for] ... [in]` daje `[range/enumerate/sorted/tqdm]`, z odpowiednimi prawdopodobieństwami zależnymi od innych parametrów
    5.
7. Kopiowanie informacji przez tokeny jest bardzo częste
8. token aktualny zwiększa prawdopodobieństwo pojawienia się kopii poprzedniego
    - Nie musi to być dokładna kopia
    - wpływ na odmianę, rodzaj, czas, liczbę pojedynczą / mnogą, etc.
9. Aa
10. postać skip-gramów może być uzależniona od kodowania pozycyjnego: kodowanie/pozycyjne, kodowanie/rotacyjne, kodowanie/

Analiza macierzy

1. Macierze OV i QK dla każdej głowicy mają bardzo wysokie wymiary, rzędu kilkudziesięciu/kilkuset tysięcy na kilkadziesiąt / kilkaset tysięcy a jednocześnie mają wyjątkowo niski rząd równy wymiarowi reprezentacji tokenu
    1. Można przeprowadzić analizę wartości własnych macierzy $W_U W_{OV}^h W_E$
    2. dodatnie liniowe kombinacje liniowe wektorów własnych zwiększają prawdopodobiępstwo logitów tych samych tokenów — czyli ich kopiowania
    3. Ujemne zmniejszają
2. potwierdza to analiza



[source: Anthropic]

3. Tutaj 10 na 12 macierzy tokenów w doświadczeniu miało w większości dodatnie wektory

własne, co sugeruje ich tendencje do kopiowania,
- zgodne z analizą ich tendencji do kopiowania, ale nie było to warunkiem wystarczającym
4. To jest chyba wszystko co się da powiedzieć o prostych jedno-warstwowych transformerach

Transformery dwu-warstwowe

1. W dwu-warstwowych transformerach (bez MLP) dochodzi dodatkowe złożenie relacji między atencjami w każdej z warstw; w całym modelu mamy sumę
   - Kanału rezydualny

     $$id \otimes W_U W_E$$

   - Przetwarzania dla głowic agencyjnych w każdej z warstw

     $$id \otimes \sum_{h \in \cup H_1 H_2} A^h \otimes (W_U W_{OV}^h W_E)$$

   - Kombinacji atencji dla każdej z warstw

     $$\sum_{h1 \in H_1} \sum_{h2 \in H_2} (A^{h_1} A^{h_2}) \otimes (W_U W_{OV}^{h2} W_{OV}^{h_1} W_E)$$

2. Najciekawsze będzie jednak rozwinięcie obwodu C_{QK}^h dla $h \in H^2$, ponieważ jego wejścia mogą pochodzić z różnych wejść, a stąd złożenia
   - sumy kanału rezydualnego wraz z przetwarzaniem *query* kanału rezydualnego w pierwszej warstwie

     $$id \otimes id \otimes W_E + \sum_{h_1 \in H_1} A^{h_1} \otimes id \otimes (W_{OV}^{h_1} W_E)^T$$

   - Przetwarzanie kanału rezydualnego przez obwód QK w drugiej warstwie

     $$id \otimes id \otimes W_{QK}^{h_2}$$

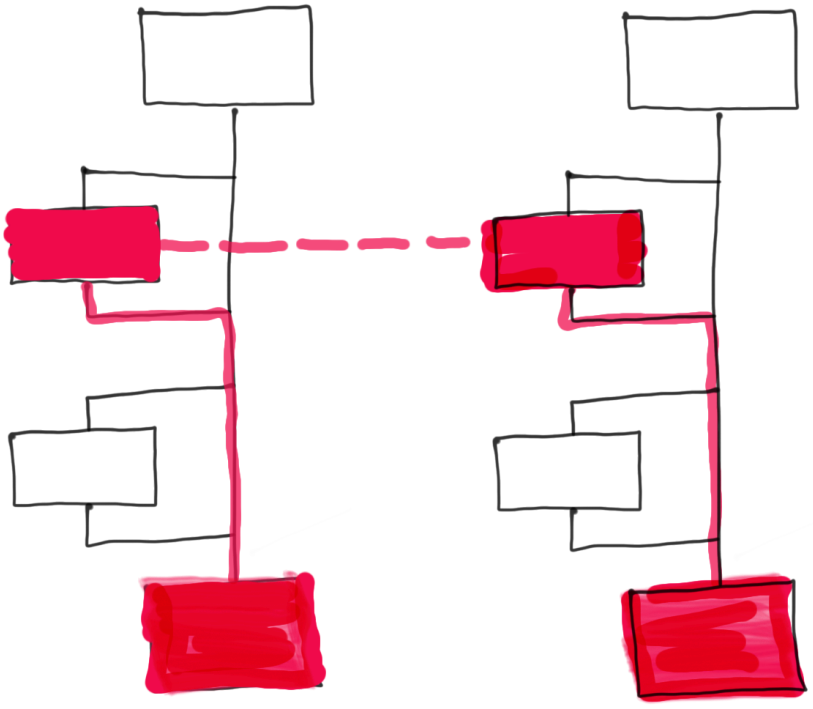
   - Sumy kanału rezydualnego wraz z przetwarzaniem *key* w pierwszej warstwie na początku warstwy drugiej

     $$id \otimes id \otimes W_E + \sum_{h_1 \in H_1} id \otimes A^{h_1} \otimes W_O V W_E$$

3. a to wszystko można przedstawić jako sumę
   - **Query** pełnego (tzn. przez jedną i drugą warstwę) kanału rezydualnego na wejściu drugiej warstwy

     $$id \otimes id \otimes (W_E W_{QK}^h W_E)$$

co **nie odpowiada** żadnemu złożeniu atencji jednej i drugiej warstwy

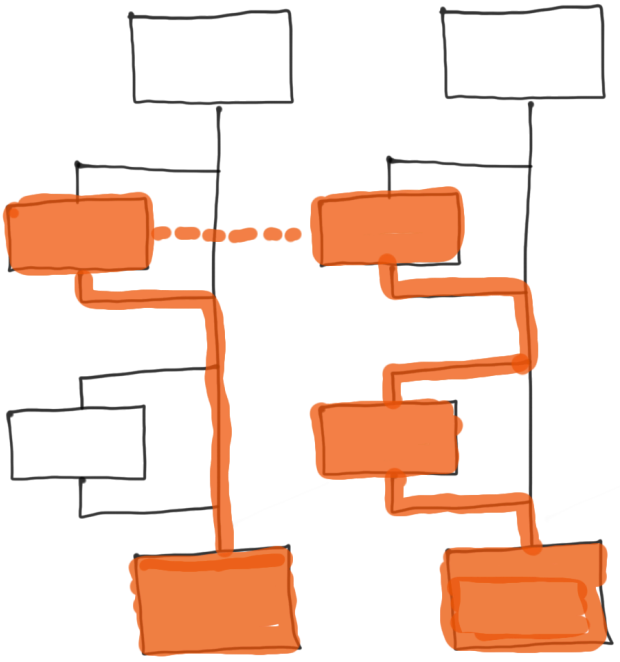


- Przetwarzania **query** (Q-composition) na wejściu kanału rezydualnego (zarówno kanału rezydualnego warstwy pierwszej jak i przez atencję warstwy pierwszej)

$$\sum_{h_q \in H_1} A^{h_q} \otimes id \otimes (W_E^T W_{OV}^{h_q} W_{QK}^h W_E)^T$$

to złożenie typu **query** gdzie poprzednia głowica atencyjna generuje query (prawa strona), a klucz pochodzi bezpośrednio z kanału rezydualnego (lewa

strona)

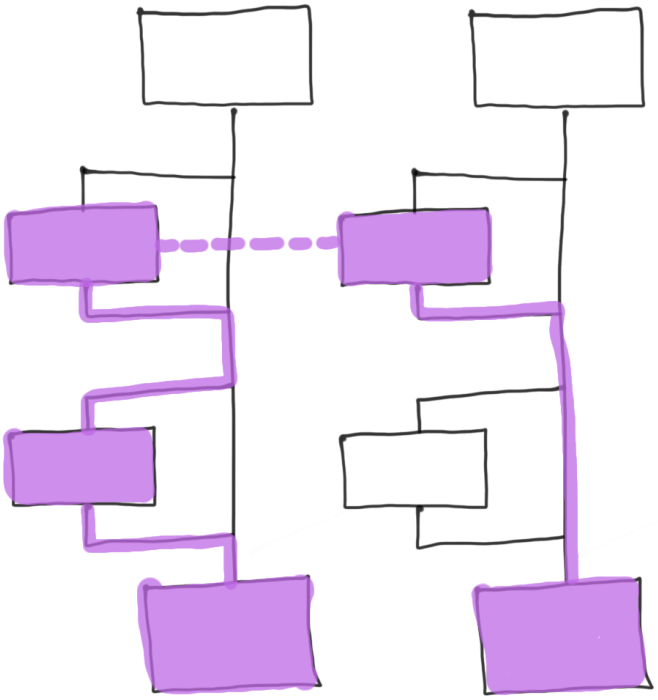


- Analogicznie, ale teraz przetwarzaniu **key** (K-composition) odpowiadająca

$$\sum_{h_1 \in H_1} id \otimes A^{h_1} \otimes (W_E^T W_{QK}^h W_{OV}^{h_1} W_E)$$

gdzie głowa atencyjna w pierwszej warstwie (lewa strona) jest wykorzystana do generowania klucza, a po stronie query (po prawej) wykorzystane są informacje z

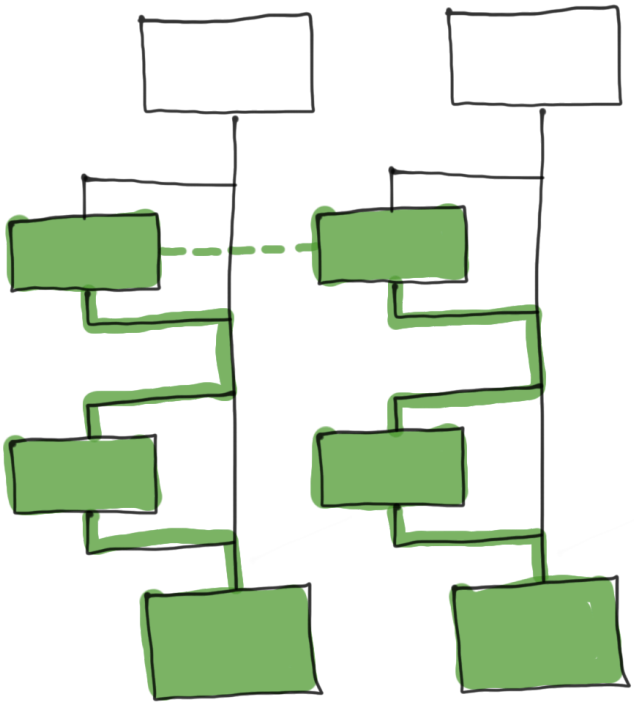
kanału rezydualnego



- Złożenia przetwarzania **query** (po prawej) i **key** (po lewej) w pierwszej i drugiej warstwie

$$\sum_{h_1 \in H_1} \sum_{h_j \in H_1} A^{h_i} \otimes A^{h_j} \otimes (W_E^T W_{OV}^{h_i T} W_{QK}^{h} W_{OV}^{h_j} W_E)$$

opisująca interakcję między złożeniami Q-composition oraz K-composition



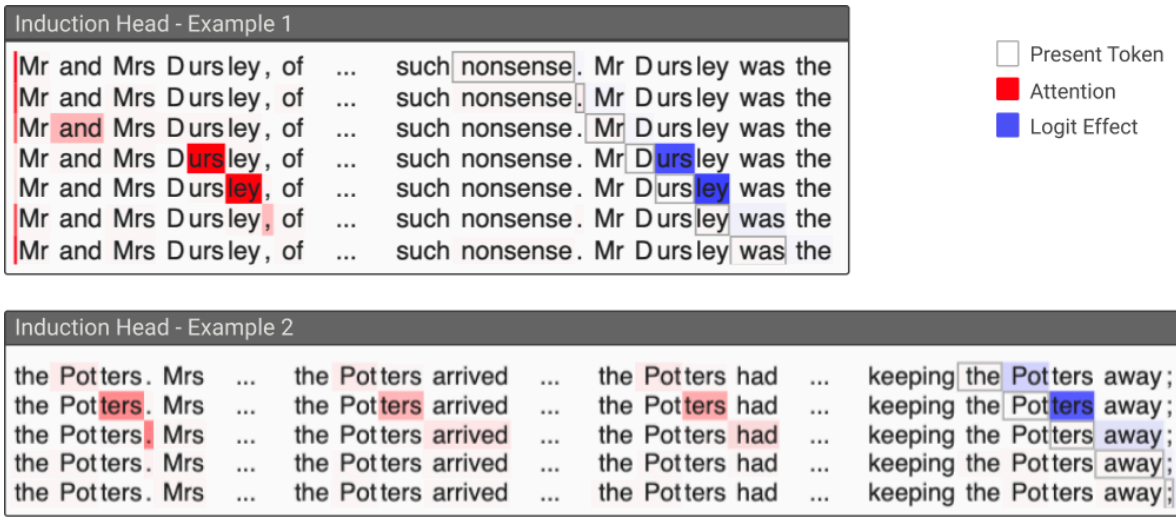
4. co oczywiście pozwala na bardziej złożone przetwarzanie

Rozpoznawane wzorce

1. Transformery z jedną warstwą tworzą układy przetwarzające

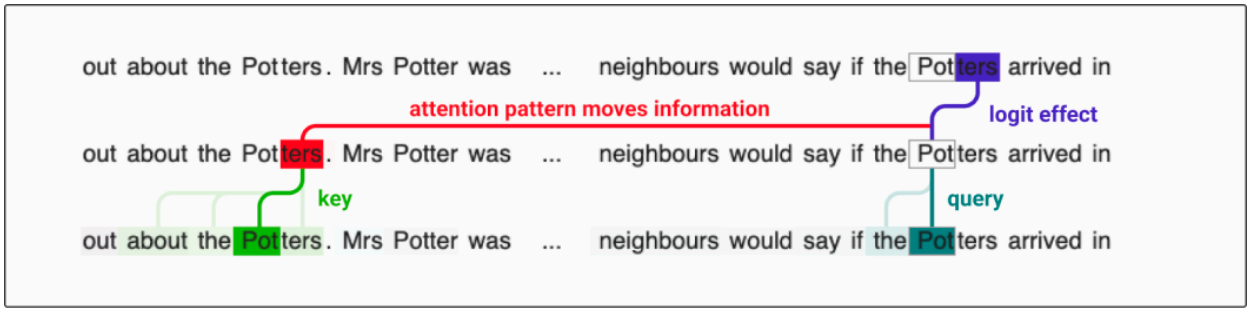
$$[source] \overleftarrow{\ldots\ldots} [active] \overrightarrow{\ } [next]$$

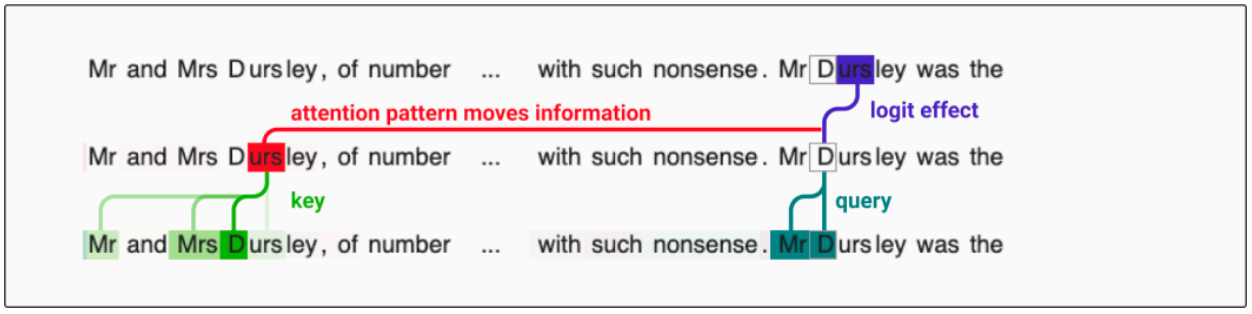
   w tym kopiujące typu $[Ala] \ldots [pani] \longrightarrow [Ala]$
2. czasem jest to układ $[pani\ Ala] \ldots [pani] \longrightarrow [Ala]$
3. Dwie warstwy wprowadzają układ współpracy dwóch głowic atencyjnych, często zwany głowicami **indukowanymi**
4. Głowica indukowana jest w stanie znaleźć w **jakim kontekście** token do powtórzenia był używany,
5. Układ głowicy indukowanej nie bazuje jedynie na statystyce czy token do skopiowania może następować po aktualnym, ale na kontekście jego użycie



[source: Anthropic]

6. Klucze Są obliczane z tokenów przesuniętych o jeden wstecz





[source: Anthropic]

- Niech aktualnym będzie token `[Pot]` czy też `[D]`
- **Query** poszukuje podobnych kluczy, biorąc pod uwagę cały kontekst poprzedzający `[about the][Pot]` dając następny `[ters]`, czy też `[Mr and Mrs][D]` dając `[urs]`

7.
8.
9. Pies ma kota

Jak sprawdzić jaki jest mierzalny wpływ kolejnych warstw

1. uruchomienie algorytmu odpowiadającego mierzeniu tzw. *ablacji*

```
1. Uruchomienie całego modelu i zapamiętanie wszystkich wzorców
atencyjnych
2. Powtórne uruchomienie z modyfikacjami
```

```
        1. Nie dodawać informacji z głowic atencyjnych i zapisać wyjście
        2. Obliczyć różnicę kosztu pomiędzy tym zapisanym wyjściem a
tensorem zerowym o odpowiednim rozmiarze
N. Kolejne uruchomienie z zapamiętaniem wzorców z dostatniego
uruchomienia, porównanie z zerowym tensorem i zapisanie różnicy kosztu
(loss)
```

4. To pozwoli określić ile informacji o ile użycie każdej z warstw atencyjnych zmniejsza błąd, przykładowo (żródło: Anthropic)
    1. sam kanał rezydualny, ucząc się jedynie predykcji bigramów, zmniejsza błąd o -1.8 nat (1 nat to około 1.443 bitów) względem równomiernej losowej predykcji
    2. sama głowica atencyjna w drugiej warstwie względem kanału rezydualnego
       2. -0.2 nat na każdy token
       3. -5.2 nat na 24 (przykładowo) tokeny
    3. Głowica indukowana względem tylko głowicy w drugiej warstwie
        1. -0.002 nat na każdy token,
        2. -0.3 nat na wszystkie kombinacje tokenów w każdej warstwie
5. Jednocześnie
6. Głowica atencyjna tylko w pierwszej warstwie to stosunkowo mały zysk
    1. -0.05 nat względem kanału rezydualnego i warstwy drugiej
    2. -1.3 nat względem samego kanału rezydualnego
7. głowica atencyjna tylko w drugiej warstwie — stosunkowo duży efekt ze względu na wykorzystanie znalezionej wiedzy w warstwy pierwszej
    1. -4.0 nat względem kanału rezydualnego i pierwszej warstwy
    2. -5.2 nat względem samego kanału rezydualnego
8. Pamiętajmy, że podczas gdy liczba warstw rośnie **liniowo**, tak liczba wirtualnych głowic atencyjnych rośnie kwadratowo dla dwóch warstw, i szybciej dla trzech, czterech, …

Czy to wszystko co możemy powiedzieć?

1. Z pewnością transformer złożony z warstw zawierających jedynie atencję bardzo ogranicza
2.