注意機構とトランスフォーマー

本谷 秀堅

名古屋工業大学

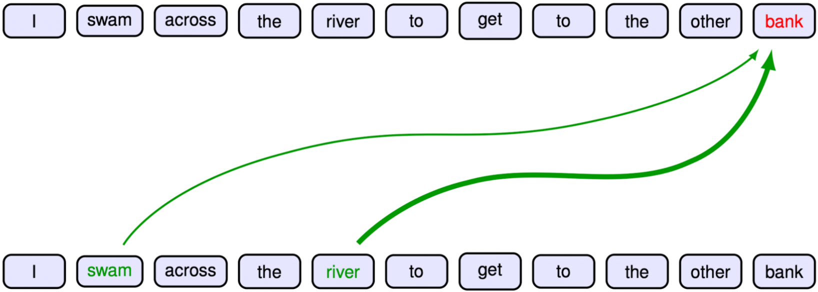
Transformer（トランスフォーマー）

注意機構 (Attention) を核とするニューラルネットワーク

言語データ（再掲）

言語データ

- シンボルの系列
- 離れたシンボル間に強い関係
- シンボル間の非対称な関係（例：対義語・類義語・包含関係）



Word2Vec による単語ベクトルの埋め込み

- 埋め込み先は文脈に依存せずに同じ
  - `bank` を表すベクトルは、堤のときも銀行のときも同じ

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x}$$

言語データと分布仮説（再掲）

分布仮説

単語の意味はその単語の周辺に現れる単語、すなわち文脈 (context) によって決まる

注意機構の目的

各単語の変換先を、系列中の他の単語に依存して変化させたい

- I swam across the river to get to the other bank.
  - “bank” は river の近くへと変換
- I walked across the road to get cash from the bank.
  - “bank” は money の近くへと変換

注意機構により文脈に依存して各単語の埋め込み先を変化させる

- 特徴ベクトルのことを「トークン (token)」と呼ぶ。

方針

文脈中のすべての単語を参照して、各単語の埋め込み先を決める

- 入力：N 単語。それぞれ D 次元のベクトル（トークン）で表現

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N \quad (\boldsymbol{x}_n \in \mathbb{R}^D)$$

- 出力：各単語の変換後の D 次元トークン

$$\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N \quad (\boldsymbol{y}_n \in \mathbb{R}^D)$$

文脈中の各単語の荷重平均を埋め込み先とする

$$\boldsymbol{y}_n = \sum_{m=1}^{N} a_{nm} \boldsymbol{x}_m$$

注意プーリング (Attetion Pooling)

注意プーリング (pooling: 集約するの意)

a_{nm}: 注意の強さ　($a_{nm} > 0, \ \sum_{m=1}^{N} a_{nm} = 1$)

$$\boldsymbol{y}_n = \sum_{m=1}^{N} a_{nm} \boldsymbol{x}_m$$

注意の強さを文脈に応じて決めたい

- I swam across the river to get to the other bank.
  - "bank" は river と似た方向へと変換
- I walked across the road to get cash from the bank.
  - "bank" は money と似た方向と変換

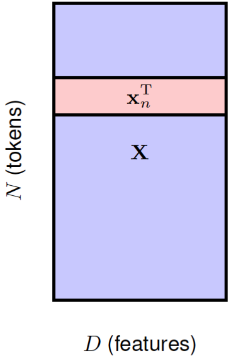
記法

- N 単語
- 各単語はベクトルで表現 $\boldsymbol{x}_n \in \mathbb{R}^D$
- 文脈は行列で表現：$\boldsymbol{X} \in \mathbb{R}^{N \times D}$

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n]^\top$$

- 出力も行列で表現：$\boldsymbol{Y} \in \mathbb{R}^{N \times D}$

$$\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_n]^\top$$



注意の大きさを文脈で決める（案 1）

もともと似たベクトルどうしを強い重みで

$$a_{nm} = \frac{\exp(\boldsymbol{x}_n^\top \boldsymbol{x}_m)}{\sum_k \exp(\boldsymbol{x}_n^\top \boldsymbol{x}_k)}$$

$$\downarrow$$

$$\boldsymbol{A} = \mathrm{SM}(\boldsymbol{X}\boldsymbol{X}^\top) \in \mathbb{R}^{N\times N},$$

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} = \mathrm{SM}(\boldsymbol{X}\boldsymbol{X}^\top) \in \mathbb{R}^{N\times N}.$$

- トークン間の類似度を内積で評価する場合の式
- 行列 $\boldsymbol{A}$ の (n, m) 要素が a_{nm}
- Softmax は各行で計算
- $\boldsymbol{X}\boldsymbol{X}^\top$ は対称行列
- a_{nm} **に「学習」の余地無し。柔軟性に欠ける。**

注意の大きさを文脈で決める（案 2）

注意の大きさの決め方を学習する

$$\tilde{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{U}$$

上記 $\boldsymbol{U} \in \mathbb{R}^{D \times D}$ を学習する

$$\boldsymbol{A} = \mathrm{SM}(\boldsymbol{X}\boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}^\top),$$

$$\boldsymbol{Y} = \mathrm{SM}(\boldsymbol{X}\boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}^\top)\boldsymbol{X}\boldsymbol{U}.$$

- 案 1 より柔軟
- $\boldsymbol{X}\boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}^\top$ は対称行列
- Softmax 関数を適用すると対称ではなくなるが大小関係は維持される
- **包含関係、対義語、類義語、指示代名詞など複雑な関係の表現には自由度が不足**

注意の大きさを文脈で決める（注意機構）

注意の大きさの決め方：クエリ・キー・バリューの考え方を採用する

$$a_{nm} = \frac{\exp(\boldsymbol{q}_n^\top \boldsymbol{k}_m)}{\sum_k \exp(\boldsymbol{q}_n^\top \boldsymbol{k}_k)}$$

$$\uparrow$$

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{X}\boldsymbol{W}^{(\mathrm{q})}, \\ \boldsymbol{K} &= \boldsymbol{X}\boldsymbol{W}^{(\mathrm{k})}, \\ \boldsymbol{V} &= \boldsymbol{X}\boldsymbol{W}^{(\mathrm{v})}. \end{aligned}$$

行列 $\boldsymbol{W}^{(\mathrm{q})}$, $\boldsymbol{W}^{(\mathrm{k})}$, $\boldsymbol{W}^{(\mathrm{v})}$ を学習により決める。
$\boldsymbol{W}^{(\mathrm{q})}, \boldsymbol{W}^{(\mathrm{k})} \in \mathbb{R}^{D \times D_{\mathrm{k}}}$, $\boldsymbol{W}^{(\mathrm{v})} \in \mathbb{R}^{D \times D_{\mathrm{v}}}$

$$\boldsymbol{Y} = \mathrm{SM}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}$$

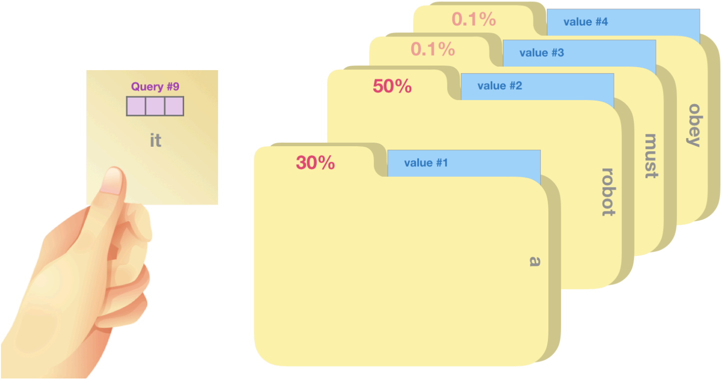
（復習）クエリ・キー・バリュー

- キー (key) とバリュー (value: 値) の組のデータ集合
- クエリ (query) による検索

例：

- データ集合:
  $\mathcal{D} = \{($“夏目”, “漱石”$)$, $($“森”, “鴎外”$)$, $($“中島”, “敦”$)$, $($“松尾”, “芭蕉”$)\}$
- 検索：query = “森”
  検索結果：value = “鴎外”

（復習）クエリ・キー・バリュー



https://jalammar.github.io/illustrated-gpt2/

注意機構 (1/2)

$$\boldsymbol{Y} = \mathrm{SM}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}$$

1. 行列 $\boldsymbol{W}^{(q)}$, $\boldsymbol{W}^{(k)}$ を学習する
   - key-vector の次元を D_k で表すと、いずれも $\mathbb{R}^{D \times D_k}$
2. $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{(q)}$, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{(k)}$
   - n 番目の単語のクエリ ($\boldsymbol{q}_n$) とキー ($\boldsymbol{k}_n$) は次のとおり

$$\boldsymbol{q}_n^\top = \boldsymbol{x}_n^\top \boldsymbol{W}^{(q)}, \quad \boldsymbol{k}_n^\top = \boldsymbol{x}_n^\top \boldsymbol{W}^{(k)}$$

3. n 番目の単語を変換するために $mm = 1, 2, \ldots, N$ 番目の単語を参照

$$\boldsymbol{q}_n^\top \boldsymbol{k}_m = (\boldsymbol{Q}\boldsymbol{K}^\top)_{nm}$$

4. $\boldsymbol{Q}\boldsymbol{K}^\top$ の各行で Softmax を計算

$$a_{nm} = \frac{\exp(\boldsymbol{q}_n^\top \boldsymbol{k}_m)}{\sum_k \exp(\boldsymbol{q}_n^\top \boldsymbol{k}_k)}$$

注意機構 (2/2)

$$\boldsymbol{Y} = \mathrm{SM}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}$$

- 行列 $\boldsymbol{W}^{(v)} \in \mathbb{R}^{D\times D}$ を学習する
- バリュー (value) の計算：$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{(v)}$
    - n 番目の単語のバリュー, $\boldsymbol{v}_n$

$$\boldsymbol{v}_n^\top = \boldsymbol{x}_n^\top \boldsymbol{W}^{(v)}$$

- トークンの更新
    - n 番目の単語更新用の注意：$\boldsymbol{a}_n^\top = [a_{n1}, a_{n2}, \ldots, a_{nN}]$
    - n 番目の単語のトークンの更新

$$\boldsymbol{y}_n^\top = \boldsymbol{a}_n^\top \boldsymbol{V}$$

注意のスケールの調整

exp の中の絶対値が大きくなることを防ぐ。（絶対値が大きいと勾配がゼロに近づき学習に失敗する。）

$$a_{nm} = \frac{\exp(\boldsymbol{q}_n^\top \boldsymbol{k}_m)}{\sum_k \exp(\boldsymbol{q}_n^\top \boldsymbol{k}_k)}$$

$$\downarrow$$

$$a_{nm} = \frac{\exp\left(\frac{\boldsymbol{q}_n^\top \boldsymbol{k}_m}{\sqrt{D_k}}\right)}{\sum_k \exp\left(\frac{\boldsymbol{q}_n^\top \boldsymbol{k}_k}{\sqrt{D_k}}\right)}$$

D_k は $\boldsymbol{q}$, $\boldsymbol{k}$ の次元

まとめ：注意機構

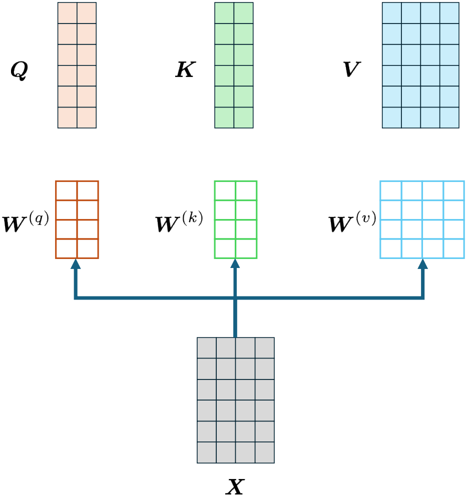
$$\boldsymbol{Y} = \mathrm{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{SM}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D_k}}\right)\boldsymbol{V}$$

Attention Head

文脈に依存して埋め込み先を変える機構：注意機構

Attention Head とも呼ぶ

- 入力文全体を見渡して「どこに注目するか」を判断するモジュール

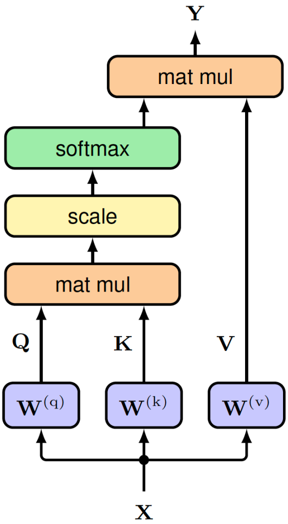


Attention Head

文脈に依存して埋め込み先を変える機構：注意機構

Attention Head とも呼ぶ

- 入力文全体を見渡して「どこに注目するか」を判断するモジュール

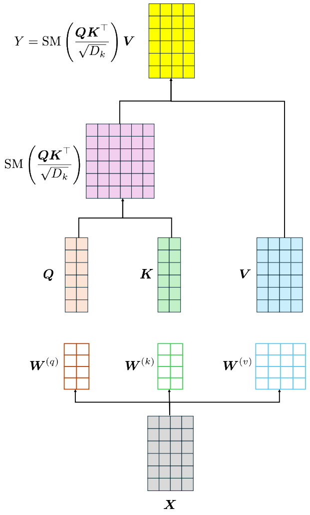


Attention Head

文脈に依存して埋め込み先を変える機構：注意機構

Attention Head とも呼ぶ

- 入力文全体を見渡して「どこに注目するか」を判断するモジュール

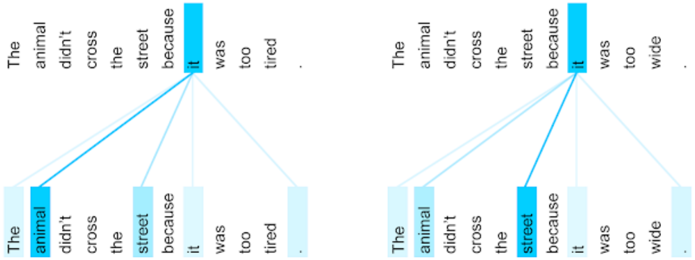


Attention Headで実現できたこと

- 文脈に依存して単語の埋め込み先を変える
- 注意の強弱を文脈に依存して決定

例

`The animal did not cross the street because it was too tired.`
`The animal did not cross the street because it was too wide.`



注意機構の性質

重要：$\boldsymbol{W}^{(q)}$, $\boldsymbol{W}^{(k)}$, $\boldsymbol{W}^{(v)}$ が注意の規準を定める（学習後は規準固定）

- 出力 $\boldsymbol{y}_n$ は入力 $\boldsymbol{v}_n$ の線形和

$$\boldsymbol{y}_n = \sum_{m=1}^{N} a_{nm} \boldsymbol{v}_m$$

- $\boldsymbol{v}_n$ は $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ の線形変換

$$\boldsymbol{V} = \boldsymbol{X} \boldsymbol{W}^{(o)}$$

- $\boldsymbol{y}_n$ は入力 $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ の線形変換
- 非線形性は注意の強度, a_{nm}, 経由で導入 ($\boldsymbol{x}_n$ に依存・Softmax)

注意機構の性質

重要：$\boldsymbol{W}^{(q)}$, $\boldsymbol{W}^{(k)}$, $\boldsymbol{W}^{(v)}$ が注意の規準を定める（学習後は規準固定）

- 出力 $\boldsymbol{y}_n$ は入力 $\boldsymbol{v}_n$ の線形和

$$\boldsymbol{y}_n = \sum_{m=1}^{N} a_{nm} \boldsymbol{v}_m$$

- $\boldsymbol{v}_n$ は、$(\boldsymbol{W}^{(o)})^\top$ の列ベクトルの線形和

$$\boldsymbol{v}_n^\top = \boldsymbol{x}_n^\top \boldsymbol{W}^{(o)} \to \boldsymbol{v}_n = (\boldsymbol{W}^{(o)})^\top \boldsymbol{x}_n$$

つまり

$$\boldsymbol{v}_n = \sum_{i=1}^{D} x_{ni} \boldsymbol{w}_i^{(o)}$$

ただし、$\boldsymbol{x}_n = [x_{n1}, x_{n2}, \ldots, x_{nD}]^\top$、

$$(\boldsymbol{W}^{(o)})^\top = [\boldsymbol{w}_1^{(o)}, \boldsymbol{w}_2^{(o)}, \ldots, \boldsymbol{w}_D^{(o)}].$$

注意機構の性質

重要：$\boldsymbol{W}^{(q)}$, $\boldsymbol{W}^{(k)}$, $\boldsymbol{W}^{(v)}$ が注意の規準を定める（学習後は規準固定）

- 出力 $\boldsymbol{y}_n$ は入力 $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_D$ が張る部分空間に限定される

$$\boldsymbol{y}_n = \sum_{m=1}^{N} a_{nm} \boldsymbol{v}_m$$

- 出力 $\boldsymbol{y}_n$ は入力 $\boldsymbol{x}_n$ の線形変換
- 出力 $\boldsymbol{y}_n$ は学習で得られる $\boldsymbol{w}_1^{(o)}, \boldsymbol{w}_2^{(o)}, \ldots, \boldsymbol{w}_D^{(o)}$ が張る部分空間に限定される。
- 非線形性は注意の強度, a_{nm}, 経由で導入 ($\boldsymbol{x}_n$ に依存・Softmax)

注意機構の性質

重要：$\boldsymbol{W}^{(q)}$, $\boldsymbol{W}^{(k)}$, $\boldsymbol{W}^{(v)}$ が注意の規準を定める（学習後は規準固定）

- a_{nm} $(m = 1, 2, \ldots, N)$ は、スケール $\sqrt{D_k}$ で調整したあとでも、ひとつの要素だけ 1 に近い値となり、その他はゼロに近い値となることが多い。
  - ソフトマックス関数の exp が主要因
  - 注意機構は「特定の規準のみ」で注意の大小を決める傾向あり

Attention Head の注意の規準

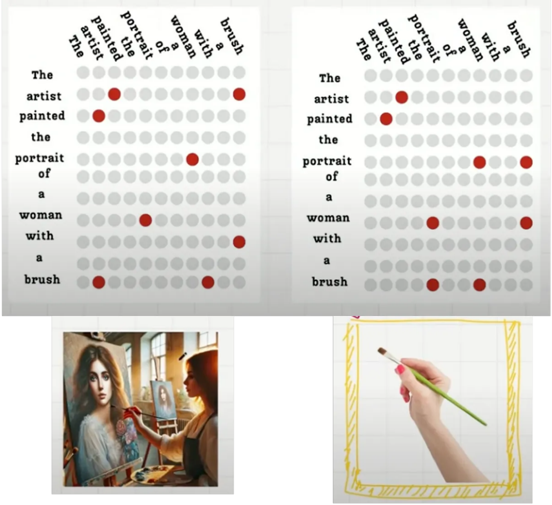
重要：$\boldsymbol{W}^{(q)}$, $\boldsymbol{W}^{(k)}$, $\boldsymbol{W}^{(v)}$ が注意の規準を定める（学習後は規準固定）

例：`The artist painted the portrait of a woman with a brush.`

上記の文の意味はどっち？

- "a woman with a brush" を描いた
- "a woman" をブラシ（筆）を使って描いた

Attention Head の注意の規準



case 1 : Painting a woman with a “brush”

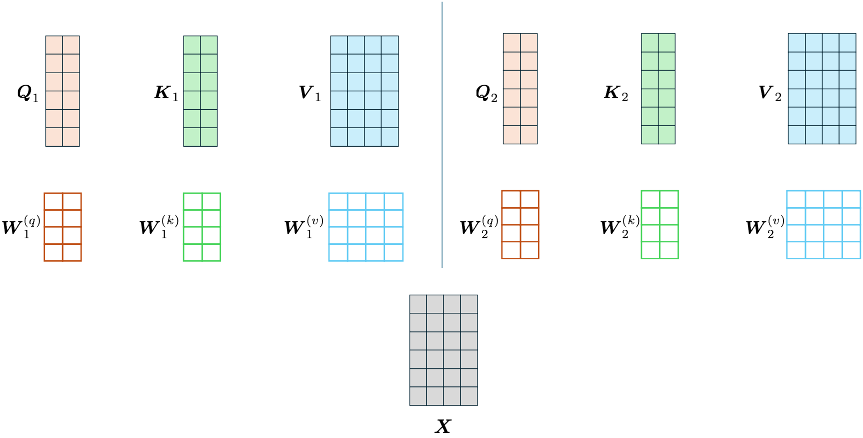
case 2: Painting of a “woman with a brush”

https://medium.com/@shravankoninti/a-visual-explanation-of-multi-head-attention-6399d86fe51c

Attention Head を複数

- 注意の強弱を決める基準はひとつでは足りない
- 時制 / 語彙 / 語順 / ...
- Multi-Head Attention: 複数の基準 ($\boldsymbol{W}^{(q)}$, $\boldsymbol{W}^{(k)}$, $\boldsymbol{W}^{(v)}$) を用意

マルチヘッド注意機構



マルチヘッド注意機構

H 個の注意機構を用いることにする。h 番目 $(h = 1, 2, \ldots, H)$ の注意機構は次式でトークンを変換する。

$$\boldsymbol{H}_h = \mathrm{Attention}(\boldsymbol{Q}_h, \boldsymbol{K}_h, \boldsymbol{V}_h)$$

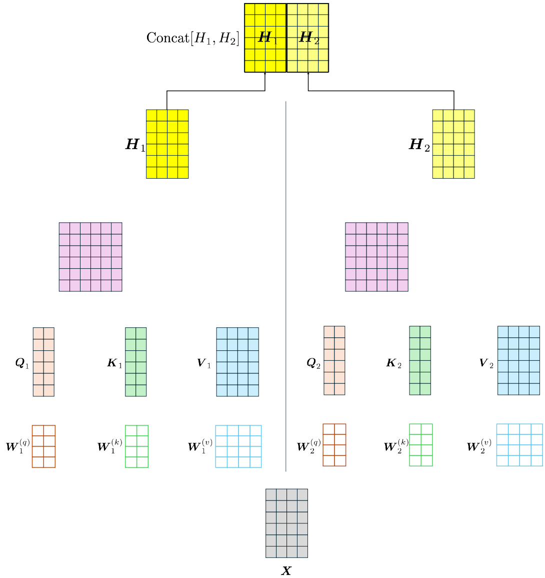
クエリ・キー・バリューは次式で計算する

$$\boldsymbol{Q}_h = \boldsymbol{X}\boldsymbol{W}_h^{(\mathrm{q})},$$
$$\boldsymbol{K}_h = \boldsymbol{X}\boldsymbol{W}_h^{(\mathrm{k})},$$
$$\boldsymbol{V}_h = \boldsymbol{X}\boldsymbol{W}_h^{(\mathrm{v})}.$$

$$\underline{\boldsymbol{Y}(\boldsymbol{X}) = \mathrm{Concat}[\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_H]\boldsymbol{W}^{(\mathrm{o})}}$$

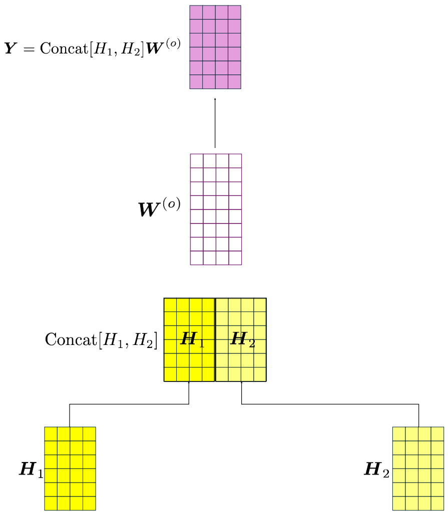
ただし、$\boldsymbol{H}_h \in \mathbb{R}^{N \times D_\mathrm{v}}$, $\mathrm{Concat}[\boldsymbol{H}_1, \ldots, \boldsymbol{H}_H] \in \mathbb{R}^{N \times HD_\mathrm{v}}$, $\boldsymbol{W}^{(\mathrm{o})} \in \mathbb{R}^{HD_\mathrm{v} \times D}$, $\boldsymbol{Y}(\boldsymbol{X}) \in \mathbb{N} \times \mathbb{D}$。多くの場合、$D_\mathrm{v} = D/H$

マルチヘッド注意機構



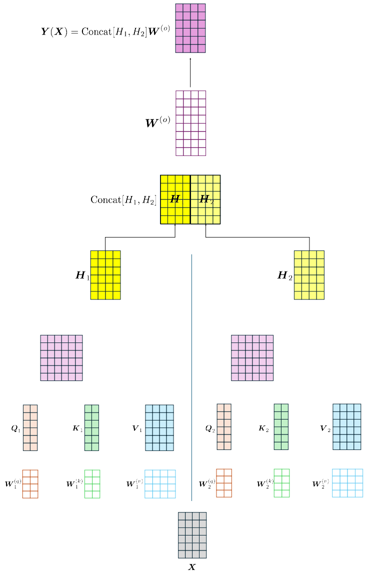
$\mathrm{Concat}[H_1, H_2]$
$\boldsymbol{H}_1$
$\boldsymbol{H}_2$
$\boldsymbol{H}_1$
$\boldsymbol{H}_2$
$\boldsymbol{Q}_1$
$\boldsymbol{K}_1$
$\boldsymbol{V}_1$
$\boldsymbol{Q}_2$
$\boldsymbol{K}_2$
$\boldsymbol{V}_2$
$\boldsymbol{W}_1^{(q)}$
$\boldsymbol{W}_1^{(k)}$
$\boldsymbol{W}_1^{(v)}$
$\boldsymbol{W}_2^{(q)}$
$\boldsymbol{W}_2^{(k)}$
$\boldsymbol{W}_2^{(v)}$
$\boldsymbol{X}$

マルチヘッド注意機構



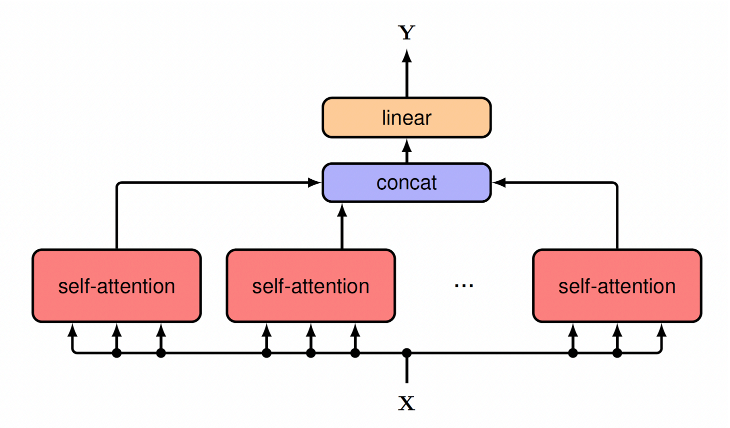
$\boldsymbol{Y} = \mathrm{Concat}[H_1, H_2]\boldsymbol{W}^{(o)}$
$\boldsymbol{W}^{(o)}$
$\mathrm{Concat}[H_1, H_2]$
$\boldsymbol{H}_1$
$\boldsymbol{H}_2$
$\boldsymbol{H}_1$
$\boldsymbol{H}_2$

マルチヘッド注意機構



$\boldsymbol{Y}(\boldsymbol{X}) = \mathrm{Concat}[H_1, H_2]\boldsymbol{W}^{(o)}$
$\boldsymbol{W}^{(o)}$
$\mathrm{Concat}[H_1, H_2]$
$\boldsymbol{H}_1$
$\boldsymbol{H}_2$
$\boldsymbol{Q}_1$
$\boldsymbol{K}_1$
$\boldsymbol{V}_1$
$\boldsymbol{Q}_2$
$\boldsymbol{K}_2$
$\boldsymbol{V}_2$
$\boldsymbol{W}_1^{(q)}$
$\boldsymbol{W}_1^{(k)}$
$\boldsymbol{W}_1^{(v)}$
$\boldsymbol{W}_2^{(q)}$
$\boldsymbol{W}_2^{(k)}$
$\boldsymbol{W}_2^{(v)}$
$\boldsymbol{X}$

マルチヘッド注意機構


Y
linear
concat
self-attention
self-attention
...
self-attention
X

準備：レイヤ正規化 (Layer Normalization)

- 誤差逆伝播の計算を安定化させる効果あり

それぞれの特徴ベクトル $\boldsymbol{y}_n \in \mathbb{R}^D$ を「正規化」する。

$$\boldsymbol{y}_n = [y_{n1}, y_{n2}, \dots, y_{nD}]^\top$$

準備：平均と分散

$$\mu_n = \frac{1}{D} \sum_{i=1}^{D} y_{ni}$$

$$\sigma_n^2 = \frac{1}{D} \sum_{i=1}^{D} (y_{ni} - \mu_n)^2$$

正規化

$$\hat{y}_{ni} = \frac{y_{ni} - \mu_n}{\sqrt{\sigma_n + \delta}}$$

記法

$$\mathrm{LayerNorm}[\boldsymbol{y}_n] = [\hat{y}_{n1}, \hat{y}_{n2}, \dots, \hat{y}_{nD}]^\top$$

後処理 (1/3)

学習効率を上げる工夫: レイヤ正規化と、残差表現

マルチヘッドの出力: $\boldsymbol{Y}(\boldsymbol{X})$

$$\boldsymbol{Y}(\boldsymbol{X}) = \mathrm{Concat}[\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_H]\boldsymbol{W}^{(\mathrm{o})}$$

元の入力を足して正規化。$Y(\boldsymbol{X})$ は元との差分のみ表現すれば良くなった。

$$\boldsymbol{Z} = \mathrm{LayerNorm}[\boldsymbol{Y}(\boldsymbol{X}) + \boldsymbol{X}]$$

後処理 (2/3)

注意機構の出力は $\boldsymbol{v}$ が張る線形部分空間に限定。柔軟性を改善するために Multi-Layer Perceptron を導入。例えば、全結合層を 2 つ追加する。

$$\boldsymbol{Z}' = \mathrm{MLP}[\boldsymbol{Z}]$$

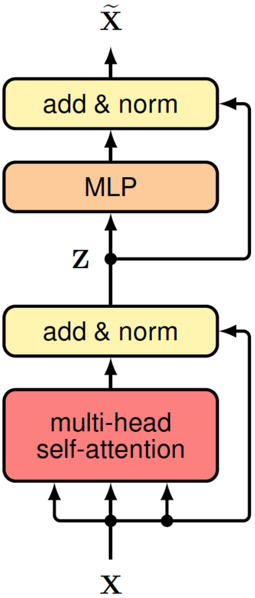
- 同じ MLP を $\boldsymbol{Z}$ の各行に適用する。つまり、各単語のトークンに同じ MLP を適用する。

後処理 (3/3)

学習効率を上げる工夫：レイヤ正規化と、残差表現

$$\tilde{\boldsymbol{X}} = \mathrm{LayerNorm}[\boldsymbol{Z}' + \boldsymbol{Z}] = \mathrm{LayerNorm}[\mathrm{MLP}[\boldsymbol{Z}] + \boldsymbol{Z}]$$

トランスフォーマーのアーキテクチャ（符号器・1層分)



位置の表現

トークンの入力順序が変わっても出力は変わらない。このままでは、位置・順序の情報が結果に反映されない。

（再掲: 系列の先頭から n 番目のトークンの変換）

$$\boldsymbol{y}_n = \sum_{m=1}^{N} a_{nm} \boldsymbol{x}_m,$$

$$a_{nm} = \frac{\exp\left(\frac{\boldsymbol{q}_n^\top \boldsymbol{k}_m}{\sqrt{D_k}}\right)}{\sum_k \exp\left(\frac{\boldsymbol{q}_n^\top \boldsymbol{k}_k}{\sqrt{D_k}}\right)}.$$

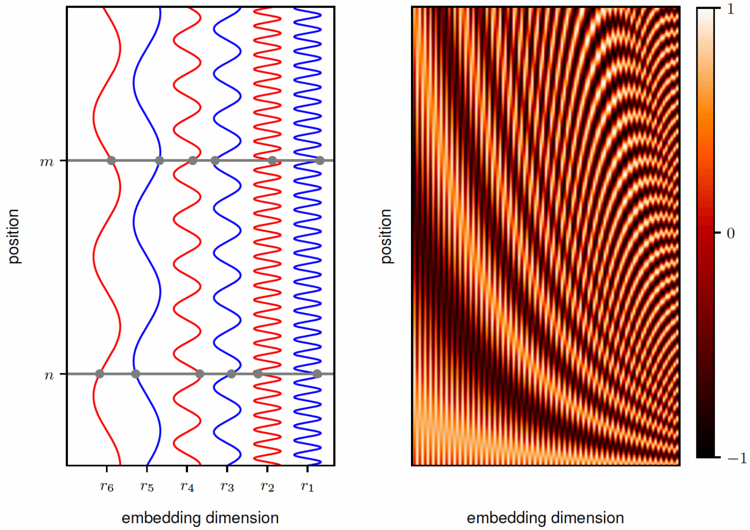
位置の表現 $\boldsymbol{r}_n = [r_{n1}, r_{n2}, \ldots, r_{nD}]^\top$ を生成して、入力トークンに足す

$$\tilde{\boldsymbol{x}}_n = \boldsymbol{x}_n + \boldsymbol{r}_n$$

最大系列長より充分大きい数, L（例えば 10,000)

$$r_{ni} = \begin{cases} \sin\left(\frac{n}{L^{i-1/D}}\right), & \text{if } i \text{ is odd}, \\ \cos\left(\frac{n}{L^{(i-2)/D}}\right), & \text{if } i \text{ is even}. \end{cases}$$

位置の表現



position
m
n
r_6 r_5 r_4 r_3 r_2 r_1
embedding dimension
position
embedding dimension
1
0
−1

位置の表現

位置の表現 $\boldsymbol{r}$ の性質

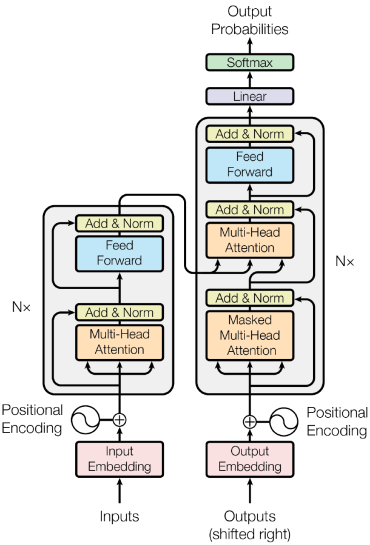
- $\|\boldsymbol{r}_n\| = D/2$
- 位置が一意に表現される
- 近い位置では似たベクトル、遠い位置では似ていないベクトル
- $\boldsymbol{r}_n$ と $\boldsymbol{r}_{n+k}$ の関係は n に依らない (k だけに依存する) 線形変換で表現可能 ($\phi = k/L^{j/D}$)

$$\sin\left(\frac{n+k}{L^{j/D}}\right) = \sin\left(\frac{n}{L^{j/D}} + \phi\right)$$

$$= \cos(\phi)\sin\left(\frac{n}{L^{j/D}}\right) + \sin(\phi)\cos\left(\frac{n}{L^{j/D}}\right)$$

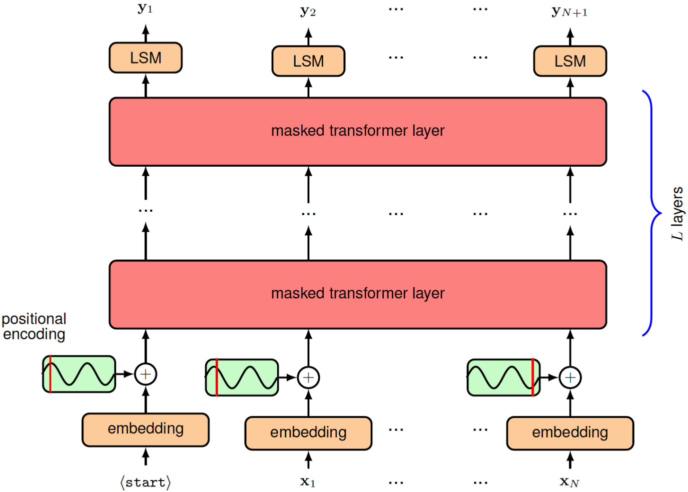
- 系列内での相対位置が重要なときには、ニューラルネットワークが ($\boldsymbol{W}^{(\mathrm{q})}$, $\boldsymbol{W}^{(\mathrm{k})}$ で) 学習可能

トランスフォーマのアーキテクチャ



Vaswani, Attention is All You Need, 2017

予告：GPT のアーキテクチャ



LSM: Linear SoftMax = 線形多クラス識別器