

学習済み transformer モデルのコンテキスト拡張 における位置エンコーディング

情報数理システム分野 研究会

平田 蓮

2024 年 5 月 22 日

Attention 機構

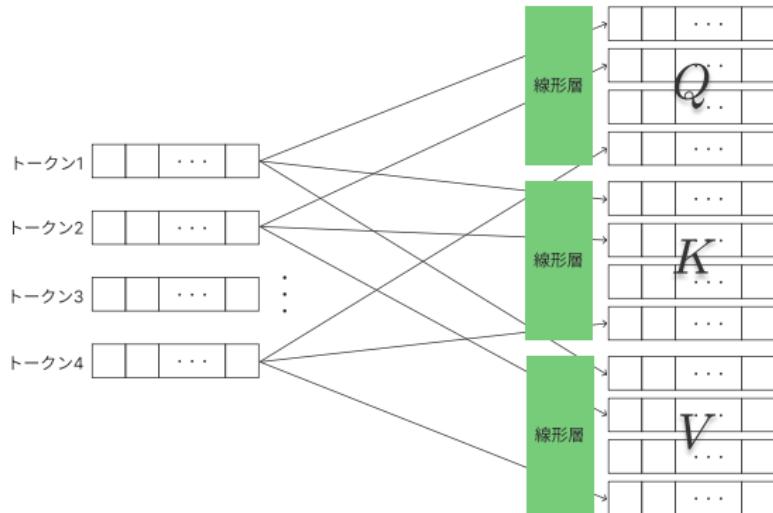
Attention 機構

入力トークンの特徴量を、他トークンとの「関係性」を加味しながら計算する仕組み

最終的なトークンの特徴量は、Attention 機構を繰り返し通すことで形成される

Attention 機構

各 Attention 機構では、前の段階で作られた特徴量からトークンごとに線形層を用いて Query, Key, Value の三種類のベクトルを構成する



Attention 機構

それぞれのベクトルの意味のイメージは次の通り

Query 自身のトークンと関連性の強いトークンを参照する

Key 各トークンの Query に対して適切な自身の Value を返す

Value トークンの「値」

Attention 機構

Q, K を用いて次の式に基づいて Attention 行列 A を計算する

$$A = \text{Softmax} \left(\frac{Q^t K}{\sqrt{d}} \right)$$

	トークン1	トークン2	トークン3	トークン4
トークン1				
トークン2				
トークン3				
トークン4				

$= \text{Softmax}$

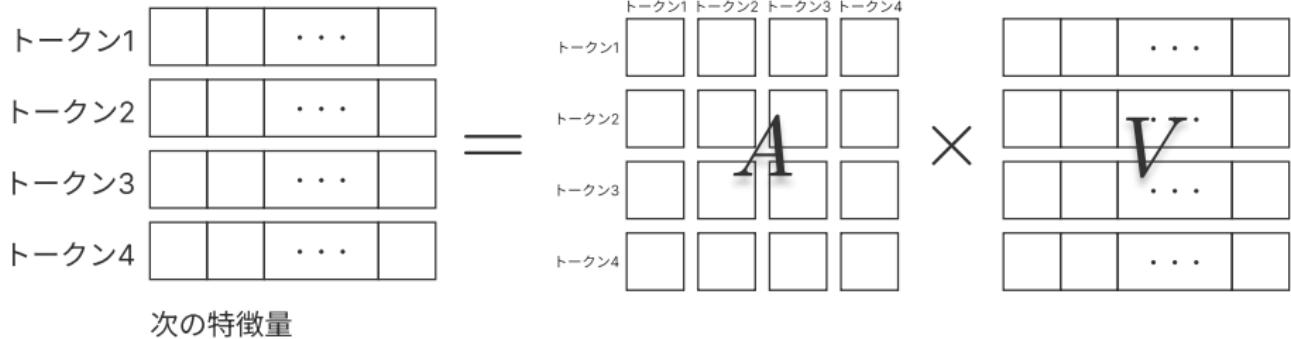
$$\frac{\begin{matrix} \text{トークン1} & \text{トークン2} & \text{トークン3} & \text{トークン4} \\ \hline \text{トークン1} & & & \\ \text{トークン2} & & & \\ \text{トークン3} & & & \\ \text{トークン4} & & & \end{matrix}}{\sqrt{d}}$$

$Q^t K$

Q と K を掛けることで各トークン同士の関係の強さを計算し、それを正規化したものが A である

Attention 機構

A を計算した後、 AV の各行が各トークンの特徴量として次の Attention 機構に渡される



位置エンコーディング

Attention 機構の問題点

トークンの順番を入れ替えても同様の特徴量が得られてしまうという問題がある。よって、上述のように計算をするとトークン同士の位置情報が失われる。

この問題を解消するため、 A を計算する際に位置情報を加味した計算を追加で行う。これを位置エンコーディングと呼ぶ。

位置エンコーディング - 絶対位置エンコーディング

次の式で計算されるベクトルを各トークンの特徴量に加算する

$$\text{PositionalEncoding} = \begin{cases} \sin\left(\frac{i}{10000^{\frac{2j}{d}}}\right) & (j \equiv 0 \pmod{2}) \\ \cos\left(\frac{i}{10000^{\frac{2j}{d}}}\right) & (j \equiv 1 \pmod{2}) \end{cases}$$

i トークンの位置

j 位置エンコーディングの要素の位置

位置エンコーディング - 相対位置エンコーディング

絶対位置エンコーディングの改善点 1

時系列データは絶対位置よりも、他のトークンとの相対位置が重要

日本語の文章で考えると、「私のペンが…」と「〇〇。私のペンが…」という 2 つのシーケンスでは「ペン」の位置が違うので絶対位置エンコーディングでは別の値となるが、実際は「私の」の次にあるという相対的な位置情報が大事である

位置エンコーディング - 相対位置エンコーディング

絶対位置エンコーディングの改善点 2

学習データに無い長さの系列を処理できない

学習データに無い長さの系列をモデルに与えた時、後ろの方の絶対位置エンコーディングは学習されていない状態であるため、適切な推論ができなくなることが確認されている

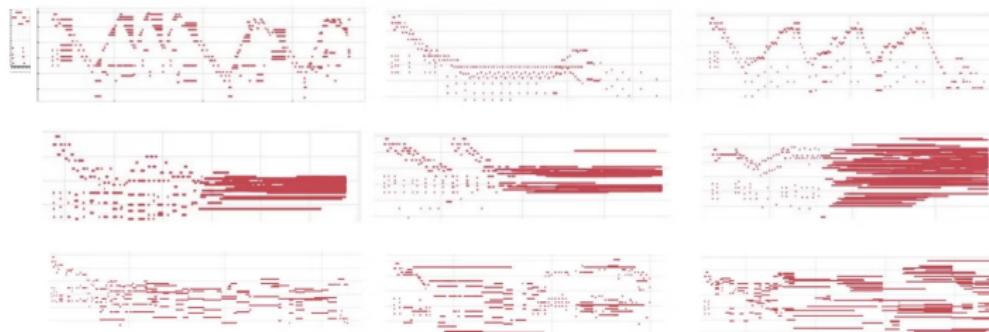
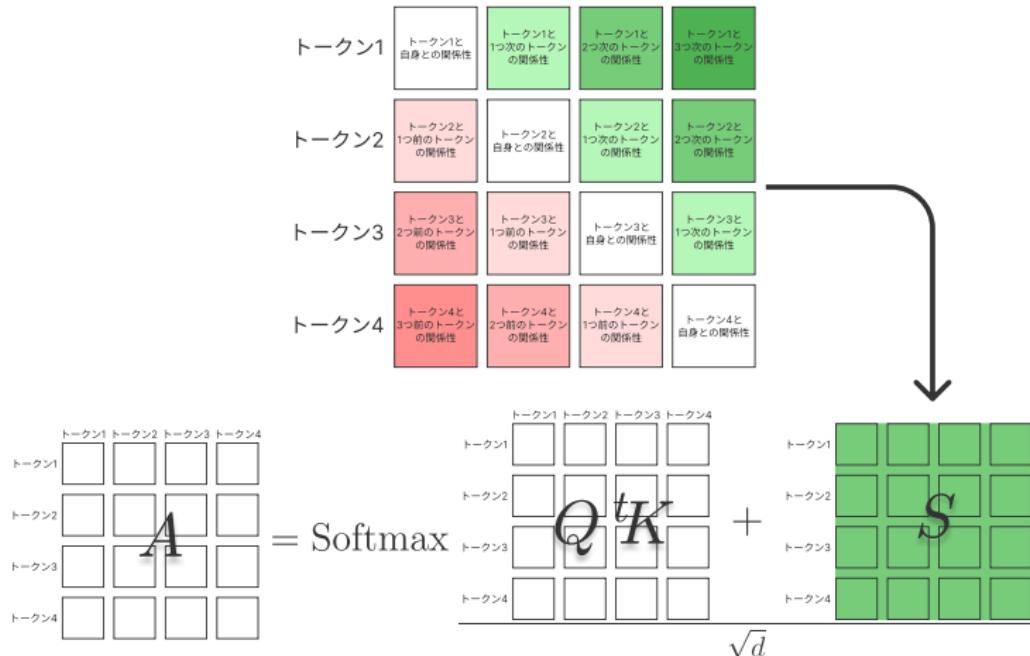


Figure 4: Comparing how models continue a prime (top left). Repeated motives and structure are seen in samples from Transformer with relative attention (top row), but less so from baseline Transformer (middle row) and PerformanceRNN (LSTM) (bottom row).

位置エンコーディング - 相対位置エンコーディング

相対位置エンコーディングは下図に示すような行列であり、Attention 行列を計算する際に Q^tK に加算される。



位置エンコーディング - 相対位置エンコーディング

相対位置エンコーディングの行列は、各 Attention 機構が固有に保持する相対位置ごとの埋め込みと、各トークンの Query の内積で計算される。

$$S_{i,j} = Q_i E_{i-j}$$

$S_{i,j}$ S の (i, j) 要素

Q_i i 番目のトークンの Query

E_{i-j} あるトークンとそれから $i - j$ の位置との相対位置埋め込み。学習可能パラメータ

位置エンコーディング - RoPE

位置エンコーディングを加算ではなく乗算で行うアプローチ。LLaMAなどで用いられており、現在主流である。

Query と Key に回転行列をかけて、ベクトルを回転させることで位置情報に乗せる。

RoPE の特徴

- トークンの位置ごとに回転角を増やしていくため、絶対位置情報を反映可能
- 距離の近いトークンの回転角同士が成す角が小さいため、相対位置情報も反映可能

位置エンコーディング - RoPE

回転行列の計算

$$R = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{\frac{d}{2}} & -\sin m\theta_{\frac{d}{2}} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{\frac{d}{2}} & \cos m\theta_{\frac{d}{2}} \end{pmatrix}$$

$$\theta_i = 10000^{-\frac{2(i-1)}{d}} \quad \left(i = 1, 2, \dots, \frac{d}{2} \right)$$

m トークン位置

これを用いて計算した ${}^tQ' = R^tQ, {}^tK' = R^tK$ により Attention 行列を計算する。

コンテキスト拡張

現在、学習済みのモデルのコンテキストを拡張する際の位置エンコーディングは、Scaled RoPE が主流に用いられている

Scaled RoPE

RoPE をコンテキスト長の拡張に応じて引き伸ばす手法

上述の m を元のコンテキスト長 n と新しいコンテキスト長 n' を用いて $\frac{n}{n'}$ 倍する。

展望 - 別アプローチによる位置エンコーディングの拡張

Scaled Rope の問題点

コンテキスト長を大きくした際に位置エンコーディングの「差」が小さくなってしまうため、極端にコンテキスト長を長くした際に位置エンコーディングの意味が薄れてしまう

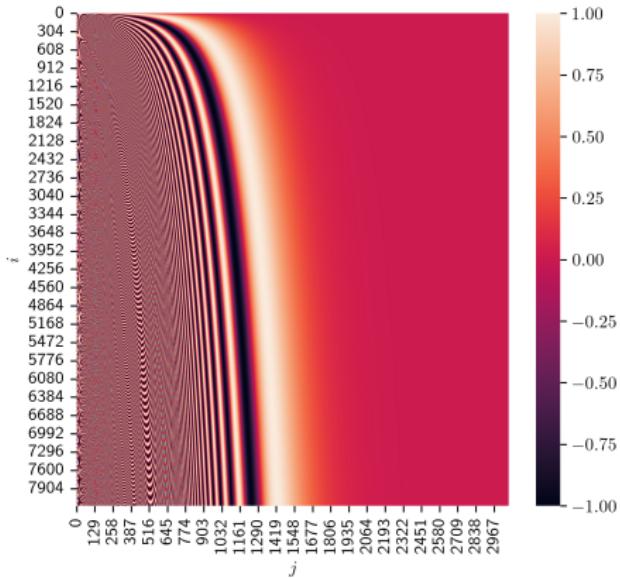
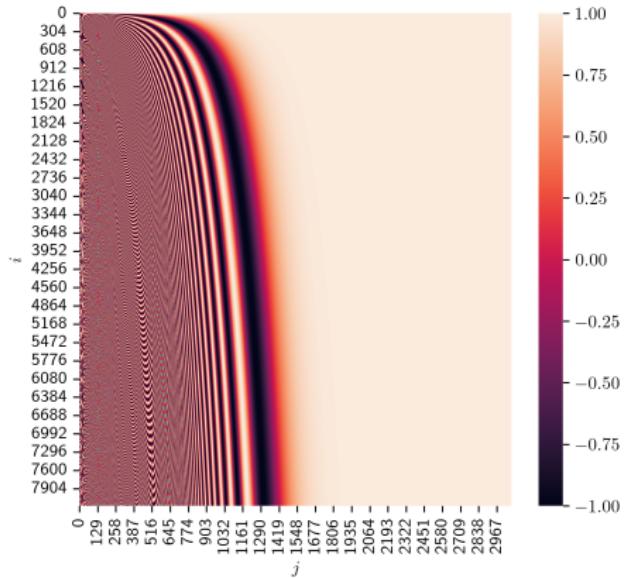
これを回避した再学習ありのコンテキスト拡張の手法も提案されているが、学習済みの位置エンコーディングを拡張する手法は見つけられていないため、調査を継続したい。

展望 - 位置エンコーディングのスパース性

gpt-3 の埋め込み次元は 3072, 最大トークン長は 8192 である¹。このときの絶対位置エンコーディングと RoPE の具体的な値を見てみる。

¹<https://platform.openai.com/docs/guides/embeddings/what-are-embeddings>

展望 - 位置エンコーディングのスパース性



図：絶対位置エンコーディングの値。 \cos (左) と \sin (右)。

展望 - 位置エンコーディングのスパース性

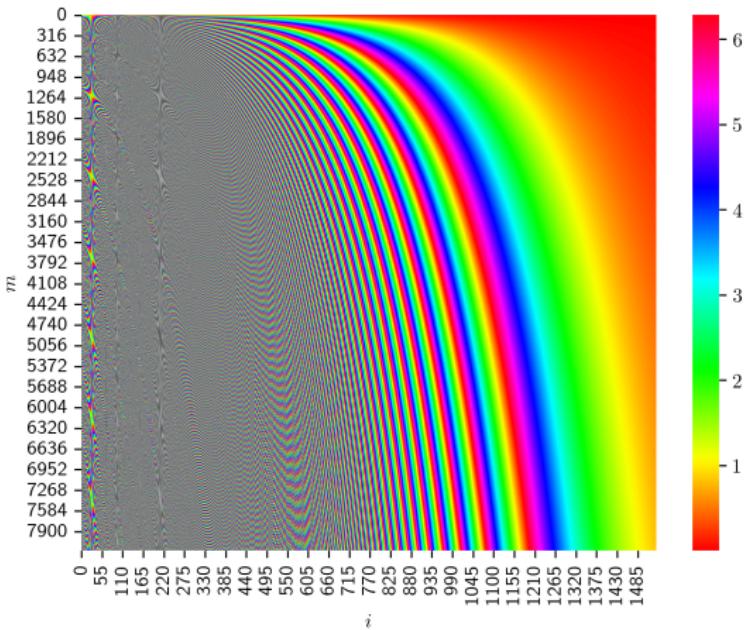


図: RoPE の回転角 $m\theta_i$

展望 - 位置エンコーディングのスパース性

図を見ると、各ベクトルにおいて、高次元の部分には位置エンコーディングの情報が乗っていないことがわかる。これを踏まえると、次の二つの可能性を感じるため、調査を続けたい。

位置エンコーディングの情報量を増やして性能が向上する場合
よりベクトル全体に情報を乗せる「効率的」な位置エンコーディング

情報量が少ない位置エンコーディングで十分な場合

よりスパースな、計算量を削減できる位置エンコーディング