

GPT と BERT トランسفォーマーによる復号器と符号器

本谷 秀堅

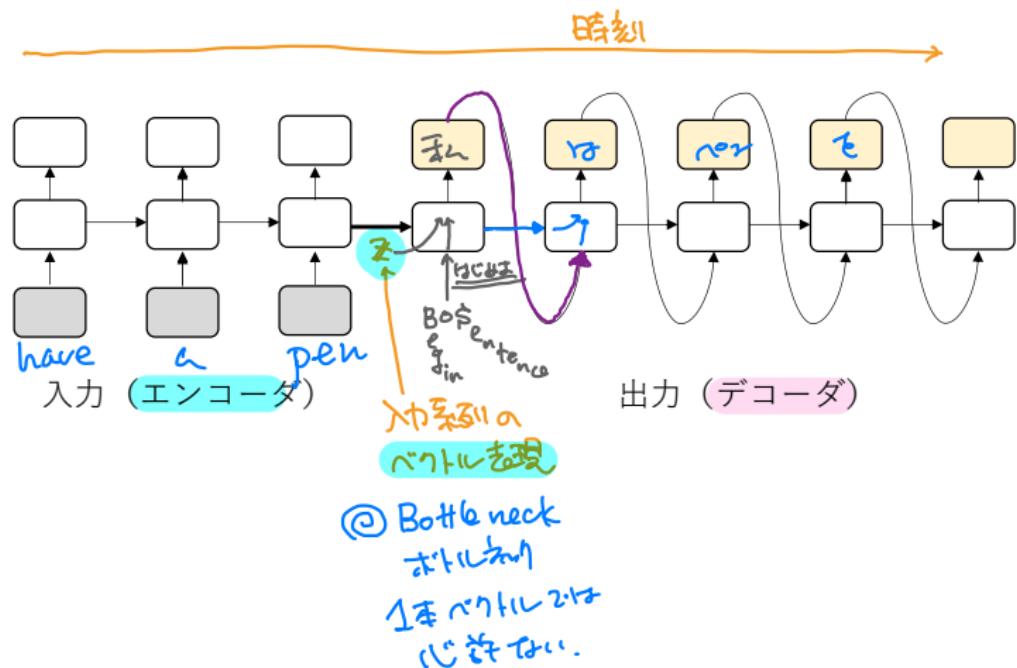
名古屋工業大学

GPT: Generative Pretrained Transformer



トランスフォーマーによる復号器（デコーダ）

復習：RNNによる系列変換



自己回帰モデル (Auto-regressive Model)

- 復習：CBoW (Continuous Bag of Words): 語順無関係

$$h = \frac{1}{C} \sum_i h_i$$

- 自己回帰モデル: 語順を考えたモデル構築に使える
 - 言語モデルの核となる考え方

$$p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n-1})$$

系列 .

- 文章が長くなるにつれて条件の違いが指数的に増大
- 直前 L 単語のみに依存すると仮定して問題を単純化

$$p(\underbrace{\mathbf{x}_3, \mathbf{x}_4, \dots}_{\text{右辺}}, \underbrace{\mathbf{x}_1, \mathbf{x}_2}_{\text{左辺}}) \quad \text{重なり分布}$$

例: $L = 2$ のとき (bi-gram)

$$p(x_1, x_2, \dots, x_N) = p(x_1)p(x_2|x_1) \prod_{n=3}^N p(x_n|x_{n-2}, x_{n-1})$$

$L=2$ 単語のみ
 1 文字。
 one-hot
 vector
 (?)

- 確率モデルを構築できれば自己回帰しながら文章を生成できる（最初の L 単語は指定）
例：

cat through shipping variety is made the aid emergency ...

- モデル, $p(x_n|x_{n-1}, x_{n-2})$, は学習文より「頻度」に基づき学習

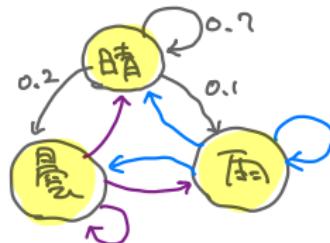
キトカ $p(x|$ "cat", "through"), $p(x|$ "through", "shipping")

- 語彙数が $K = 10^5$ のとき条件は 10^{10} 通り

復習：マルコフモデル (Hidden Markov Model)

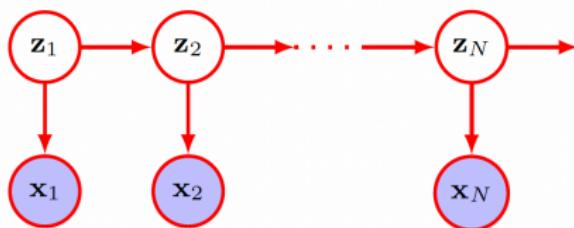
マルコフモデル

- 統計モデル
- 複数の/離散の「状態」を持つ
- 次の時刻の状態は、現時刻の状態のみに依存して確率的に遷移する



隠れマルコフモデル

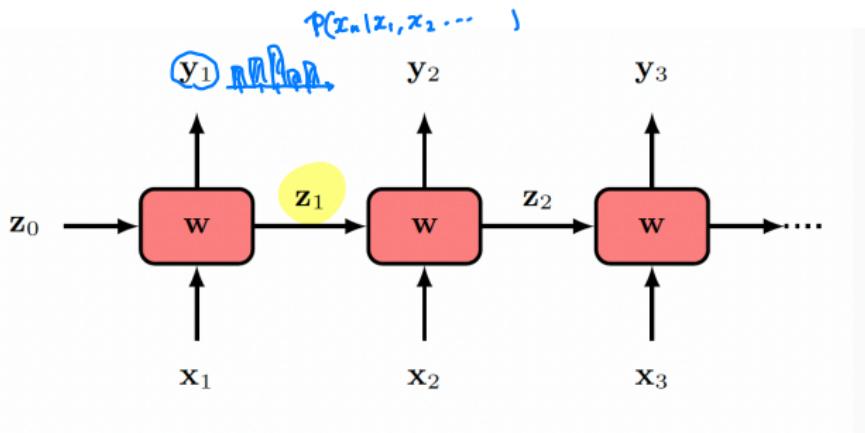
- 「状態」を観測できない



復習：Recurrent Neural Network

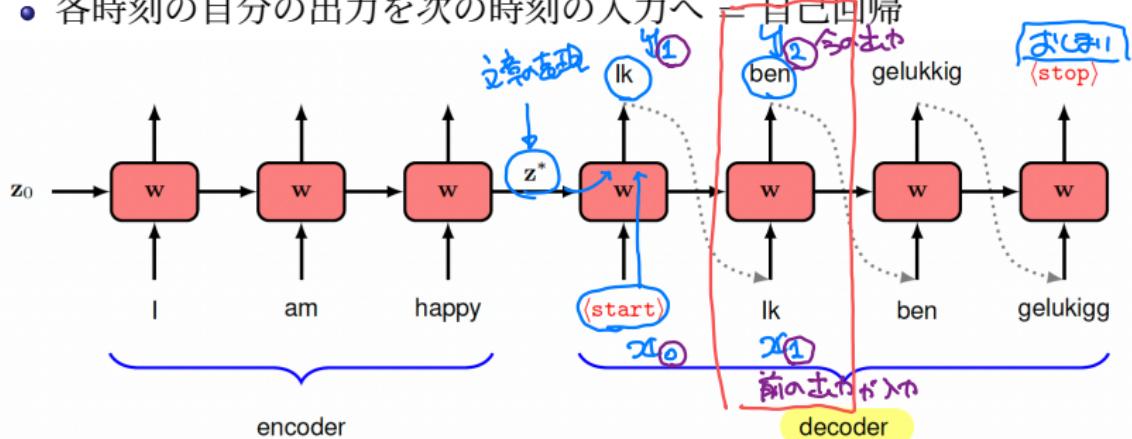
潜在表現
中間層の出力

- 「状態」は離散ではなく連続値, $z^t \in \mathbb{R}^D \rightarrow$ 頻度から脱却
- 遷移は確率的ではなく確定的
- 出力も確率的ではなく確定的



復習：RNN二つによる系列変換と文章生成

- 文章をベクトル, z^* , へと埋め込むエンコーダ（符号器）
- 文を生成するデコーダ（復号器）
 - 文章を表現するベクトル, z^* , と開始記号<start>を入力
 - 各時刻の自分の出力を次の時刻の入力へ = 自己回帰



- ボトルネック問題: 文章が長くなるほど z^* による要約が困難に

トランスフォーマーを用いる言語モデル

- エンコーダ: 単語系列が入力 → 単一のベクトルを出力
 - 例: 映画評を読んで、ポジティブ・ネガティブを判断
- デコーダ: 単一のベクトルが入力 → 単語系列を出力
 - 例: 画像を入力して、脚注を出力
- 系列変換器: 単語系列が入力 → 単語系列を出力
 - 例: 英語から日本語へと翻訳

トランスフォーマーによる復号器

- 目標：トランスフォーマーを用いて次式の自己回帰モデルを実現すること

手書き見込み

$$p(x_n | x_1, x_2, \dots, x_{n-1})$$

自分の過去の歴史
に依存して
今の出力分布を
推定する。

- $n - 1$ 個のトークンの系列・特徴ベクトルの系列を受け取る。
- n 番目のトークンの確率分布を推定する。 ←
- 推定した分布からトークンをひとつサンプルして、 x_n とする。
- 次は $p(x_{n+1} | x_1, x_2, \dots, x_n)$ を推定する
- 条件文の長さの上限は定めておく

GPT 以前から
ある
考え方。

GPT: Generative Pre-trained Transformer

- トランスフォーマー層を積み重ねたアーキテクチャ

- 入力: x_1, x_2, \dots, x_N ($x_n \in \mathbb{R}^D$)
- 出力: $\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_N$ ($\tilde{x}_n \in \mathbb{R}^D$)

- 各単語より得たトークン $\tilde{x}_n \in \mathbb{R}^D$ から K 個単語に対する事後確率を推定する

$$\mathbf{Y} = \text{SM}(\tilde{\mathbf{X}} \mathbf{W}^{(p)})$$

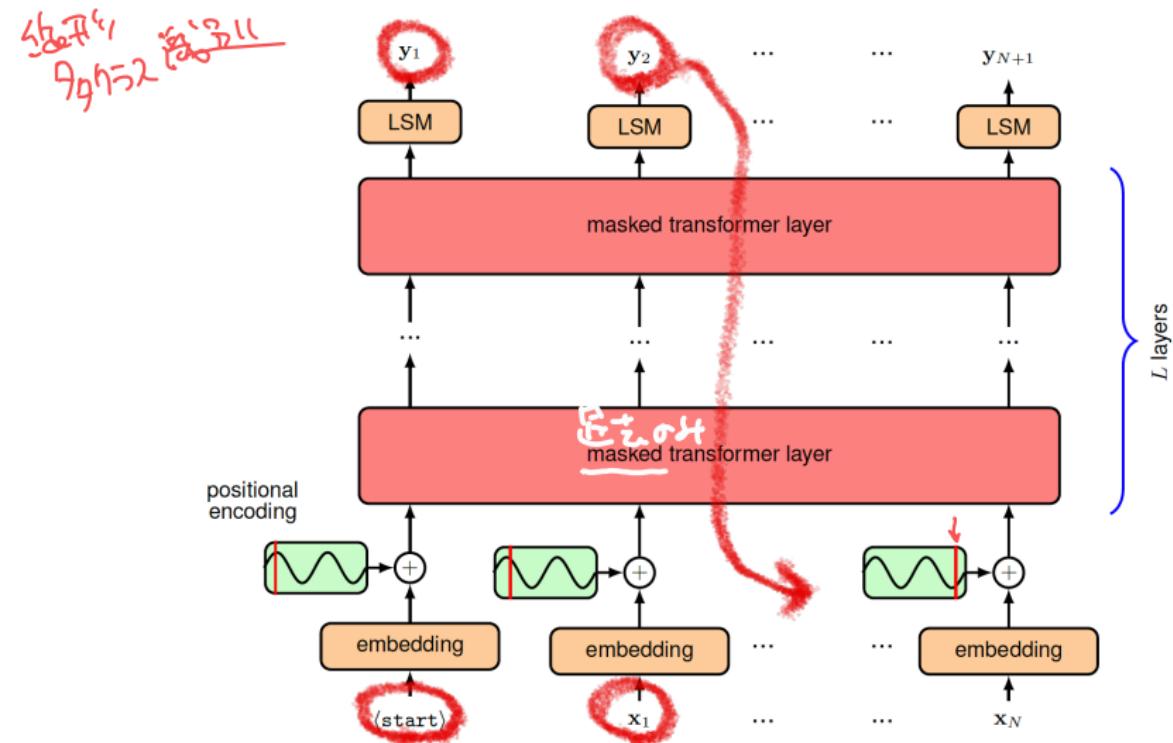
ただし、

The diagram shows a token \tilde{x} (input) being processed by a matrix W (weights) to produce output probabilities y (output). A blue double-headed arrow labeled "单語辨別" (word discrimination) connects the input \tilde{x} and the weight matrix W .

$$\begin{aligned}\mathbf{Y} &= [y_1, y_2, \dots, y_N]^\top \in \mathbb{R}^{N \times V}, \\ \tilde{\mathbf{X}} &= [\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_N]^\top \in \mathbb{R}^{N \times D}, \\ \mathbf{W}^{(p)} &\in \mathbb{R}^{D \times K}.\end{aligned}\tag{1}$$

それぞれのトークン \tilde{x}_n は同一の多クラス識別器に入力される

GPTのアーキテクチャ



注：入力に対して出力が左にひとつずれている

vec.

LSM: Linear SoftMax = 線形多クラス識別器



意味.

多量の文章より、自動的に教師信号を作り、学習する。

- pretext task：「次の単語を予測する」

うふべのり27.

例：I swam across the river to get to the other bank.

- 入力 I swam across、正解 the
- 入力 I swam across the、正解 river
- 入力 I swam across the river、正解 to

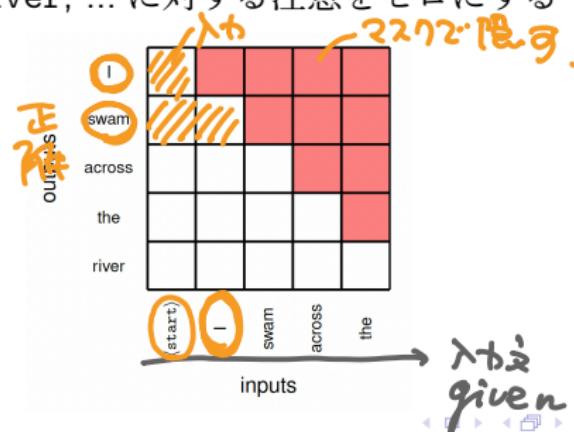
長文を与えて、見せる文の長さを変えつつ、各タスクを解かせる
($W^{(q)}$, $W^{(k)}$, $W^{(v)}$ を更新する)

トランスフォーマーによる次単語の予測

- 入力文の冒頭に<start>を挿入して語順を右に一つずらす
 - x_n に対応する出力が y_{n+1} になる
- 注意機構では、現時刻と過去のみ参照して未来を参照しないようにマスクをかける

例：I swam across the river to get to the other bank.

- across は<start>, I, swam だけから予測
- across, the, river, ... に対する注意をゼロにする

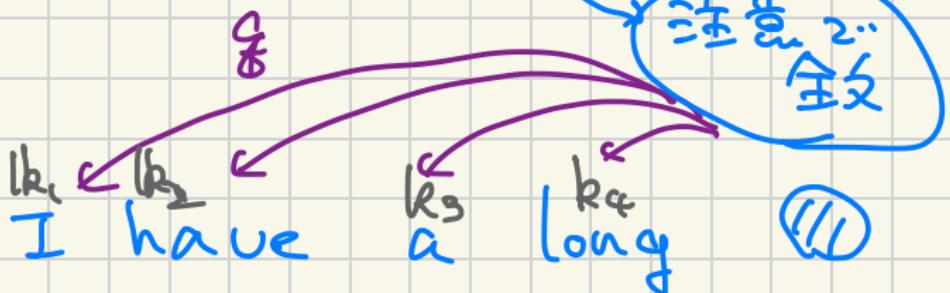


RNN

記憶を失う

状態ベクトル
過去は失う。
今しか失う。

GPT は



単語のサンプリング戦略

下記のふたつの戦略は、いずれも確率の高い系列を出力

- 貪欲法：事後確率最大の系列を選択

- 出力が入力から一意に決まるようになる
- 最大確率を与える単語や系列の選択
- 各時刻で最大確率の単語を選択しても最大確率の系列にはならない

$$p(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N) = \prod_n p(\mathbf{y}_n | \mathbf{y}_1, \dots, \mathbf{y}_{n-1})$$

- 最大確率を与える系列の算出は計算量的に無理

- ビームサーチ

さき B あり

といひながら今 B

- 上位 B の候補を選択
- 候補単語を次の時刻に入力して、それぞれで上位 B 候補を選択
- B^2 通りの候補それぞれで出現確率を算出
 - 事後確率の積 → 長い系列はどんどん確率がゼロに近づく。正規化必要
- 上位 B 系列を選択。以下繰り返し。

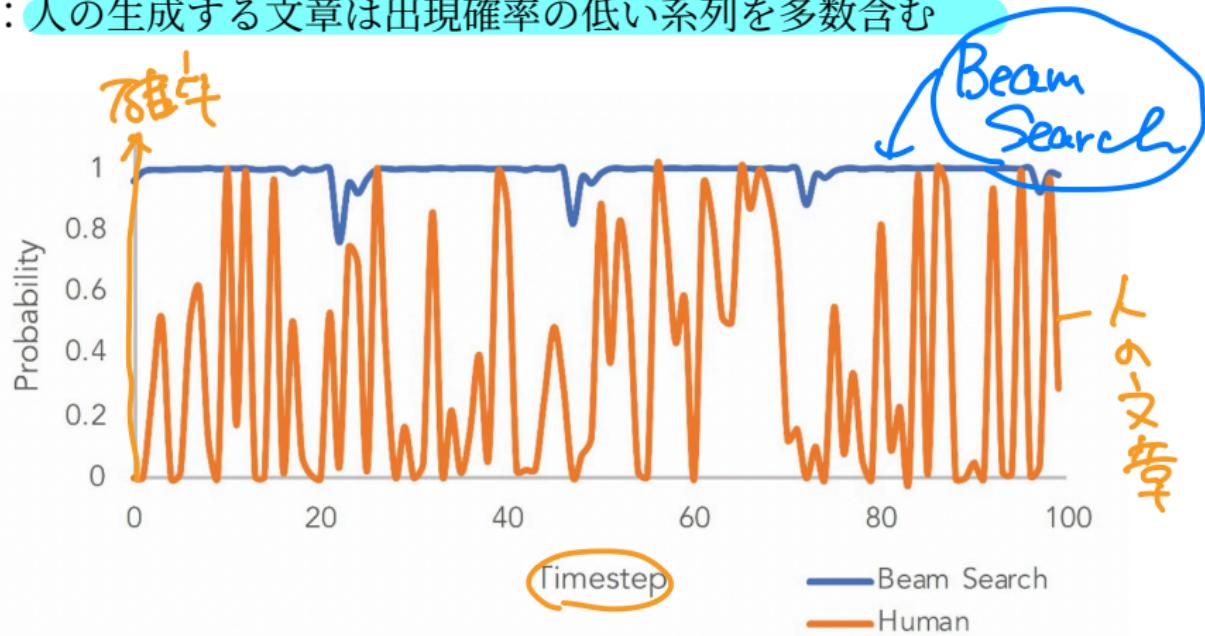
確率の高い系列を出力すると、決まり切った定型文が繰り返し出てくる。

$B = 3$



単語のサンプリング戦略

参考：人の生成する文章は出現確率の低い系列を多数含む



単語のサンプリング戦略

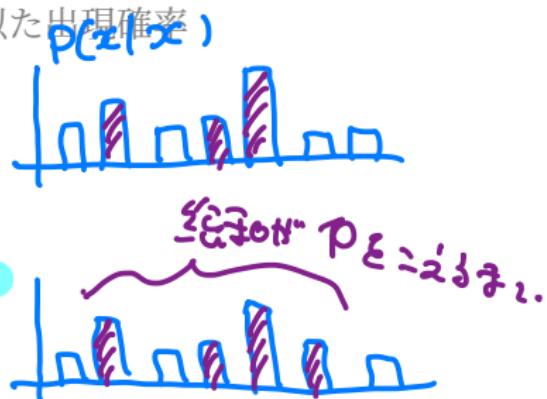
- 各時刻で全語彙から確率値でサンプリング

- 確率値は long tail。多数の単語が似た出現確率
- 無意味な文章が出力されやすい

- Top- K sampling

$K=3$

- 閾値 K を決める
- 上位 K 個の単語を選択
- 確率値を正規化 → サンプリング



- Top- p (nucleus) sampling

- 閾値 p を決める
- 累積確率が p になるまで上位の候補を選択（分布の「核」を選択）
- 確率値を正規化 → サンプリング

自己回帰による文章生成：サンプリングした単語を次の入力とする。過去の出力といまの出力から、次の出力を予測する。繰り返すことで文章を生成できる。

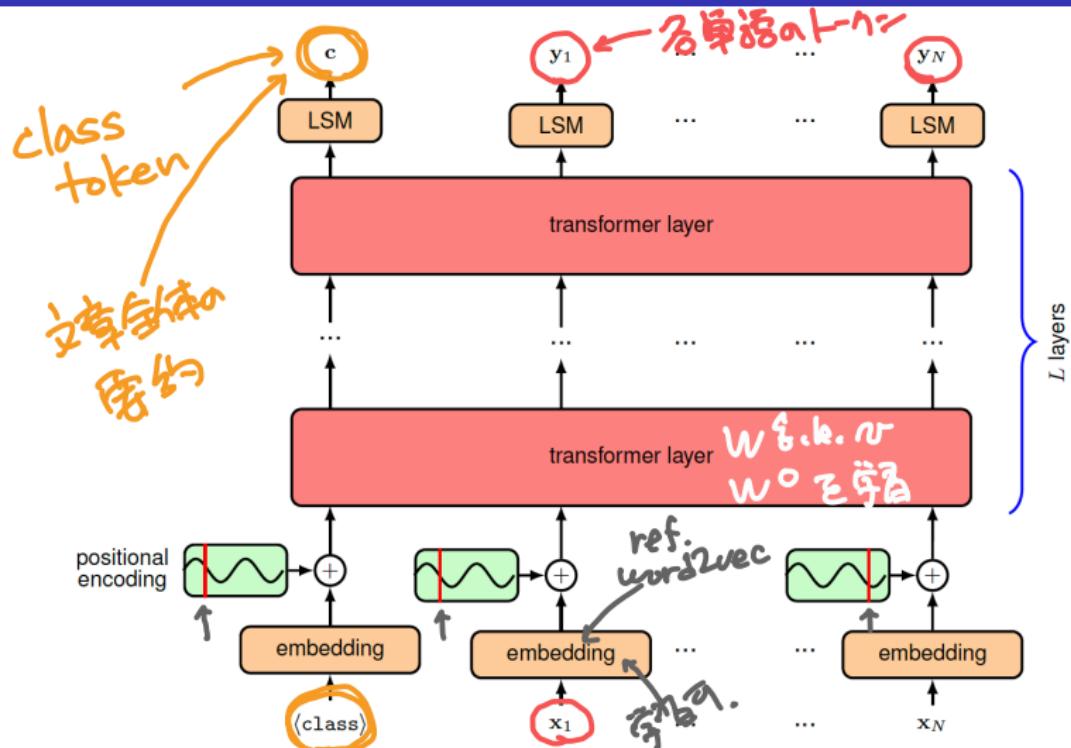
6/26

BERT: ↪ 也も未采土

Bidirectional Encoder Representations from Transformers

トランسفォーマーによる符号器（エンコーダ）

BERT のアーキテクチャ



注：GPT と違って入力に対して出力はずれていない

注：クラストークン, c , が出力される

BERT の自己教師学習

多量の文章より、自動的に教師信号を作り、学習する。

- pretext task : 「mask をかけた単語を予測する」 (MLM: Masked Language Modeling) ↗

例：

- I <mask> across the river to get to the <mask> bank.

bidirectional: mask の前後を参照するの意

- デコーダのときと比べて、学習に利用できる例題が減る
 - (一単語ずつ延ばして例題を作るといったことはできない)
- デコーダのように文章を生成することは出来ない
- mask を予測するときに、他の<mask>トークンが文脈に含まれる
 - 学習結果を応用するときには<mask>は含まれないので、このままだと不整合 (<mask>のトークンに依存する学習がなされる)

BERT の自己教師学習

再掲

- mask を予測するときに、他の<mask>トークンが文脈に含まれる
 - 学習結果を応用するときには<mask>は含まれないので、このままだと不整

対策：文脈中 15% の単語を選択して、次の事柄をおこなう

- 80%: <mask>に置き換えて補完タスクの問題に使う
- 10%: ランダムな単語に置き換える
 - ノイズ耐性改善
- 10%: 単語を変化させず、token からの予測対象に加える
 - <mask>に依らずに各単語の token が単語を表現するように

BERT（符号器）の使い方

Fine-tuning して下流タスクに用いる

- 最初から教師あり学習するよりも良い結果になることを期待

転移学習 (Transfer Learning)

例: mask を予測する

◎ 多量のデータを
(自己教師など)

例: 文章語句トーン
クラス

タスク T' の学習で得られた対象の潜在表現をタスク T へと応用する

- ① 学習データ集合 D' を使って $y' = f'(x')$ を学習
 - 事前学習: pre-training
- ② f' をタスク T の学習に利用する

f' の利用価値

- f' を「特徴抽出器」として利用する
- f' の構造と重みを再利用する (Fine-tuning)

f' の日を初期化する。

我々が
実験にて
タスク

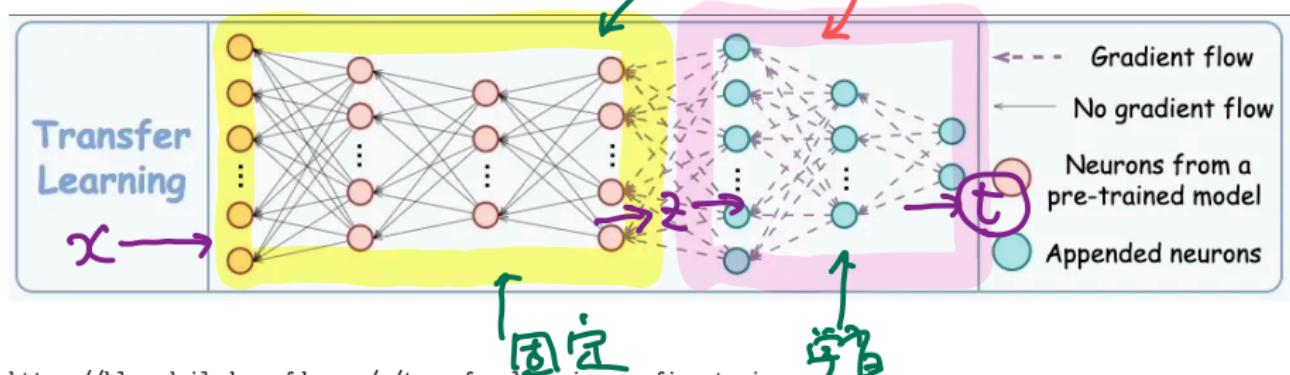
◎ 教科書の
入力は
容易ですが、

転移学習：事前学習による特徴抽出器の構築

f' (の中間層) を新しいタスクに利用する

- $\mathcal{D} = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$
- x_i を f' に入力して $z_i = f'(x_i)$ を得る
- $\tilde{\mathcal{D}} = \{(z_1, t_1), (z_2, t_2), \dots, (z_N, t_N)\}$
- $\tilde{\mathcal{D}}$ を使って学習する

f' : 事前に学習
(大量のデータ)
 f のために追加

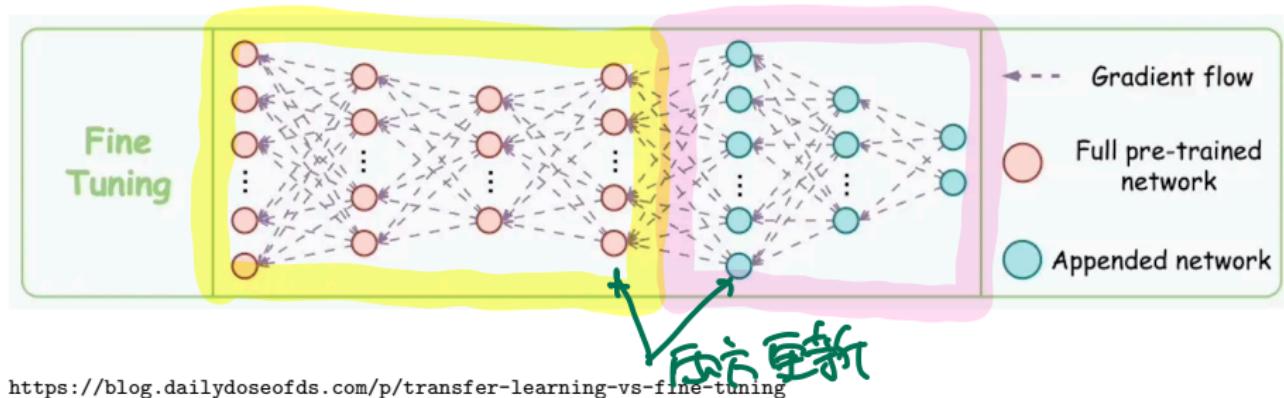


<https://blog.dailydoseofds.com/p/transfer-learning-vs-fine-tuning>

Fine Tuning: 転移学習の一種

タスク T 用のデータ D を用いて f' を修正

- ① f' の一部（もしくは全部）をコピーして f の初期値を得る
 - f' の出力層に近い部分（タスク固有）を更新
- ② D を用いて f を学習する



符号器(BERT)のファインチューニング

- 文章全体の識別：クラストークン, c , を入力とする識別器
 - タスク例：映画評がポジティブかネガティブか
 - 学習中はクラストークンは<mask>の予測に使われる
 - ただしクラストークンが<mask>されることはない
- 各単語の識別：各単語の線形識別器直前のトークンを入力
 - タスク例：各単語の品詞

事前学習の効能：隠した単語の予測可 / 文脈を反映した埋め込みベクトルベクトル

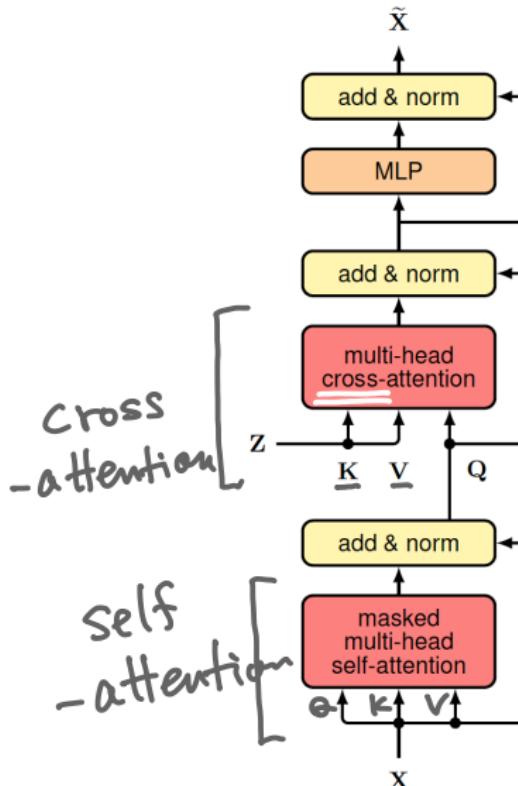
系列変換：符号器と復号器の組み合わせ

自己注意 (self-attention) と交差注意 (cross-attention)

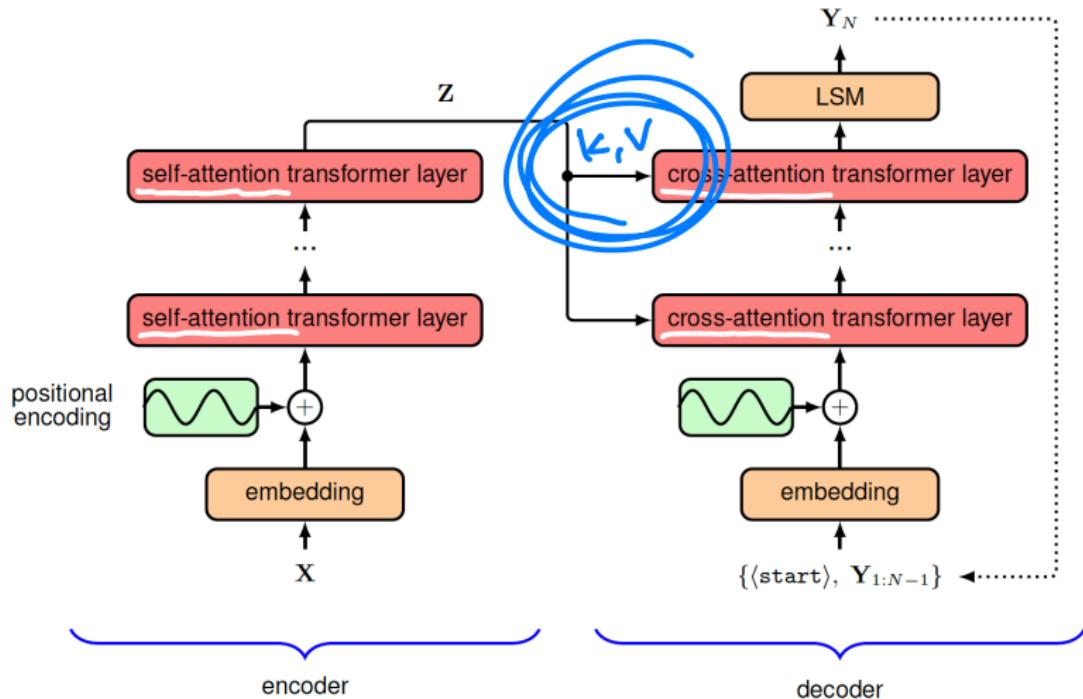
- 自己注意：クエリ・キー・バリューが同じ単語から
- 交差注意：クエリは復号側の単語、キー・バリューは符号側の単語

系列変換：符号器と復号器の組み合わせ

Cross-attention

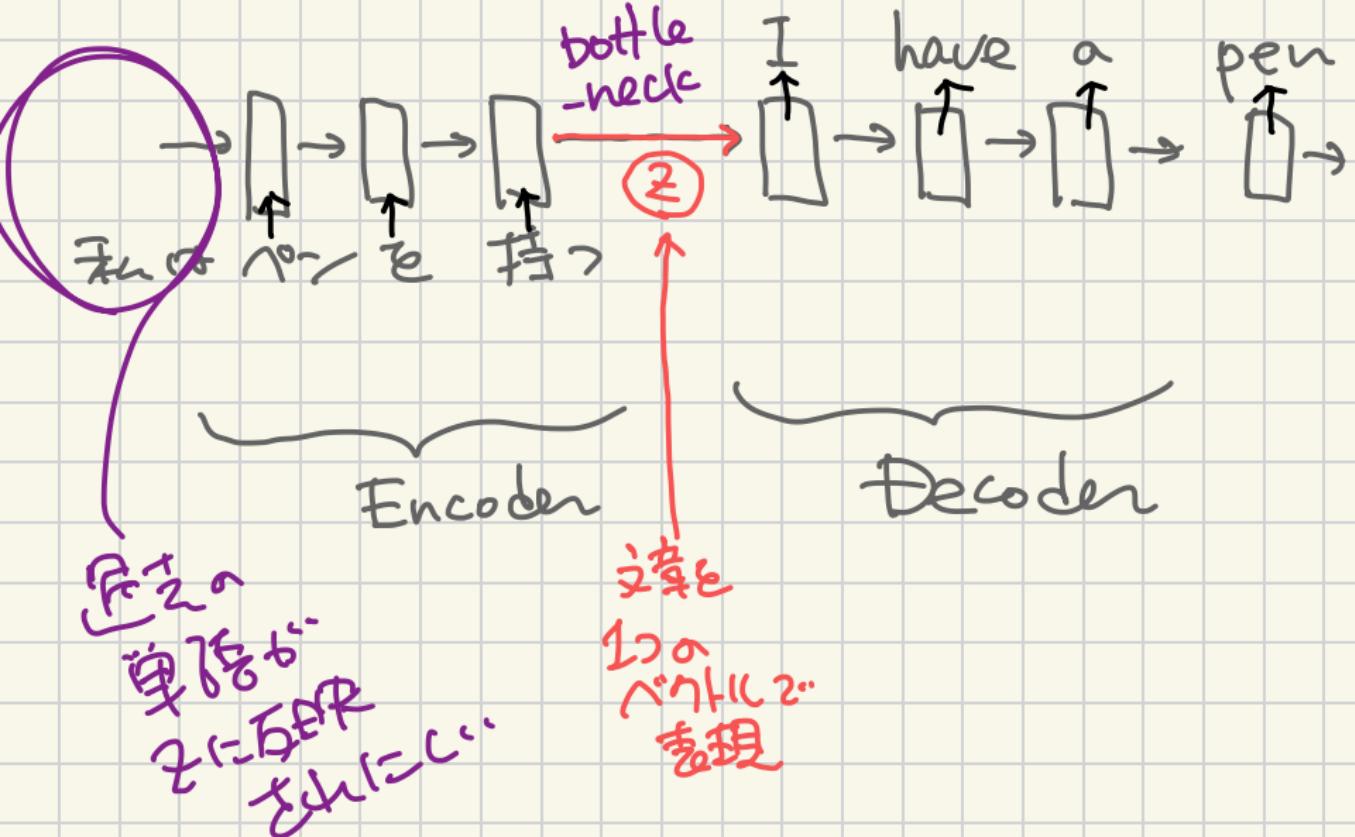


系列変換：符号器と復号器の組み合わせ

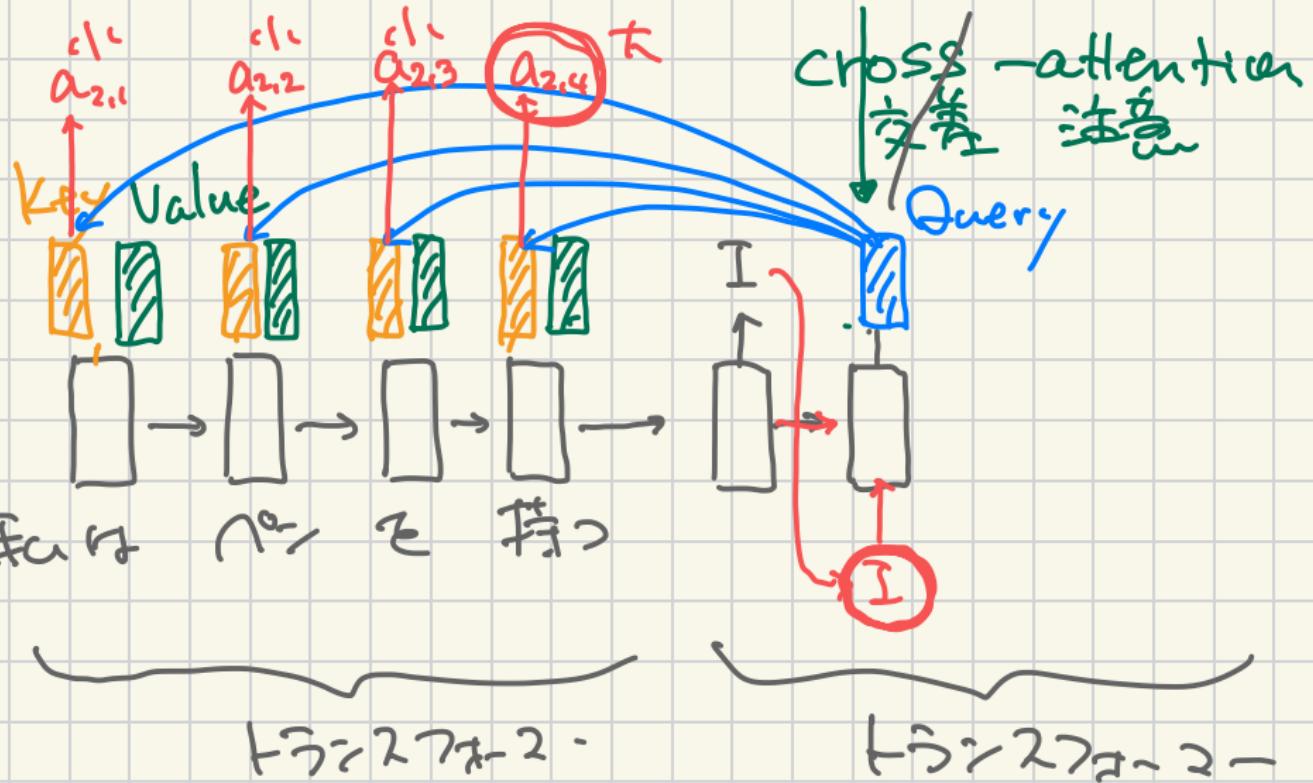


教師あり学習できる

RNN の系列変換





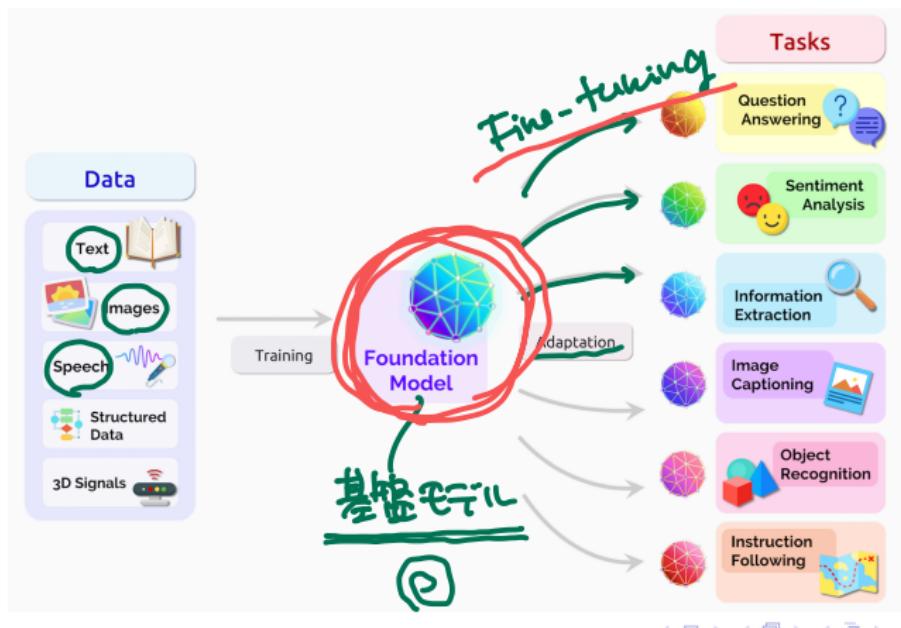


大規模言語モデル ◎ LLM

大規模なトランスフーマーを、多量のデータを用いて、自己教師学習

大規模言語モデル

- 以前：教師あり学習でタスク毎に識別器・回帰器
- 最近：多量のデータで自己教師学習・そのあとタスク毎にファインチューニング
 - いろいろなタスクに使えるモデル：基盤モデル



Prompt

- プロンプト：ユーザが入力するトークンの系列

- フайнチューニング：タスクに特化したモデルを構築
- 同じモデルに、プロンプトを調整することで、様々なタスクを実行させる

4