GPT-nモデル – 事前学習と転移学習

- 巨大な文章のデータセット（コーパス）を用いて**事前学習（pre-trained）**
  → 汎用的な特徴量を習得済みで、**転移学習（transfer learning）**に使用可能
- 転移学習を活用すれば、手元にある新しいタスク（翻訳や質問応答など）に特化したデータセットの規模が小さくても、高精度な予測モデルを実現できる
- 転用する際にはネットワーク（主に下流）のファインチューニングを行う
- 代表的な事前学習モデルはBERTやGPT-nのモデルであり、事前学習と転移学習では全く同じモデルを使うことが特徴的
- 汎用的な学習済み自然言語モデルは、オープンソースとして利用可能なものもある

- 以下ではGPT-nモデルを解説していく
  GPT-3原論文：「Language Models are Few-Shot Learners」
  https://arxiv.org/abs/2005.14165

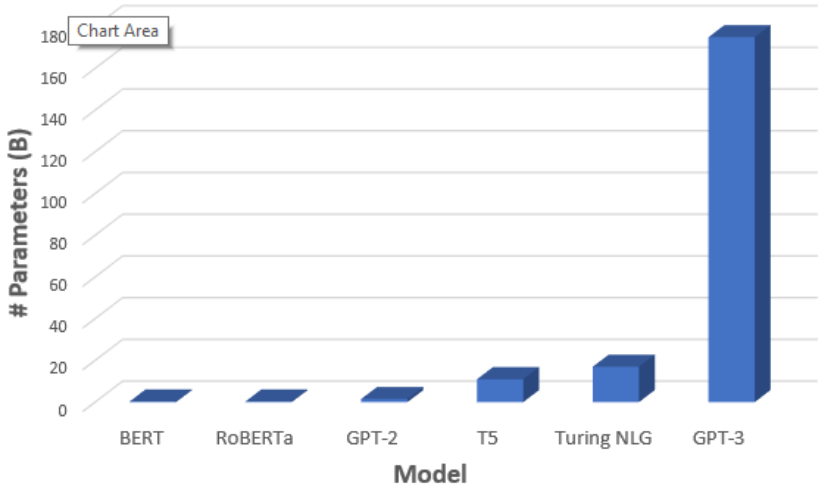
GPT-nモデル – GPTの特徴

G P T（Generative Pre-Training）

- 2019年にOpenAIが開発した有名な事前学習モデル

- その後、GPT-2、GPT-3が相次いで発表
- パラメータ数が桁違いに増加

GPT-nモデル – GPTの特徴



特に、**GPT-3**のパラメータ数は**1750億個**にもなり、約45TBのコーパスで事前学習を行う

（左図）GPT-nモデルを含む事前学習モデルのパラメータ数
引用：OpenAIのブログ, https://towardsdatascience.com/gpt-3-the-new-mighty-language-model-from-openai-a74ff35346fc

GPTの仕組み

- GPTの構造はトランスフォーマーを基本とし、**「ある単語の次に来る単語」を予測し、自動的に文章を完成**できるように、教師なし学習を行う
- 出力値は「その単語が次に来る**確率**」
  例えば、単語系列”After”、”running”、”I”、”am”、の次に来る単語の確率が
  ”tired”:40%、”hot”:30%、”thirsty”:20%、 ”angry”:5%、”empty”:5%
  になったと仮定すると、”tired”や”hot”が可能性の高い、”angry”や”empty”は低い
- 学習前の1750億のパラメーターはランダムな値に設定され、学習を実行後に更新される
- 学習の途中で誤って予測をした場合、誤りと正解の間の誤差を計算しその誤差を学習する

GPT-3について報告されている問題点

■社会の安全に関する課題

- 「人間らしい」文章を生成する能力を持つため、フェイクニュースなどの悪用のリスクがある
- 現在は完全オープンソースとして提供されておらず、OpenAIへAPIの利用申請が必要

（参考）https://openai.com/blog/openai-api/

■学習や運用のコストの制約

- 膨大な数のパラメータを使用したGPT-3の事前学習を現実的な時間で実現するためには、非常に高性能なGPUを必要とする

■機能の限界（人間社会の慣習や常識を認識できないことに起因）

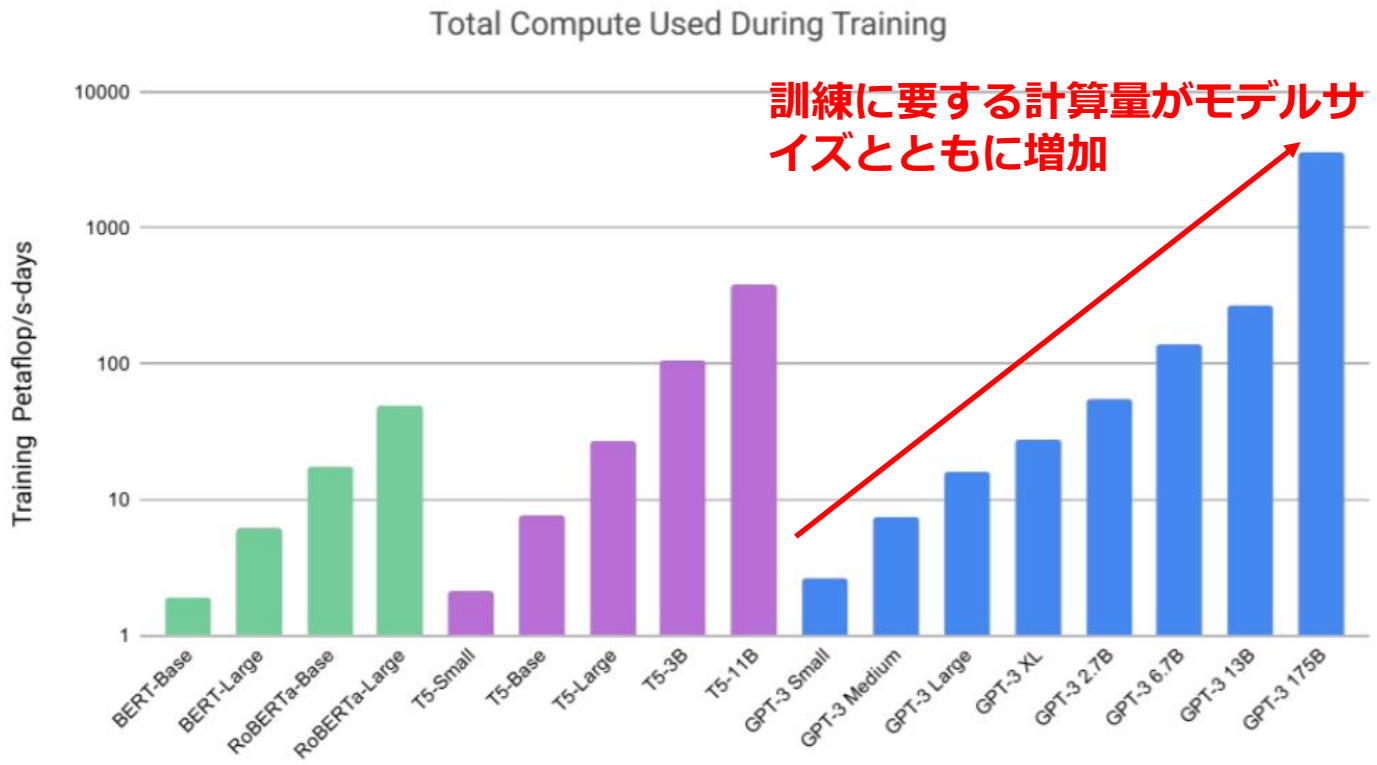
- 生成した文章の文法が正しくても、**違和感や矛盾**を感じることがある
- **「物理現象に関する推論」が苦手**（例：「アイスを冷凍庫に入れると硬くなりますか」には答えにくい）

GPT-3のモデルサイズ・アーキテクチャー・計算量

一般的に”GPT-3”と呼ばれているGPT-3 175B のパラメータ数は1750億

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

petaflop/s-days = 1秒に1ペタ（10^5）回の演算操作を1日分実施した計算量



毎秒1ペタ回の演算を行う場合、 GPT-3 175Bの学習には数年以上を要する

GPTの事前学習

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

事前学習では、この式を最大化するように
学習する

GPTの事前学習

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

$\mathcal{U} = \{u_1, \ldots, u_n\}$ が言語データセット(ラベルなし)で、{}の中はそのデータセットの一つ一つの単語を示している

GPTの事前学習

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

kがコンテキストウィンドウといって
単語u_iを予測するために
その前kの単語を何個使うかを示している

Θがニューラルネットワークのパラメーター

GPTの事前学習

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

文に出てくる単語u_iを、その前の単語
を使って予測し、その単語u_iと予測する確率を
最大化する

GPTの事前学習

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

学習の流れを表している

GPTの事前学習

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

$U = (u_{-k}, \ldots, u_{-1})$ が、先程出てきた、対象の単語を予測するために使う複数の単語

GPTの事前学習

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

W_e が単語の埋め込み表現、W_p が
位置エンコーディングベクトル

GPTの事前学習

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

h0は単語の埋め込み表現に位置エンコーディングを足したもの

GPTの事前学習

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

transformer_blockはtransformerのdecoderを使う　後で図を使い解説する

GPTの事前学習

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

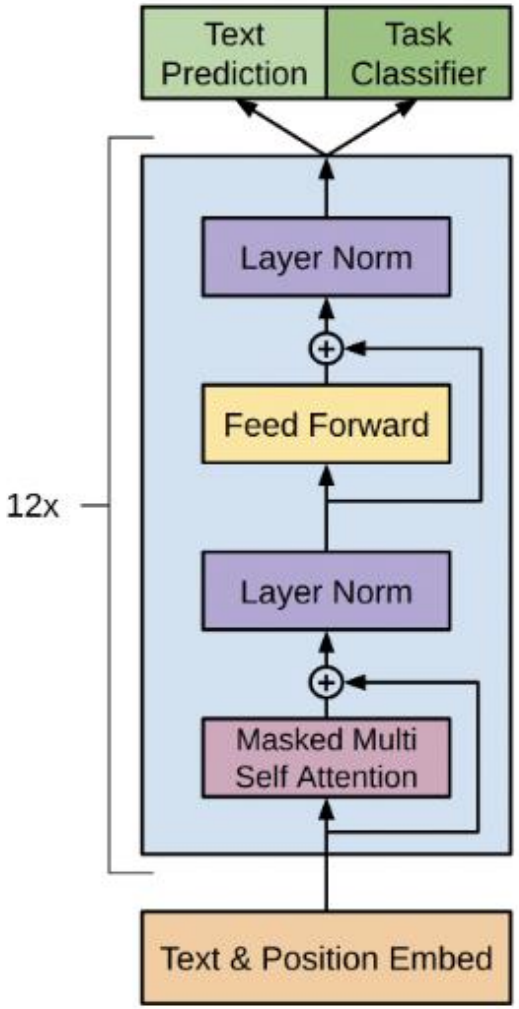
nがtransformerのレイヤーの数を表していて、
h0を入力として入れ、その出力のh1を次のレイヤーの
入力として入れ次のレイヤーにh2を…
という操作をn回行う

GPTの事前学習

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

transformerの出力と埋め込み表現の転置行列を
かけたものをsoftmax関数に入れ、
最終的な出力とする

GPTの事前学習



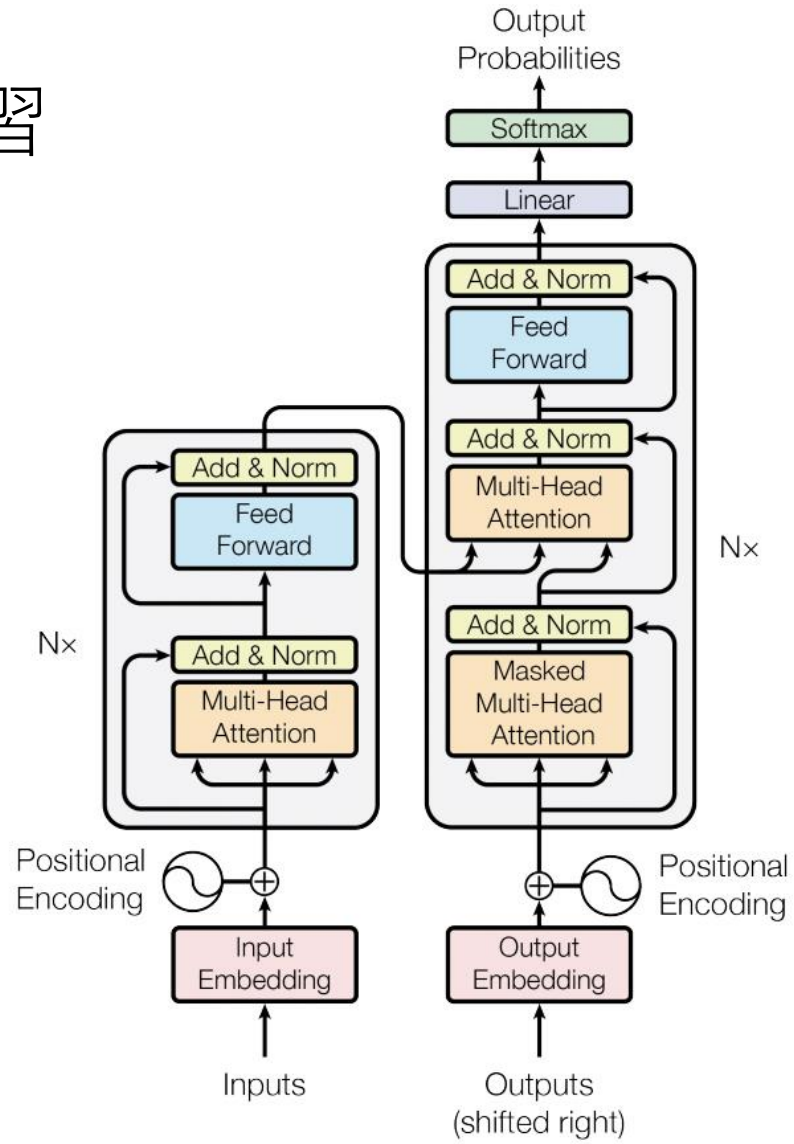
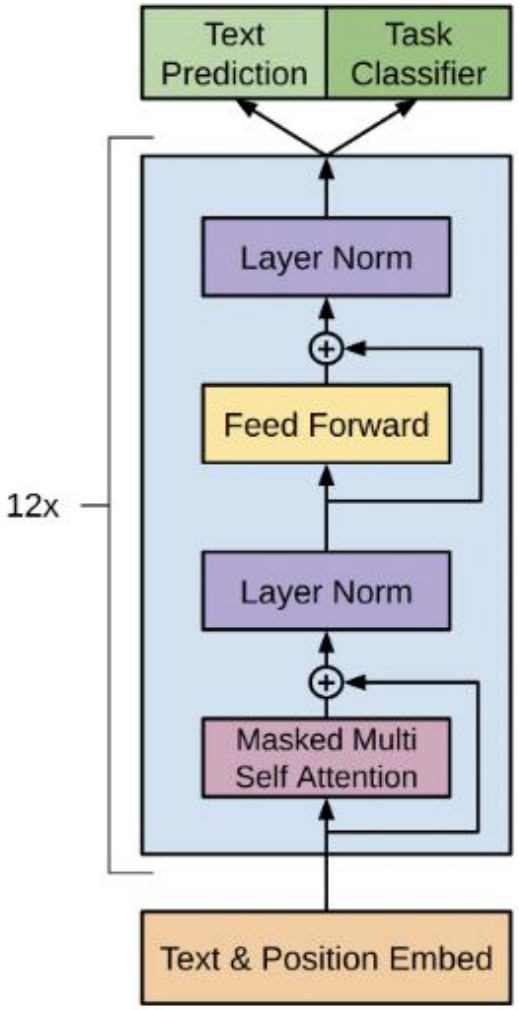
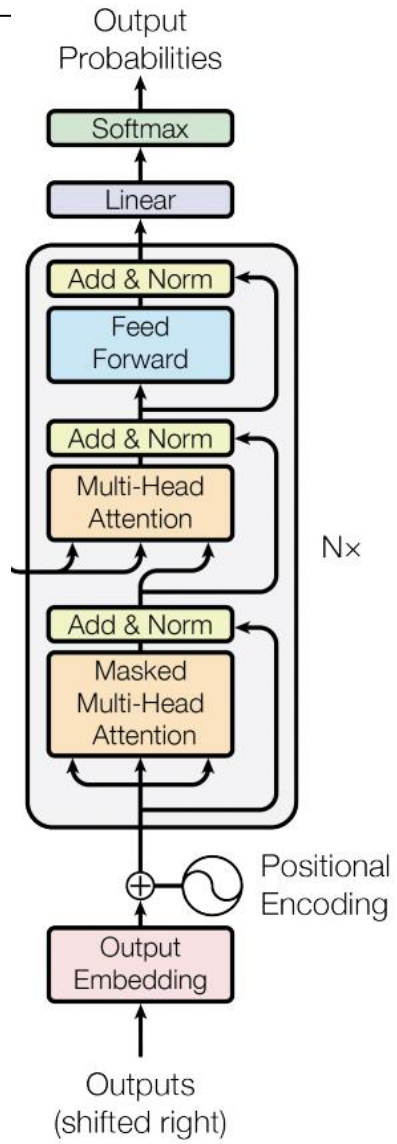


Figure 1: The Transformer - model architecture.

左がGPTのtransformer、右がベースのtransformer

GPTの事前学習

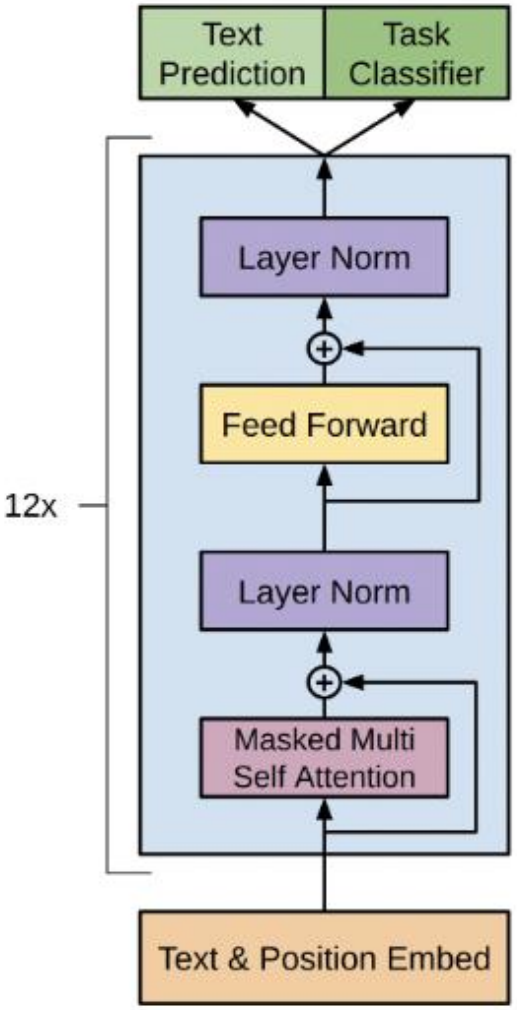


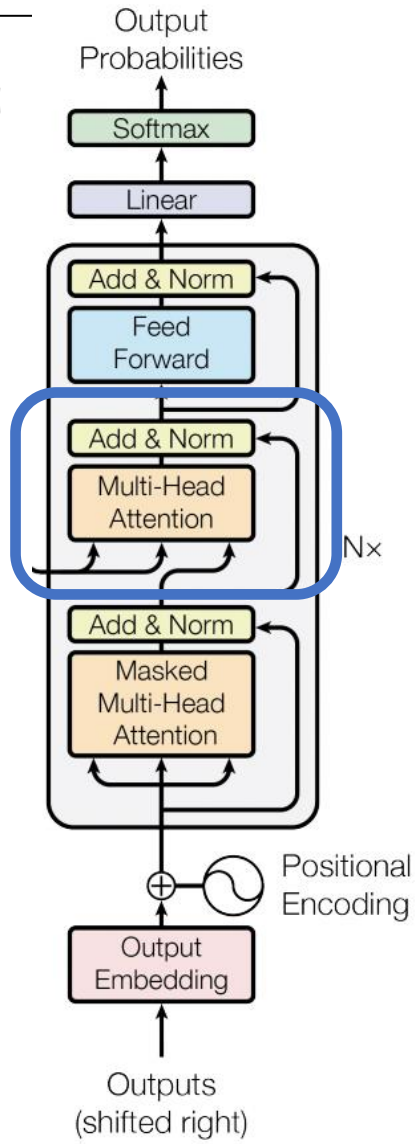


er - model architecture.

GPTはデコーダーのみを
使うのでデコーダーを比較する

GPTの事前学習



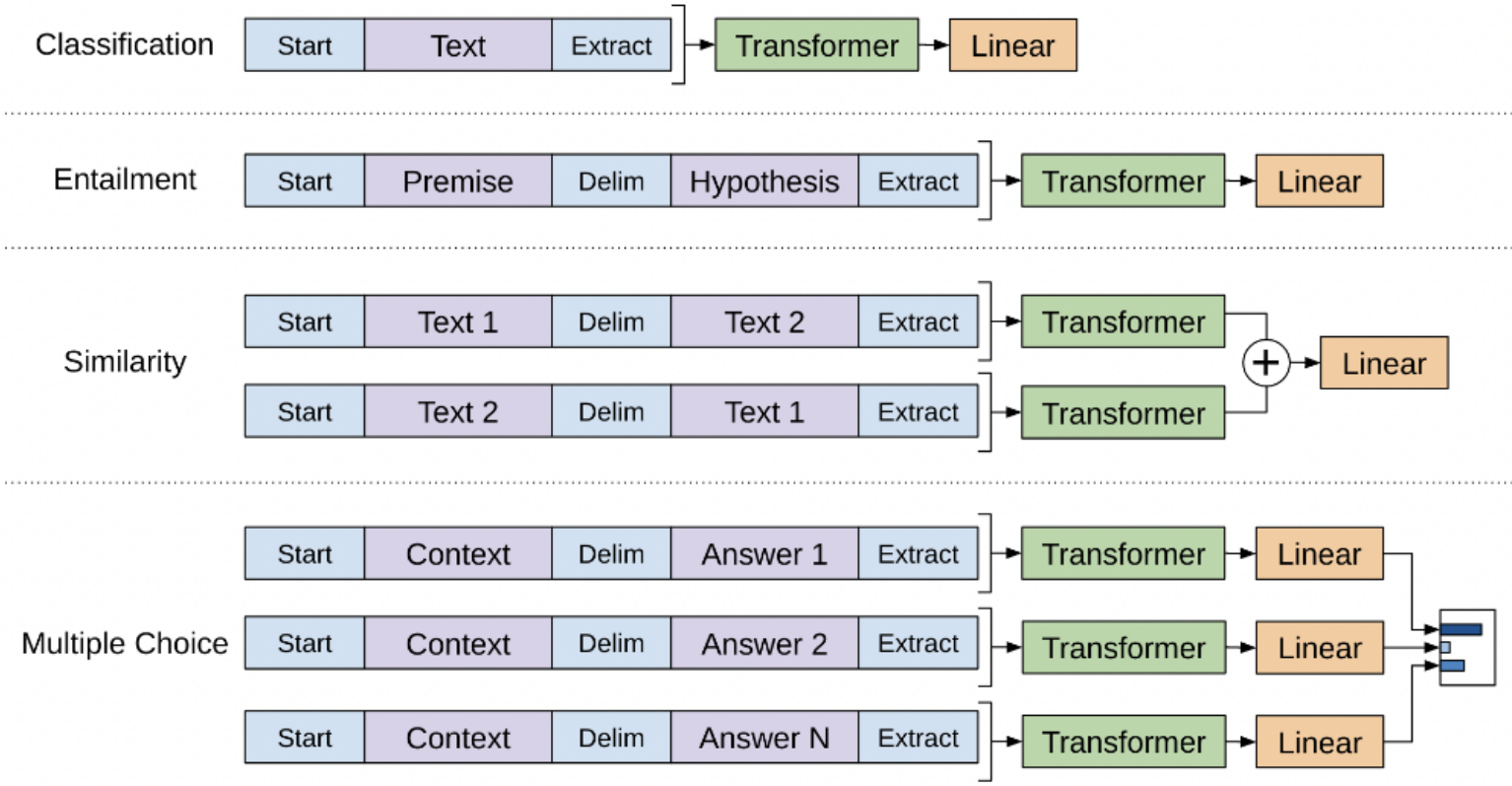


er - model architecture.

青で囲まれている部分が減った以外ほぼ同じ構造

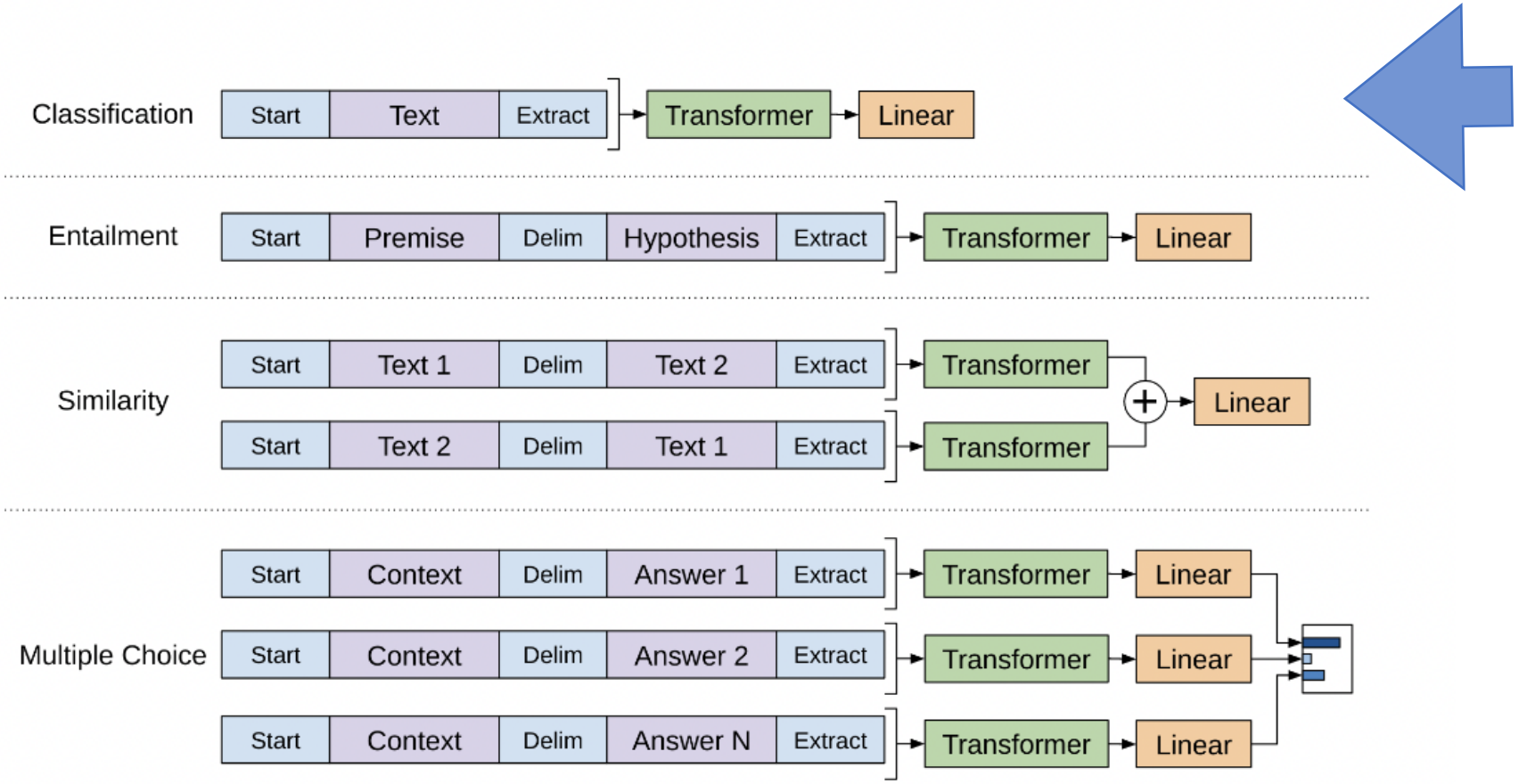
GPT-1のfine-tuning

GPTの



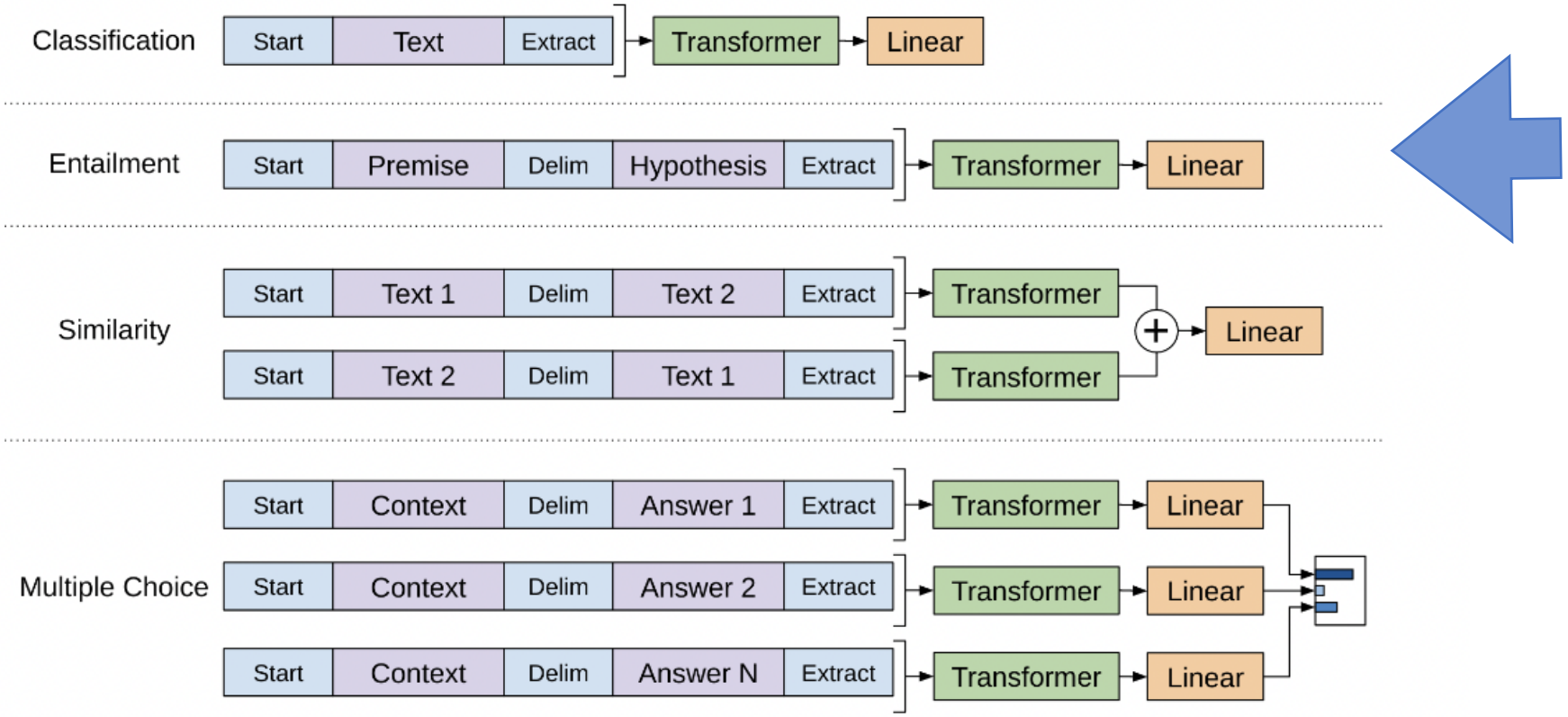
転移学習では、始まりを表す記号、文と文を区切る記号、終わりを表す記号を使う

GPTの



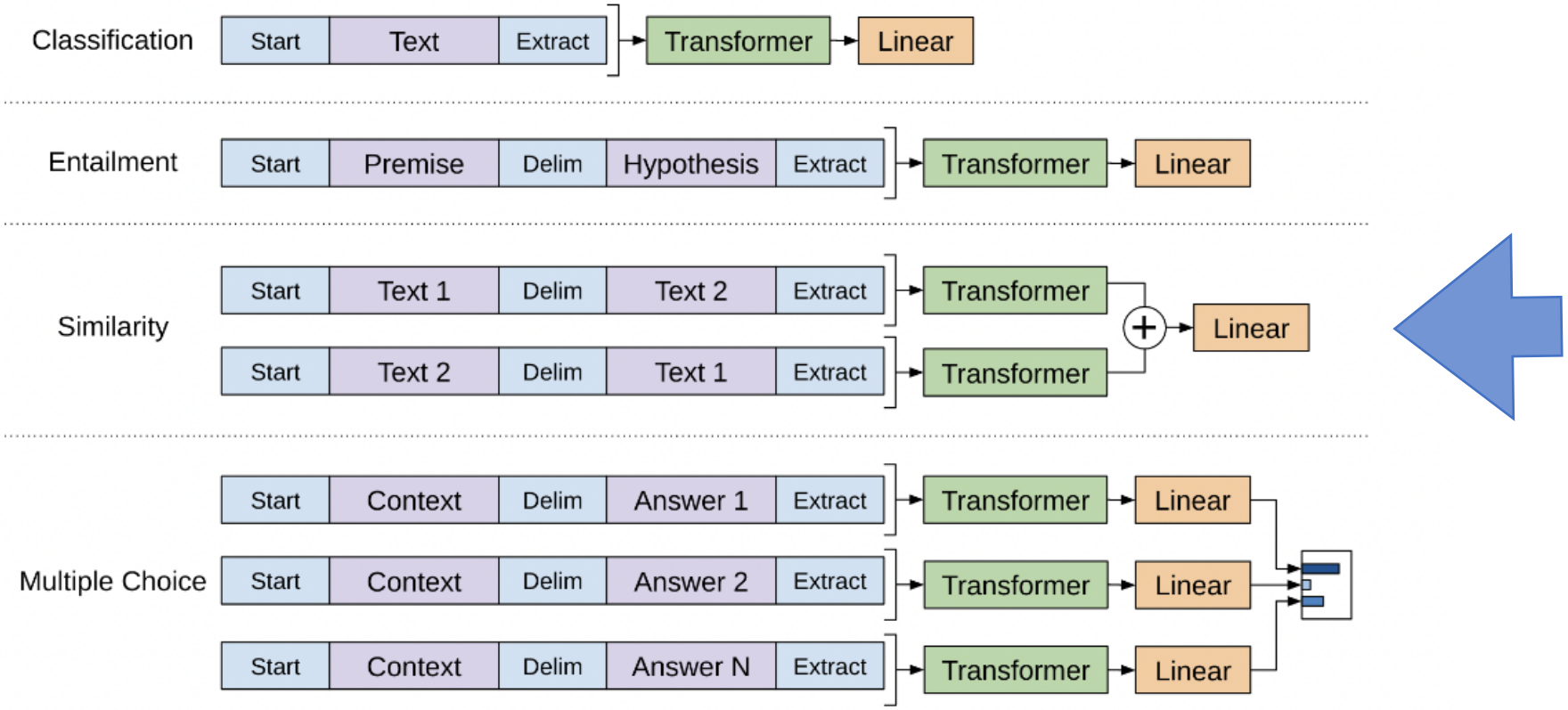
テキスト分類では区切りの記号は使わず、
テキストをTransformerに入れ、線形結合、softmaxに通し答えを得る

GPTの



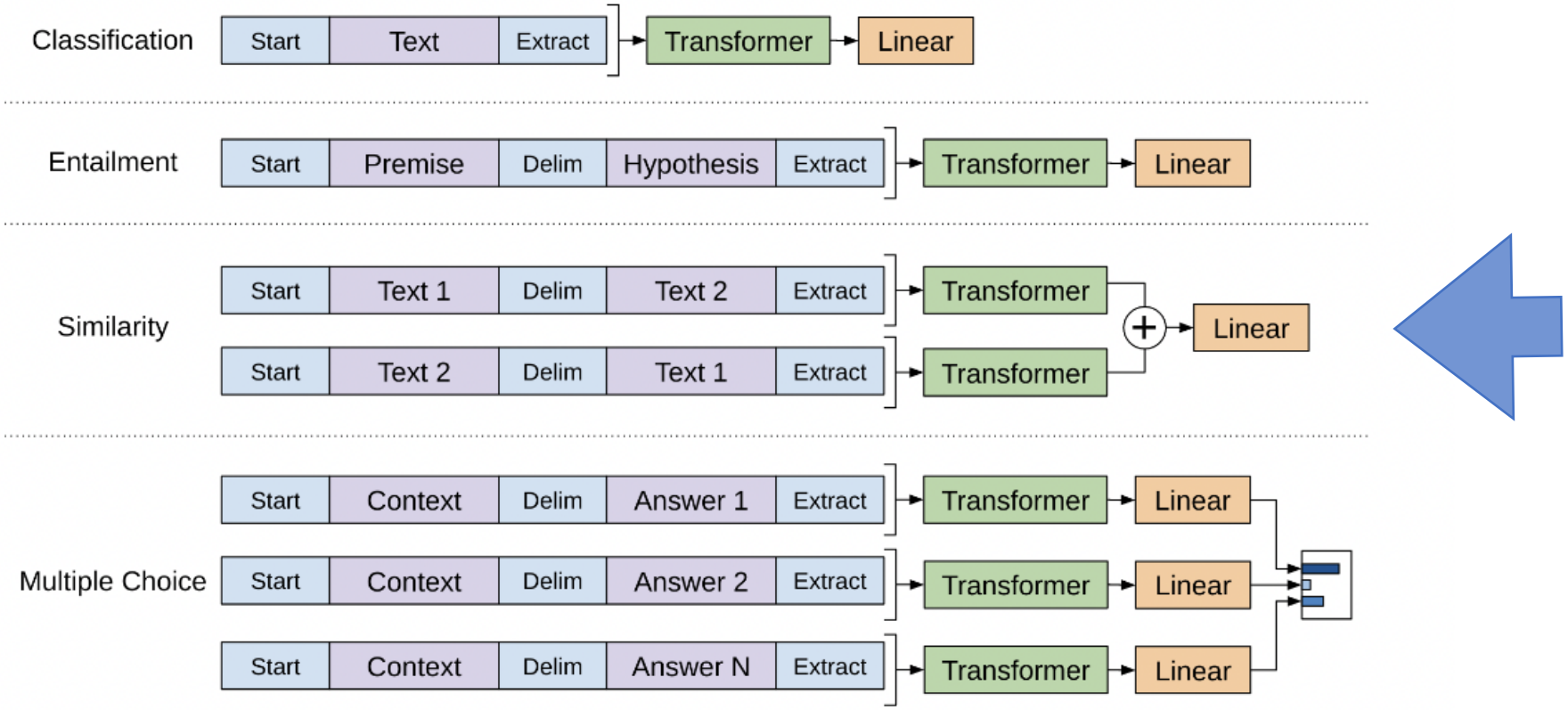
文同士の関係を予測する場合は区切り記号を使って予測する

GPTの



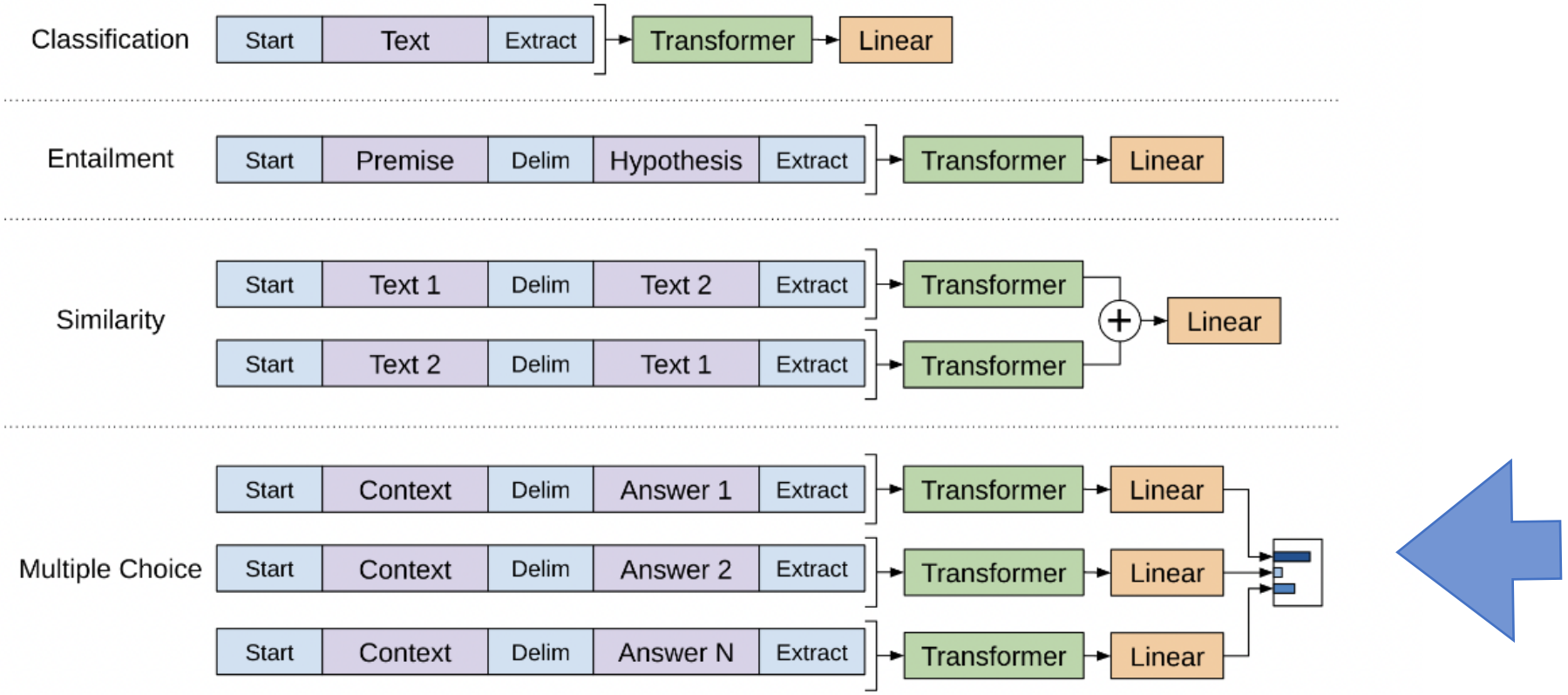
文の類似度を予測する場合は、二つの文を区切り文字で区切り、順番を入れ替えた入力をもう一つ用意する

GPTの



それぞれTransformerに入力し、その出力を足して
線形結合-> softmaxで類似度を得る

GPTの

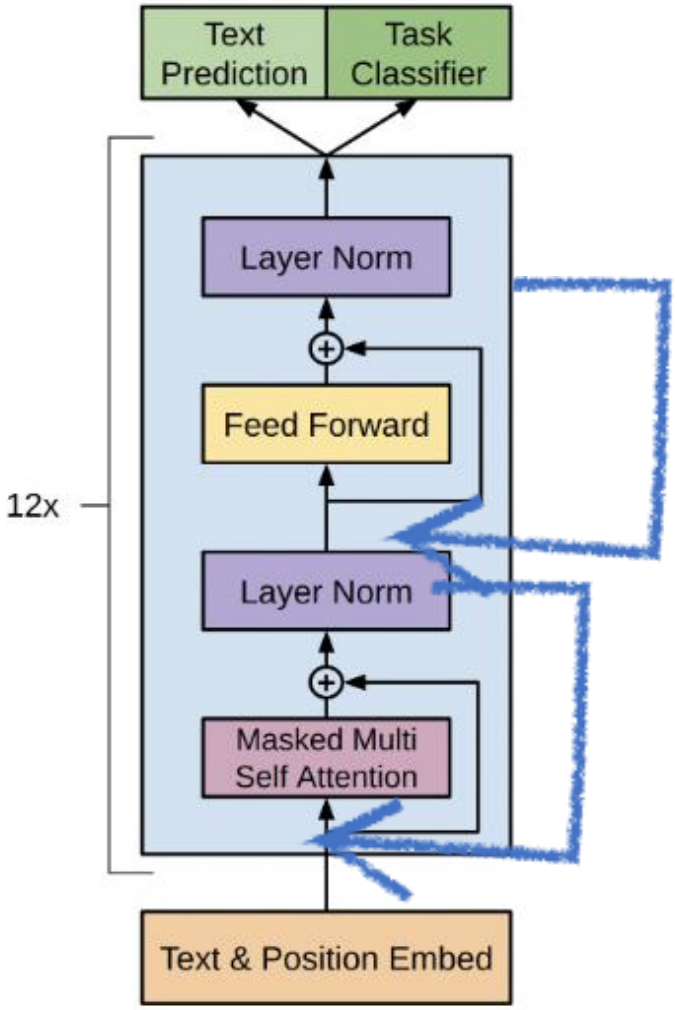


複数の文から一つを選ぶ場合、線形結合層もそれぞれ
分かれていて、それぞれの出力を比較して最終的な答えとする

GPTの事前学習

GPT-2、GPT-3でも細かい変更はあるものの
基本的なモデルの構造は変わらない

GPTの事前学習



GPT-2での変更点

Layer Normの位置を前にずらしたこと

最後のself-attentionブロックの後にも
Layer Norm層を入れたこと

GPTの事前学習

GPT-2のその他の変更点：
バッチサイズやデータセットを大きくした

GPT-3の変更点：
埋め込みサイズ、層、Multi-Head Attentionの数、
コンテキストウィンドウの数を増やした

GPTの事前学習

GPT-2のその他の変更点：
バッチサイズやデータセットを大きくした

GPT-3の変更点：
埋め込みサイズ、層、Multi-Head Attentionの数、
コンテキストウィンドウの数を増やした

1、2、3と発表されるたびに規模が大きくなっている

GPT3

GPT-3では、fine-tuningをしない
改めて勾配を更新し直すことをしない

GPT3

GPT-3の推論は、zero-shot、one-shot、few-shotに分類できる

GPT3

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

zero-shotでは、なんのタスクか（翻訳なのか、文生成なのかなど）を指定した後、すぐ推論させる

GPT3

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

one-shotでは、なんのタスクかを指定したあと
一つだけラベル付きの例を教え、その後推論させる

GPT3

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:          ← task description
2  sea otter => loutre de mer            ← examples
3  peppermint => menthe poivrée          ←
4  plush girafe => girafe peluche        ←
5  cheese =>                             ← prompt
```

few-shotでは、なんのタスクかを指定した後、
2つ以上の例を教え、そのあと推論する

【GPTのすごいところ】

■幅広い言語タスクを高精度で実現できる

- あったかも人間が書いたような文章を生成できて、翻訳、質疑応答、文章の穴埋め、ソースコード生成など、様々なアプリケーションに使用できる

（参考）https://gptcrush.com/resources/

BERTとGPTの比較

BERTは、Tranformerのエンコーダー部分を使っている

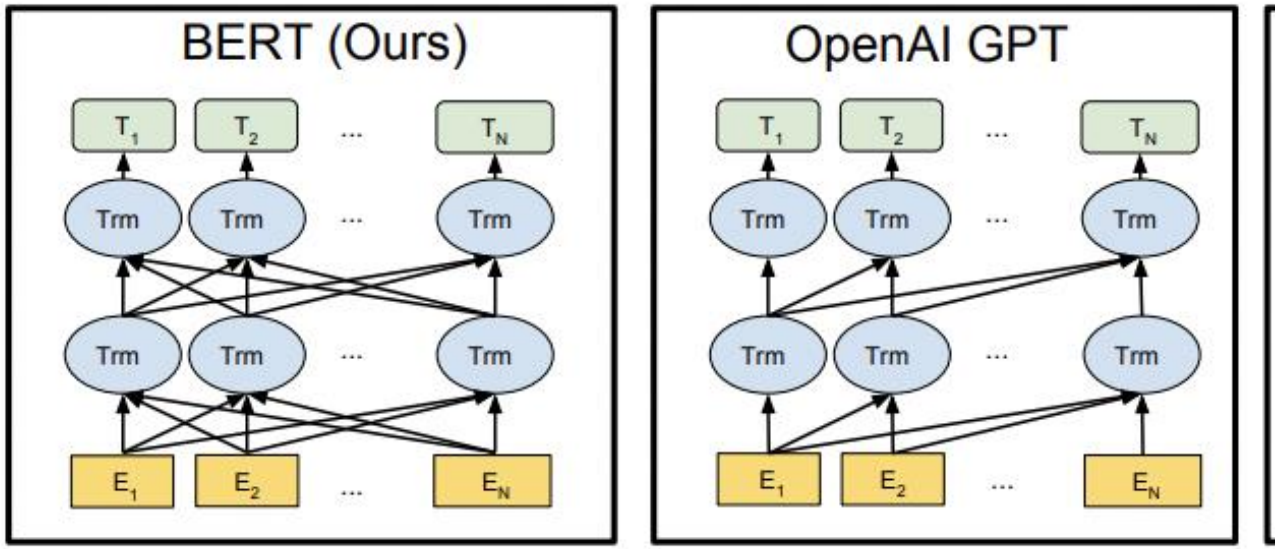
GPTはTransformerのデコーダー部分を使っている

BERTとGPTの比較

BERTは、新しいタスクに対してファインチューニングが必要

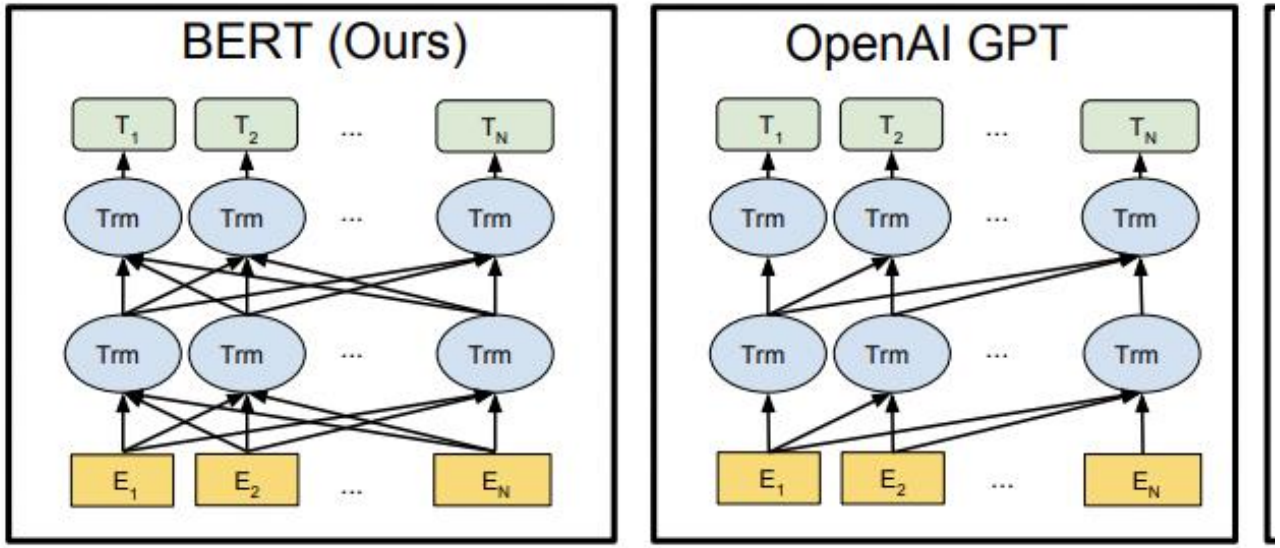
GPT-3はファインチューニングをしない

BERTとGP



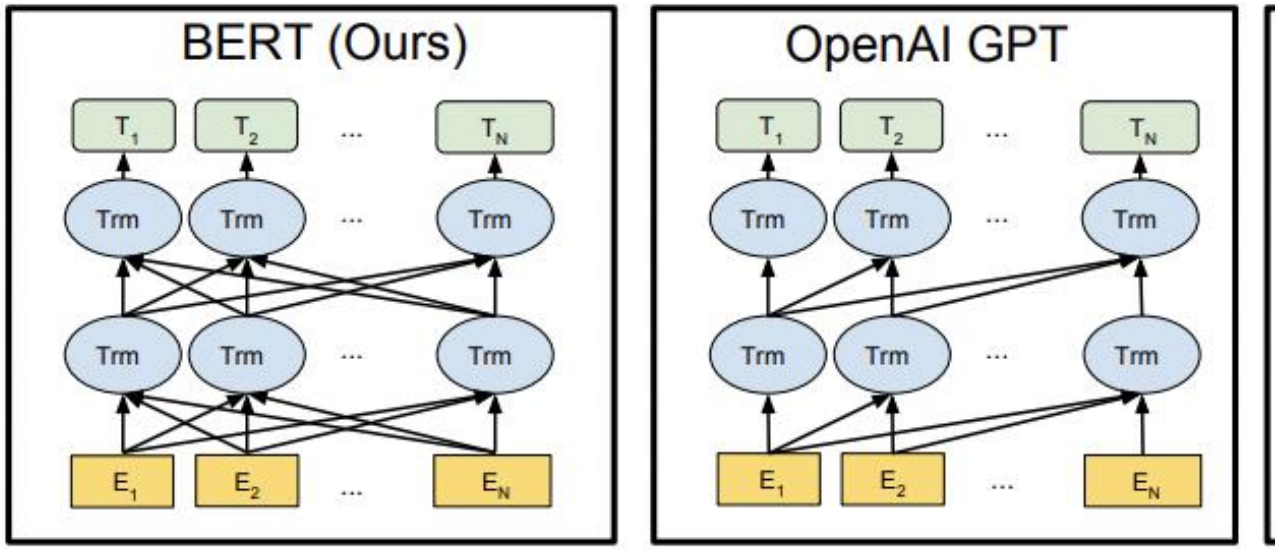
BERTは双方向Transformerで、GPTは単一方向のTransformerである

BERTとGP



BERTは文の中のどの位置の単語もマスクされる可能性があり、
マスクした前の単語も後ろの単語にも注目する必要がある

BERTとGP



GPTは常に次の単語を予測するため、双方向ではない