学生・若手研究者のための BERTワークショップ

RIKEN AIP 中山功太

本ワークショップの目的

目的

BERTについての知識を得る。

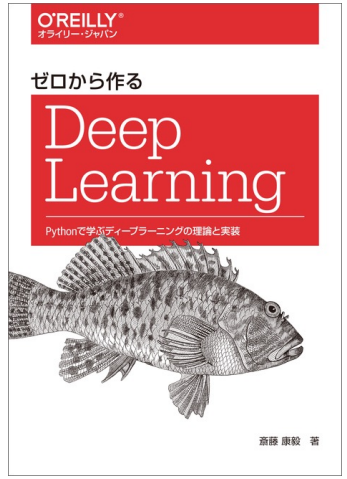
BERTを用いて文書分類タスクを試す(Pytorch実装)。

より発展的な理解のために

本ワークショップでは深層学習の知識がなくともデモの実行は可能ですが、より深く理解したい場合は、「ゼロから作るDeep Learning」を読んだり、以下の深層学習や誤差逆伝播を数学的に解説した動画の視聴をお勧めします。

https://youtu.be/xzzTYL90M8s

https://youtu.be/0itH0iDO8BE



深層学習についての説明

ザクっとわかる深層学習

深層学習は一般的に予測と正答ラベルの間の誤差を最小化することでモデルを学習する。

入力　吾輩は猫である。 名前はまだ無い。

深層学習モデル e.g. BERT
（計算グラフ）

予測

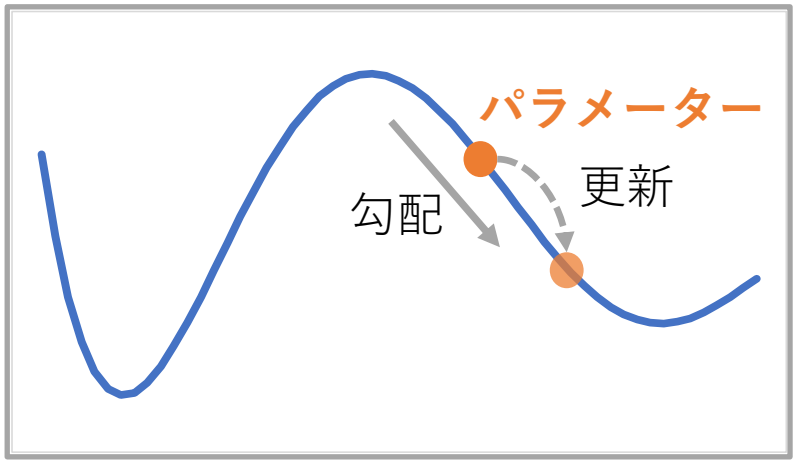
夏目漱石	太宰治
0.3	0.7

誤差

正答ラベル

夏目漱石	太宰治
1	0

誤差逆伝播+パラメーター更新
深層学習モデルの誤差を微分し
誤差が小さくなる方向にパラメーターを更新する。
⇨オプティマイザーが勝手にやってくれる。



バッチ入力

入力

吾輩は猫である。 名前はまだ無い。
私は、その男の写真を三葉、見たことがある。
親譲の無鉄砲で小供の時から損ばかりしている。
…

バッチサイズ

深層学習モデル e.g. BERT（計算グラフ）

バッチ入力とは
複数の入力をもとに誤差を最小化することで学習を安定化させるテクニック。

予測

夏目漱石	太宰治
0.3	0.7
0.8	0.2
0.4	0.6
…	…

誤差
(基本的には
誤差の平均)

夏目漱石	太宰治
1	0
0	1
1	0
…	…

正答ラベル

BERTについての説明

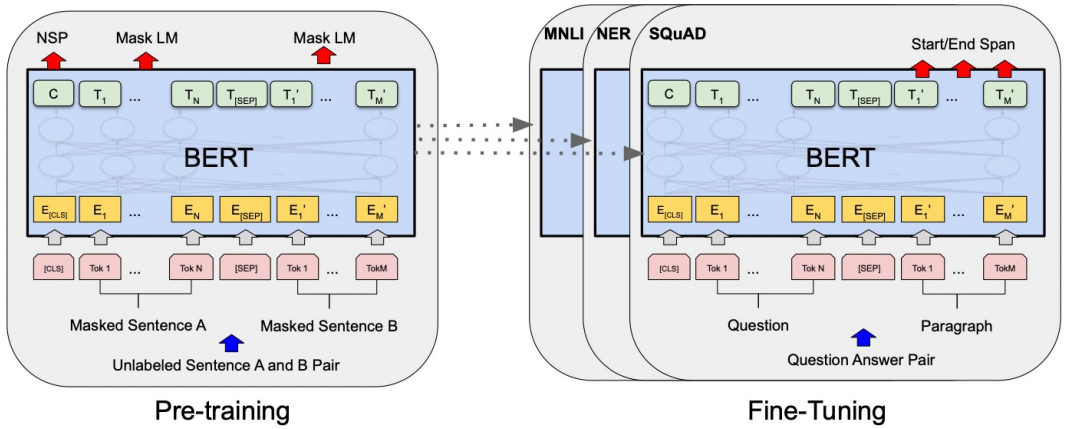
BERTとは

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)
で提案された深層学習モデル。2022/7/22現在 44471引用!!!

簡単に、**さまざまな自然言語処理タスクに適用可能**であり、実際に多くの研究で用いられている。

以下の順序でBERTについて説明していく。

1. 入力文章の前処理
2. BERTの構造
   A) 埋め込み
   B) 自己注意機構(Self-Attention Mechanism)
   C) 残差接続(Residual Connection)
   D) 層正規化(Layer Normalization)
3. 事前学習
   A) マスク言語モデリング(MLM : Masked Language Modeling)
4. 分類タスクへの適用法



1. 入力文章の前処理

BERTでは自然文を分割した後、ID配列に変換し入力とする。

入力文章　　本日は晴天なり。

↓ **単語分割**　※正確には単語より細かいサブワードに分割する　※##は1つ前のサブワードと結合して1つの単語を意味することを示す。

トークン　　[本，##日，は，晴，##天，なり，。]

↓ **特殊トークン結合**

特殊トークン
[CLS]：文章の先頭を示す
[SEP]：文章の終わり(もしくは区切り)
[PAD]：入力長の調節用
[UNK]：語彙リストに含まれないトークン
[MASK]：のちに説明

トークン　　[[CLS]，本，##日，は，晴，##天，なり，。，[SEP]，[PAD]，…，[PAD]]

↓ **辞書を用いてID化**

辞書は事前に大規模データから単語頻度を用いて作成する
辞書のサイズは32000程度が用いられる場合が多い。
辞書に含まれなかったトークンは事前に[UNK]トークンに置き換える。

トークンID　[　2，　108, 28486，　9, 4798, 28849，297，　8，　　3，　　0，　…，　0　]

1. 入力文章の前処理

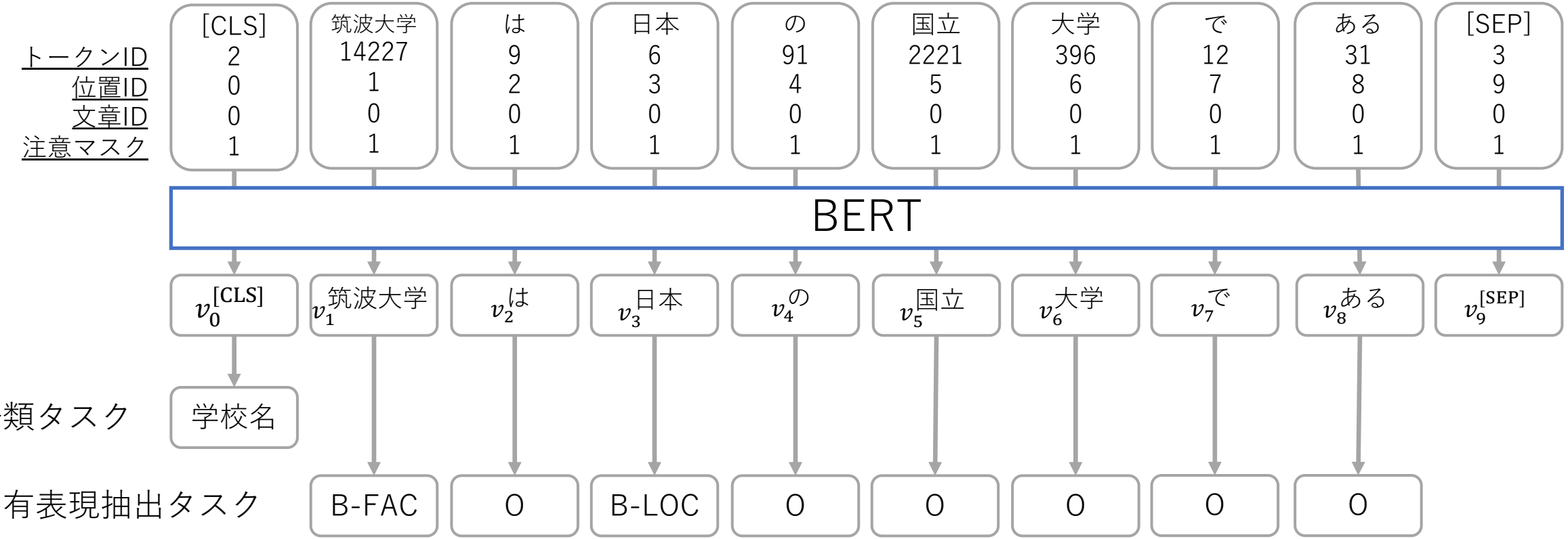
BERTは各トークンIDに対応する位置ID、文章ID、注意マスクも入力に取る。

トークンID	2	108	28486	9	4798	28849	297	8	3	0	…	0
位置ID	0	1	2	3	4	5	6	7	8	9	…	N-1
	各入力トークンの位置をモデルが把握するために使用する。0~入力長N-1までの連番をとる。											
文章ID	0	0	0	0	0	0	0	0	0	0	…	0
	複数入力文章がある場合にモデルが文章のスパンを把握するために使用する。分類等単一文章が入力の場合は0のみをとる。											
注意マスク	1	1	1	1	1	1	1	1	1	0	…	0
	意味を持たない[PAD]トークンの位置をモデルが把握するために使用する。[PAD]の場合0、それ以外の場合1をとる。											

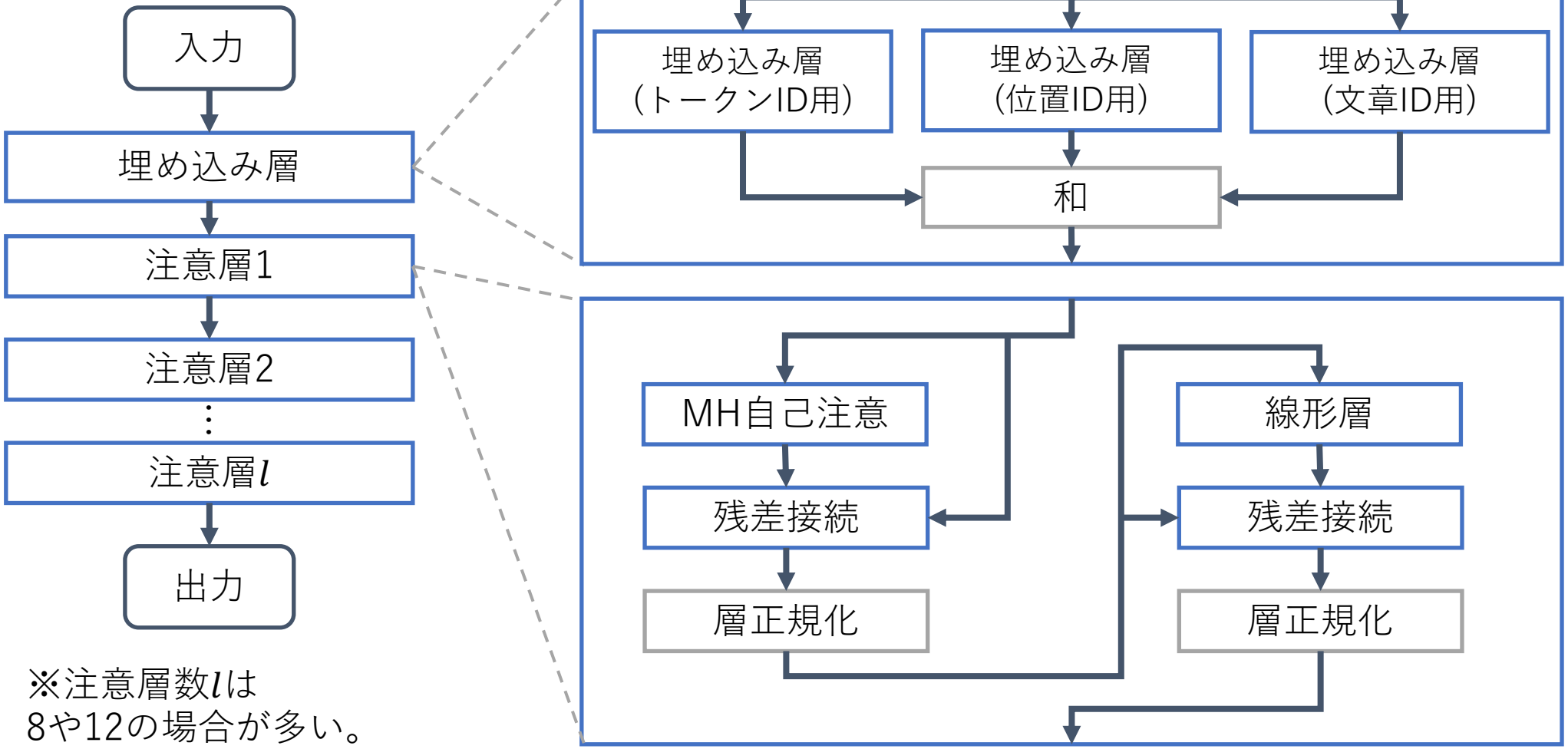
入力長N

2. BERTの構造

BERTは各トークンごとにベクトルを出力する。
多くの場合この出力ベクトルをもとにさまざまなタスクを解くことになる。

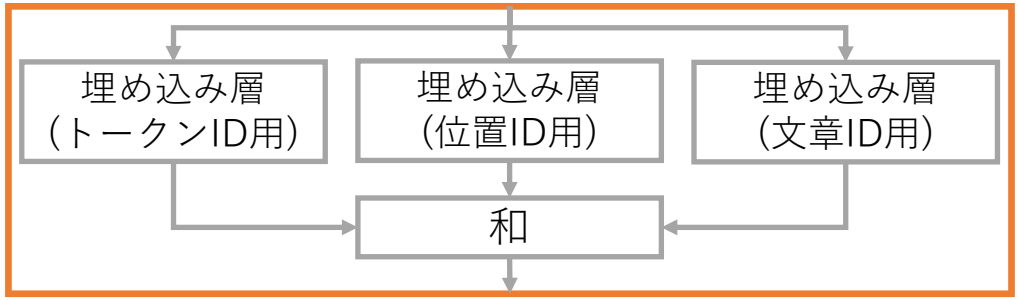


2. BERTの構造



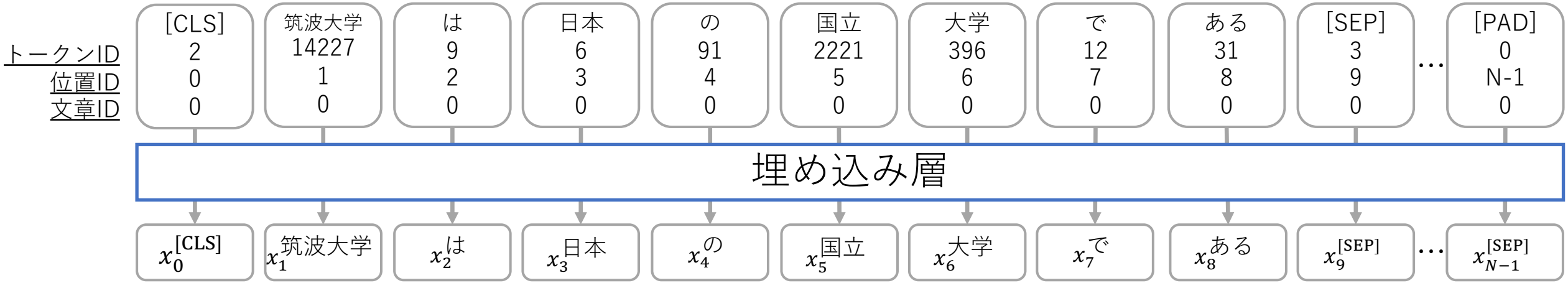
※注意層数lは
8や12の場合が多い。

2.A 埋め込み

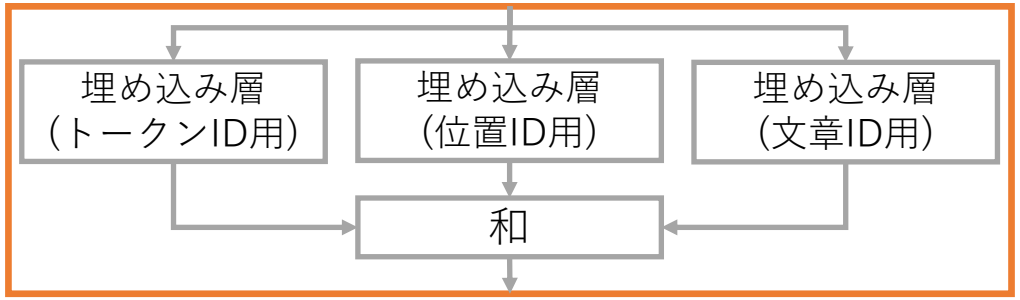


埋め込みとは
トークンIDなどの整数値はニューラルネットでは扱いにくいため、連続値ベクトルに変換することが多い。
この連続値ベクトルのことを埋め込みと呼ぶ。

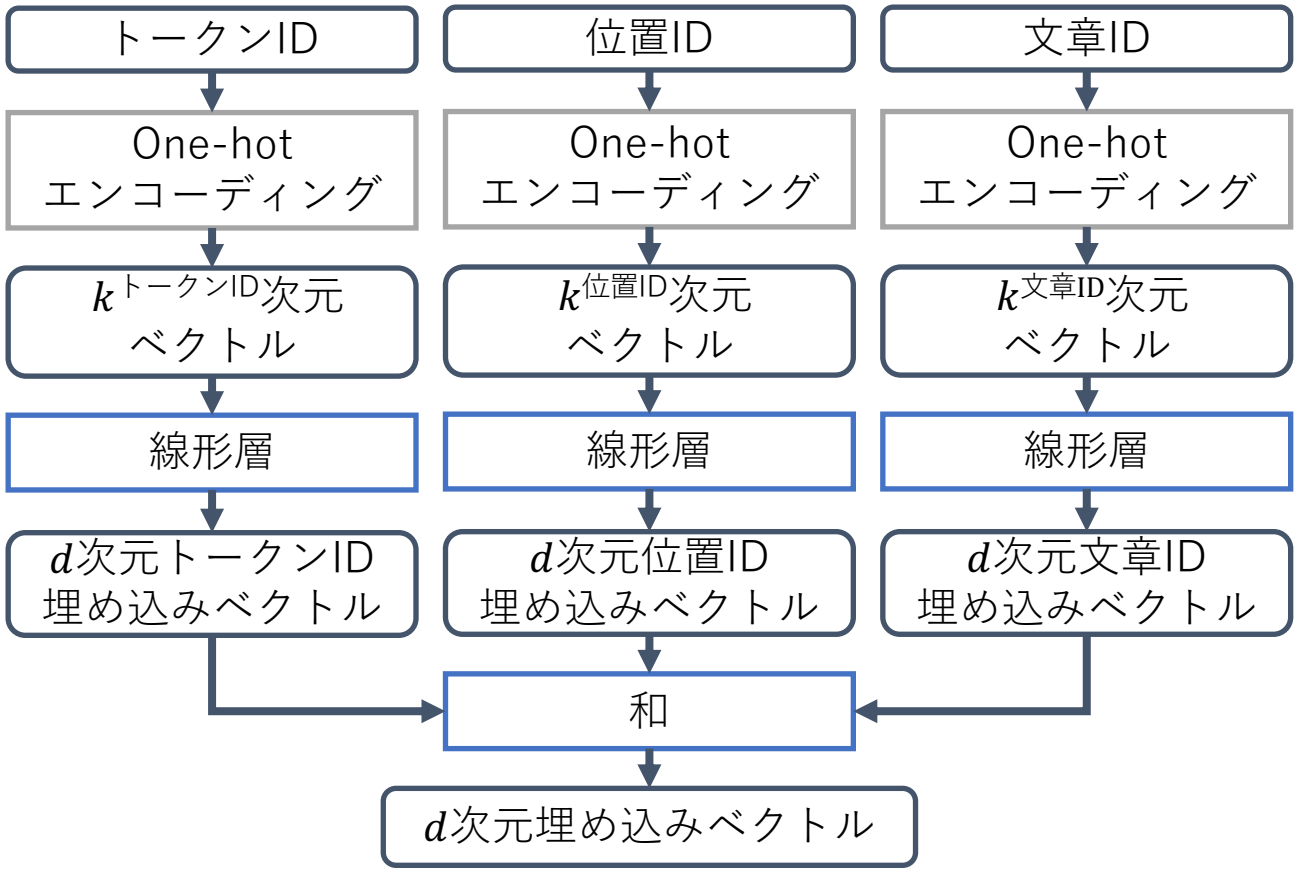
各トークンごとにd次元の埋め込みを得る。



2.A 埋め込み



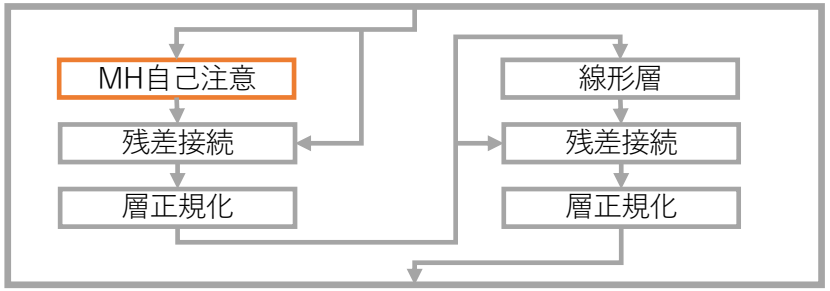
各埋め込み層は
One-hotエンコーディングと線形層からなる。



One-hotエンコーディングとは
対応するインデックスのみ1、それ以外0のベクトルに変換する。
次元数kは、入力整数値の範囲に対応する。
$k^{トークンID}$⇨全体の語彙数
$k^{位置ID}$⇨入力長
$k^{文章ID}$⇨1入力に含まれる文章数

2.B 自己注意機構

Self-Attention Mechanism



自己注意(Self-Attention)とは
入力ベクトルを、**文脈を加味した**ベクトルに変換する機構

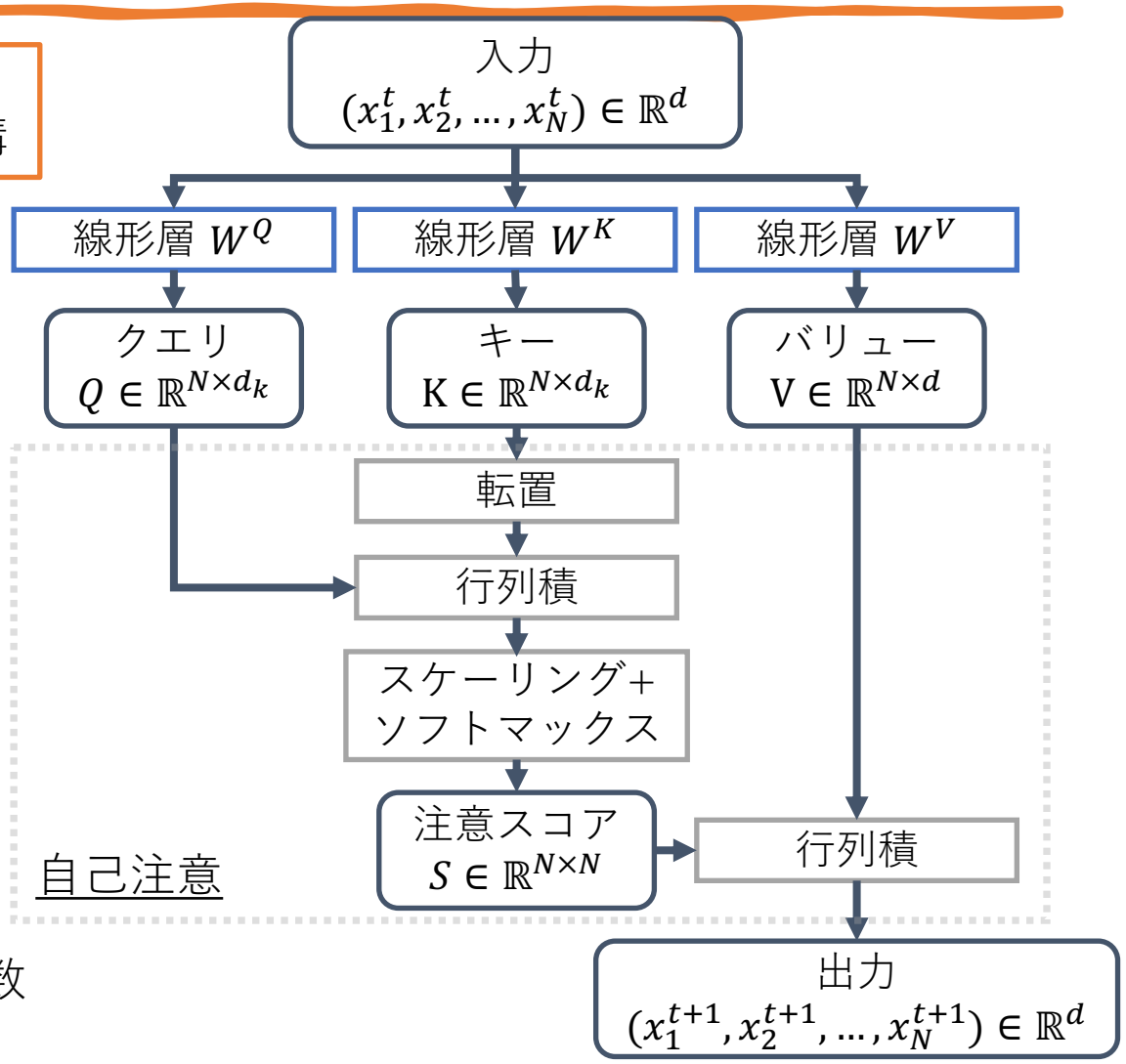
以前から、自己注意はLSTMの補助等で活用されていたが、Attention Is All You Need(2017)で提案されたTransformerでモデルの核として利用されて以来画像処理等でも広く使用されるようなった。

自己注意を数式で表すと

$$\mathrm{Attention}(Q,K,V)=\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

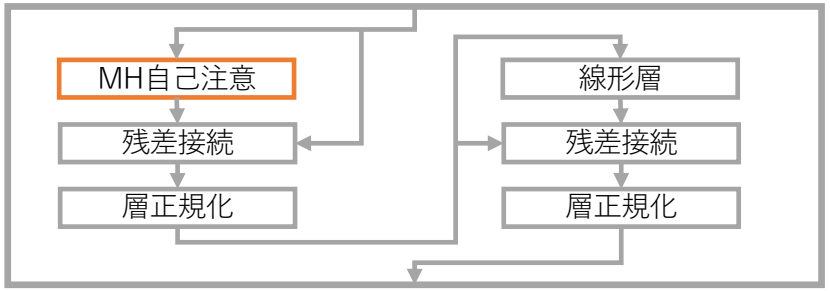
d_k：Query, Keyのベクトル次元数
$\sqrt{d_k}$：次元数d_kが大きいと内積値QK^Tも大きくなりやすい
次元数の影響を調節するスケーリング因子

ソフトマックスは値を0~1の範囲、合計1に正規化する関数



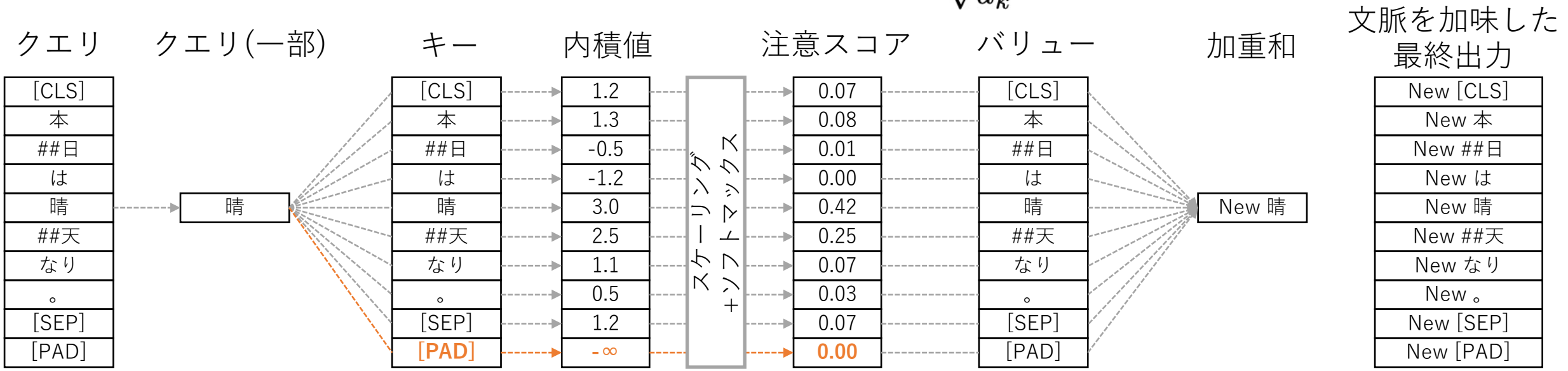
2.B 自己注意機構

Self-Attention Mechanism



どのように文脈を加味したベクトルを得ているのか図解的に説明していく。

$$\mathrm{Attention}(Q,K,V)=\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



わかりやすさのため、一旦クエリのうち1ベクトルのみで考える。

各キーと内積を取る

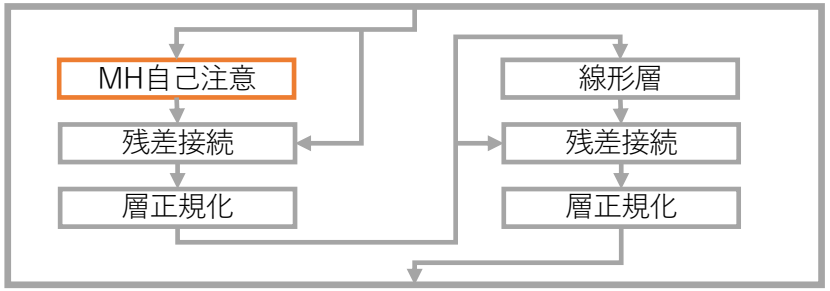
[PAD]とのスコアは-∞で置き換える。
(この時に注意マスクが使用される)
⇨ソフトマックスを通した際に値が限りなく0に近くなる。

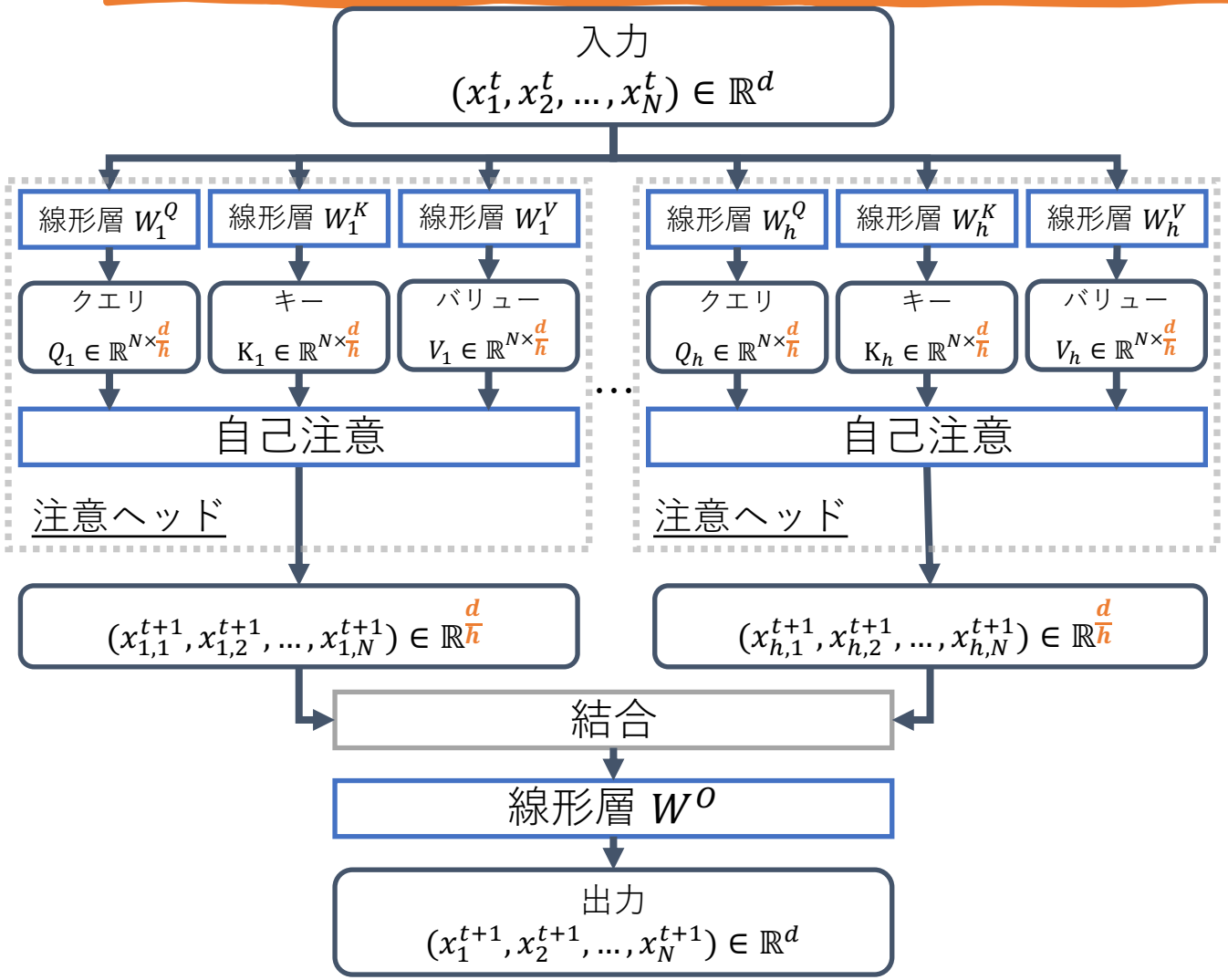
注意スコアをもとにバリューの加重和を取る。
⇨**文脈を加味した埋め込み**を得ることができる。

全てのクエリで同じ処理をすることで、最終出力を得る。

2.B マルチヘッド自己注意

Multi-Head Self-Attention





マルチヘッド自己注意とは

複数の自己注意を並列で行うことで、単一の場合より**多くの文脈関係を扱うことができる**ようになる手法。

線形層を用いて、入力$N\times d$を$N\times\frac{d}{h}$次元に圧縮。

ここでhはdの約数。

h個の注意ヘッドを用いることで出力$\mathrm{h}\times N\times\frac{d}{h}$を得た後、結合し$N\times d$とする。

結合位置を隠すため線形層に通し最終出力を得る。

式で表すと以下のようになる。

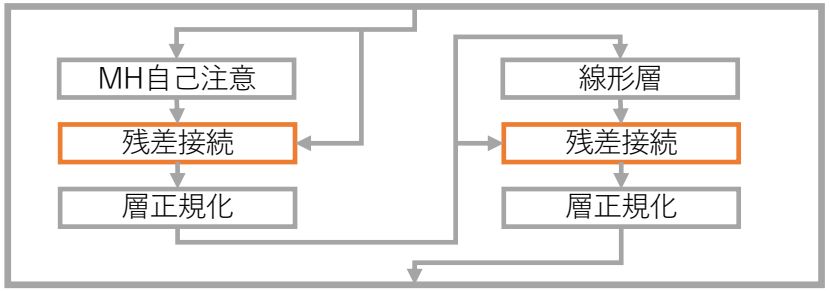
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1,\dots,\mathrm{head}_\mathrm{h})W^O$$

$$\mathrm{where\ head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

W_i^Q, W_i^K, W_i^V：i番目の頭の次元圧縮用の線形層の重み

W^O：結合の境目を隠すための線形層の重み

2.C 残差接続
Residual Connection



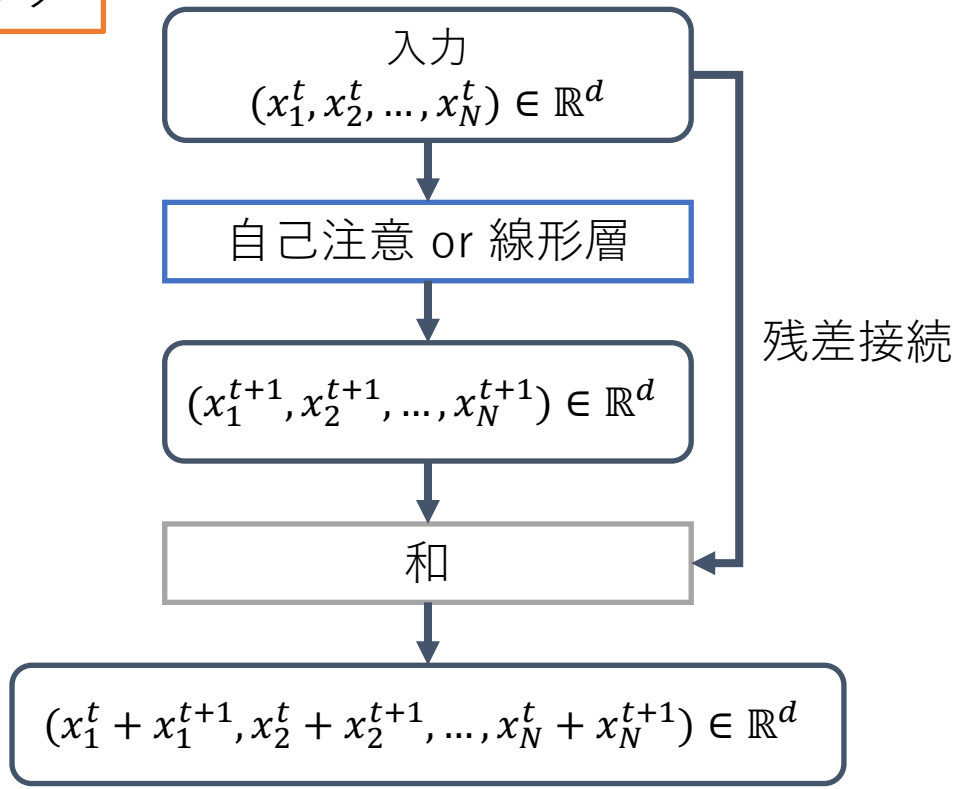
残差接続とは
深い(つまり多層の)ネットワークを**安定して学習**するためのテクニック

Deep Residual Learning for Image Recognition(2015)で提案された。

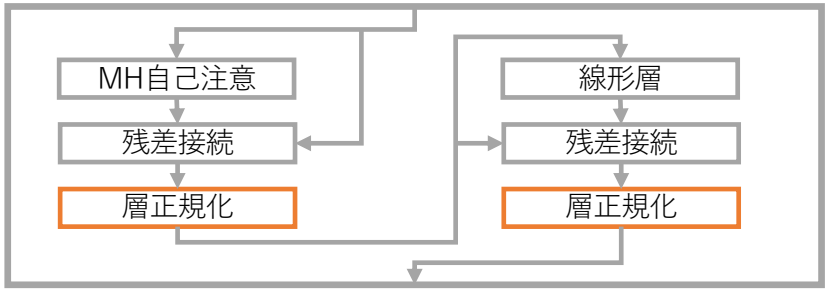
何かしらのネットワーク(今回の場合は自己注意か線形層)の入力 x^t と出力 x^{t+1} の和 $x^t + x^{t+1}$ を最終出力とする。

この場合、モデルは入力との差分x^{t+1}を学習すれば良いため全体のネットワークが深くなっても**学習が安定**する。

やや詳しく説明するとニューラルネットは層が深くなればなるほど、誤差逆伝播の際に勾配が消失しやすくなることが知られている。
残差接続はこの勾配消失を防ぐ効果を持っており、結果として深いネットワークの学習の安定に繋がる。



2.D 層正規化
Layer Normalization



層正規化とは
ネットワーク出力の分布変動を抑制することで、学習を**安定化&高速化**できる手法。

Layer Normalization(2016)で提案された。

1次元に結合した残差接続後の出力：$a^t = concat(x_1^t, x_2^t, \ldots, x_N^t),\ a^t \in \mathbb{R}^{Nd}$

平均と分散を用いて、出力を平均0、分散1に正規化する。

$$\text{平均：}\ \mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \qquad \text{分散：}\ \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^t - \mu^t)^2}$$

$$h^t = \frac{\mathbf{g}}{\sigma^t} \odot (a^t - \mu^t) + \mathbf{b}$$

$\odot$：要素積
$(\mathbf{g}, \mathbf{b}) \in \mathbb{R}^{Nd}$：学習可能なベクトル

$$(x_1^{t\prime}, x_2^{t\prime}, \ldots, x_N^{t\prime}) = \mathrm{decompose}(h^t)$$

3 事前学習
Pre-Training

事前学習とは
モデルを大規模データセットで学習した後、解きたいタスクに転用することで**収束を早めたり精度を向上させる**ことができる学習手法。

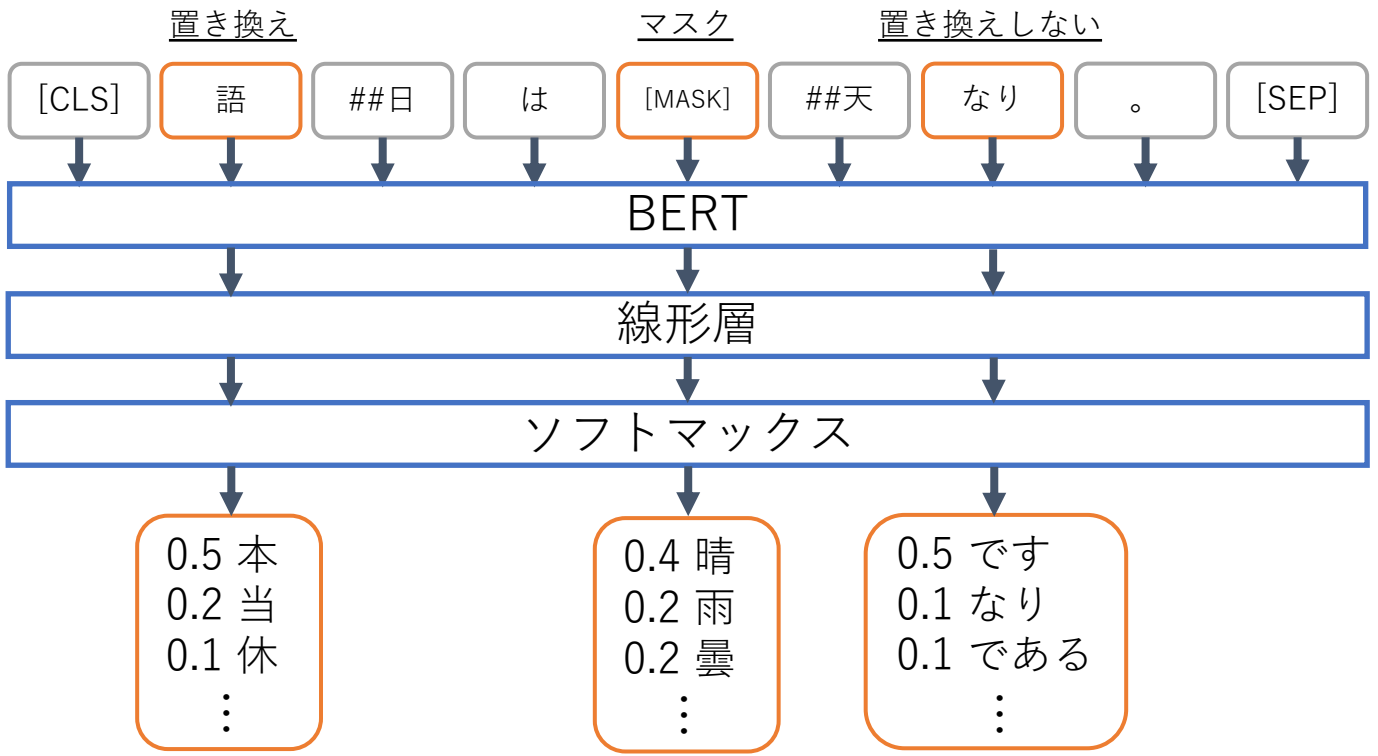
膨大な計算コストを必要とするため、一般的には配布されている事前学習済みのBERTモデルを使用することが多い。
本ワークショップでも東北大が公開している日本語事前学習済みBERTを使用する。

BERTの事前学習は
A. マスク言語モデリング (Masked Language Modeling)
B. 次文予測 (Next Sentence Prediction) ⇦あまり効果がないことがわかっているため説明を省く※
からなる。

※RoBERTa: A Robustly Optimized BERT Pretraining Approach(2019)

3.A マスク言語モデリング
Masked Language Modeling

一定の確率で単語を[MASK]に置き換えて単語を当てるタスク。
Wikipediaなどの大規模コーパスで事前学習を行う。



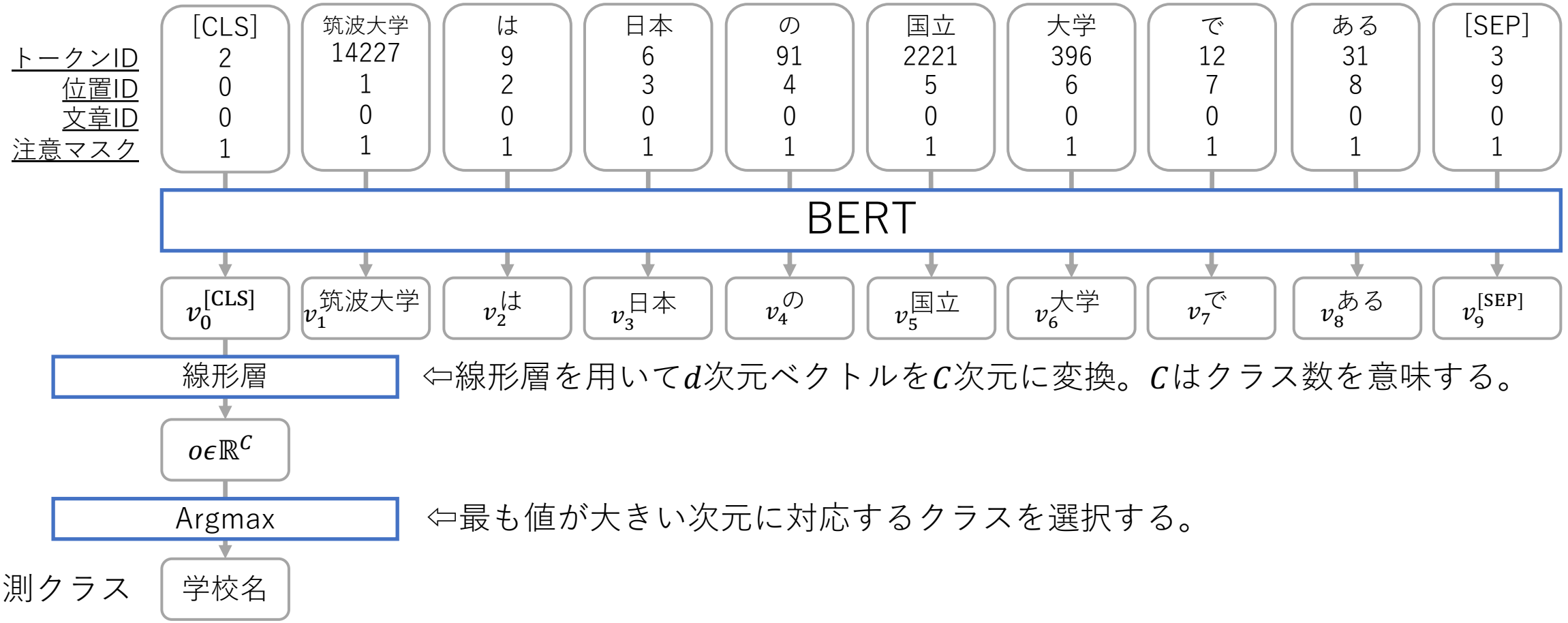
全体の15%の単語が対象となる。

そのうち、
80%を[MASK]トークン
10%を別のトークン
10%を同じトークン
に置き換える。

BERTは対象の単語が何であったか
を当てる必要がある。

4 分類タスクへの適用

分類タスクの場合、[CLS]トークンに対応する出力ベクトルを用いてラベル予測を行う。



森羅分類データセットの説明

森羅分類データセットとは

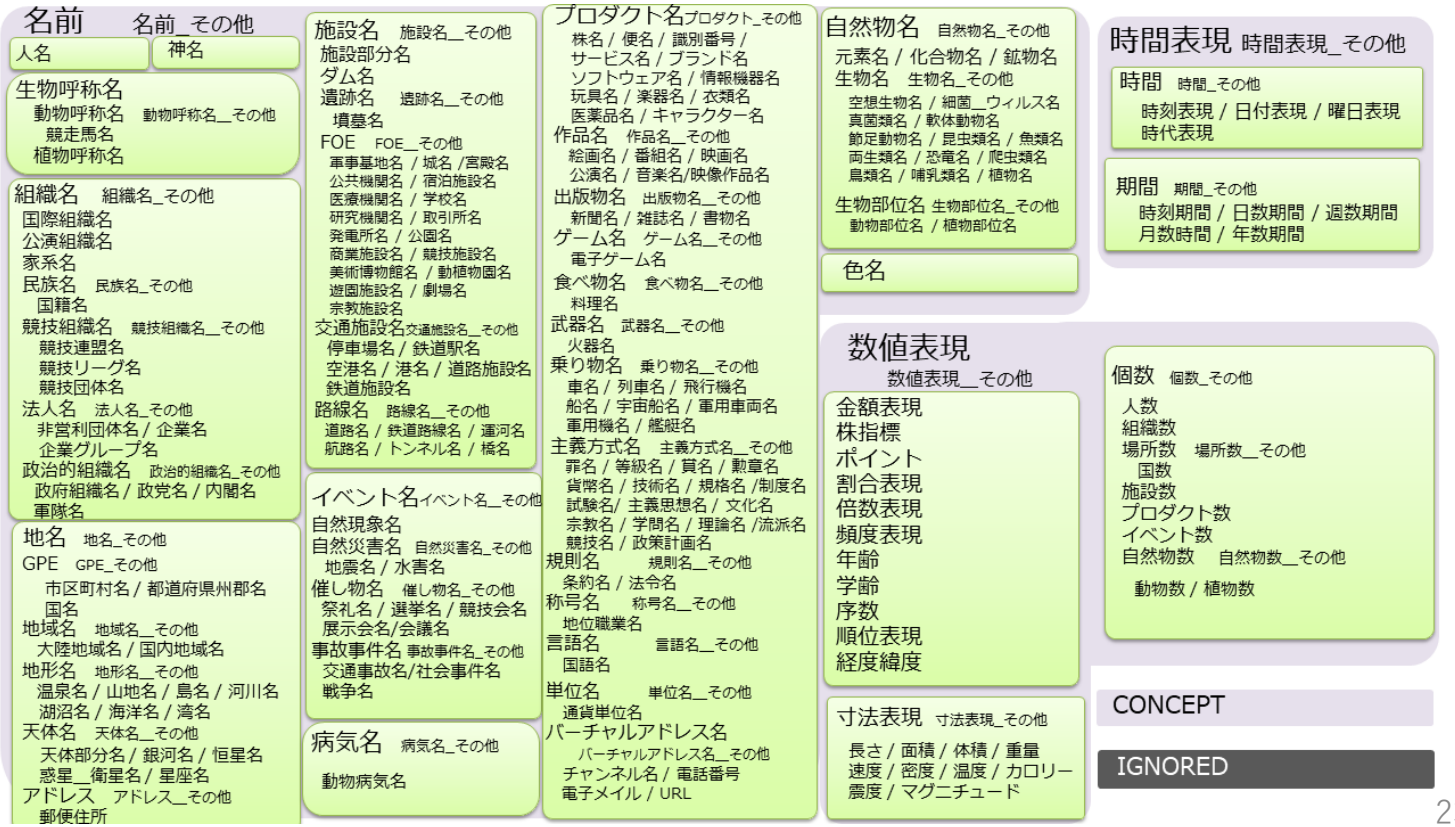
今年度開催している森羅共有タスクのデータセットの一つ。
日本語Wikipediaを拡張固有表現に分類する。

日本語Wikipedia

分類

ラベルは複数付与される場合がある。
例えば、映画化された書籍の場合、
書物名、映画名が付与される。
全体の約3%に複数ラベルが付与されている。

拡張固有表現階層(約200カテゴリー)



森羅分類データセットとは

学習データ：分類済み日本語Wikipedia(2019年時点)
予測データ：未分類日本語Wikipedia(2021年時点)

学習データは100万件近く存在する。
計算コストの都合上、ワークショップでは5000件をランダムにサンプリングし学習データとする。
また、簡易化のため約3%の複数ラベルを除去し、単一ラベルのデータセットとして扱う。

本ワークショップでは学習、予測、リーダーボードによる評価まで行う。

いざ実践

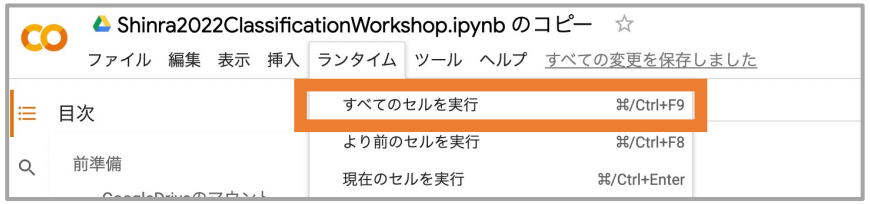
まずは動作確認

①Colabを開きます。(Googleアカウントが必要です。)

https://colab.research.google.com/drive/1M2xbF4gqCH5O7LnmnAvVOkbBO22SiX8K?usp=sharing

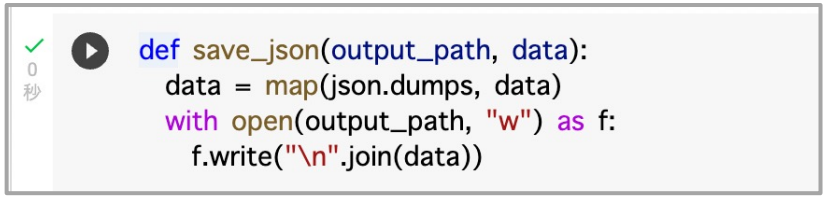
②自分のGoogleDriveにコピーします。(コードの編集が可能になります。)

③ランタイム>全てのセルを実行を押す



④ポップアップが出るのでGoogleドライブに接続

④5分ほど待って最後行のコードの横に緑のチェックがつけば成功



もし動かない場合はSlack
#bert_workshop2022チャンネルでご相談ください🙇