

AI技術共有会 発表資料
BERT入門

2020/1/9

株式会社ディー・エヌ・エー
AIシステム部 データサイエンス 2G
松井 健一



Delight and Impact the World

アジェンダ

1

自己紹介

2

GLUE Leaderboard

3

BERTの工夫ポイント

4

学習方法の工夫

5

モデルの構造

自己紹介：松井健一（Ken'ichi Matsui）

株式会社 ディー・エヌ・エー AIシステム部
データサイエンス 2G グループマネージャー

経歴

- ✓ 大手SIer ⇒ 大手通信キャリア ⇒ 外資系コンサルティングファーム
⇒ 現職

主な職務経験

- DRIVE CHARTにおける危険シーン検知の開発
- センサーデータによる異常検知
- 地理情報解析による人口動態分析
- Deep Learning/機械学習による画像解析
- SNS解析による商品評判解析
- ドライブデータの解析 (DRIVE CHART)
- 「アクセントニアのプロフェッショナルが教える データ・アナリティクス実践講座」共著



ブログ (Qiita: 技術系ブログサイト)

2015年～2016年頃は統計、機械学習、プログラミングに関するブログをよく書いていました。

まつけん
@kenmatsu4

会社員です。データ解析のことや、統計学のこと、機械学習などについて書いています。【今まで書いた記事一覧】
<http://qiita.com/kenmatsu4/items/623514c61166e34283bb> 【English Blog】
<http://kenmatsu4.tumblr.com>

kenmatsu4 が2017/12/29に投稿
moviepyによる動画変換処理 with ndarray

kenmatsu4 が2017/12/18に投稿
EMアルゴリズム徹底解説

kenmatsu4 が2017/07/19に投稿
Variational Autoencoder徹底解説

kenmatsu4 が2017/06/30に投稿
pythonからdlmを呼び出して時変係数回帰モデルを実行する

昔はPythonタグで1位になったこともありました。今3位。

2019年はcontribution数で、記事投稿者約5万人中21位でした。

Python

69 Items 14188 Contribution

24.8% 22.9% 13.7% 12.6% 10.3% 5.3%

Python statistics 統計学 機械学習 MachineLearning 数学 Twitter 自然言語処理 matplotlib Others

まつけん
@kenmatsu4

32625 記事 63209 フォロワー

Python

32625 記事 63209 フォロワー

About

- 公式サイト: [Top - python.jp](#)
- 公式リファレンス: [3.7.4 Documentation](#)
- Wikipedia: [Python - Wikipedia](#)

トレンド 先週いいねの多かった記事

旅行予約サイトの「今あなた以外に〇〇人が見ています」はウソなのか
by piyoSakai 218 Python, Selenium, 統計

Qiita ホーム コミュニティ キーワードを入力

Qiita ホーム コミュニティ キーワードを入力

Qiita ホーム コミュニティ キーワードを入力

Qiita のいろいろランキング2019

ランク	前年比	ユーザ名	総Contribution	前年比	記事数	平均いいね	Top数	Top占有率	Top記事
2	+3	@kenmatsu4	14749	+2699	71	207.732	1768	11.99%	【機械学習】ディープラーニングフレームワークChainerを試しながら解説してみる。

Kaggle戦歴

Kaggleで銀メダル5つ（ソロ2つ、チーム3つ） 2019年11月現在 125,564人中 627位 (top 0.5%)



kenmatsu4
Data Scientist at DeNA
Tokyo, Japan
Joined 5 years ago · last seen in the past day
[Twitter](#) [LinkedIn](#)

Home Competitions (18) Datasets (1) Kernels (23) Discussion (32) ... Edit Profile

Competitions Expert	
Current Rank 627 of 125,564	Highest Rank 564
PLAsTiCC Astronomical Classification • a year ago · Top 2% Predicting Molecular Properties • 3 months ago · Top 2% Microsoft Malware Prediction • 8 months ago · Top 2%	
Datasets Contributor	
Unranked	
petfinder_state_imbalance 8 months ago	
NN outputs gauge • 16 days ago	
Basic data configuration • a year ago	
3d structure plots • 4 months ago	
Competitions Expert	
Current Rank 214 of 105,545	Highest Rank 213
PLAsTiCC Astronomical Classification • a year ago · Top 2% Predicting Molecular Properties • 3 months ago · Top 2% Microsoft Malware Prediction • 8 months ago · Top 2%	
Kernels Expert	
Discussion Contributor	
Unranked	
43 place solution • 3 months ago	
Adversarial IEEE • 2 months ago	
Basic data configuration • a year ago	
Discussion Contributor	

PLAsTiCC Astronomical Classification
Can you help make sense of the Universe?
Featured · a year ago · time series, astronomy, tabular data
 19/1094 Top 2%

Predicting Molecular Properties
Can you measure the magnetic interactions between a pair of atoms?
Featured · 3 months ago · chemistry, tabular data, regression
 43/2749 Top 2%

Microsoft Malware Prediction
Can you predict if a machine will soon be hit with malware?
Research · 8 months ago
 43/2426 Top 2%

Home Credit Default Risk
Can you predict how capable each applicant is of repaying a loan?
Featured · a year ago · home, banking, tabular data
 79/7190 Top 5%

IEEE-CIS Fraud Detection
Can you detect fraud from customer transactions?
Research · 2 months ago · tabular data, binary classification
 287/6381 Top 5%

Avito Demand Prediction Challenge
Predict demand for an online classified ad
Featured · a year ago · tabular data, image data, text data
 103/1871 Top 6%

PetFinder.my Adoption Prediction
How cute is that doggy in the shelter?
Featured · Code Competition · 7 months ago · image data, text data
 192/2023 Top 10%

Quora Insincere Questions Classification
Detect toxic content to improve online conversations
Featured · Code Competition · 9 months ago · text data, binary classification
 229/4037 Top 6%

Santander Value Prediction Challenge
Predict the value of transactions for potential customers.
Featured · a year ago · finance, banking
 317/4477 Top 8%

Santander Customer Transaction Prediction
Can you identify who will make a transaction?
Featured · 7 months ago · banking, tabular data, binary classification
 631/8802 Top 8%

日本での実績 SIGNATE

✓ 産業技術総合研究所 衛星画像分析コンテスト 2位入賞

<https://www.slideshare.net/matsukenbook/signate-108228406>

アジェンダ

1

自己紹介

2

GLUE Leaderboard

3

BERTの工夫ポイント

4

学習方法の工夫

5

モデルの構造

GLUE (General Language Understanding Evaluation) Leaderboard

GLUE SuperGLUE Paper </> Code Tasks Leaderboard FAQ Diagnostics Submit Login

top10のほとんどがBERTに影響を受けている！

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	ERNIE Team - Baidu	ERNIE		90.1	72.2	97.5	93.0/90.7	92.9/92.5	75.2/90.8	91.2	90.6	98.0	90.4	94.5	49.4
2	Microsoft D365 AI & MSR AI & GATECHMNT-DNN-SMART			89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	50.2
3	T5 Team - Google	T5		89.7	70.8	97.1	91.9/89.2	92.5/92.1	74.6/90.4	92.0	91.7	96.7	92.5	93.2	53.1
4	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.5	71.3	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.7	90.4	99.2	87.4	91.8	48.4
5	XLNet Team	XLNet (ensemble)		89.5	70.2	97.1	92.9/90.5	93.0/92.6	74.7/90.4	90.9	90.9	99.0	88.5	92.5	48.4
6	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5	91.3	91.0	99.2	89.2	91.8	50.2
7	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3	91.1	90.7	98.8	88.7	89.0	50.1
8	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	98.9	88.2	89.0	48.7
9	Junjie Yang	HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7	90.4	95.5	87.9	89.0	49.3
10	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9	87.4	96.0	86.3	89.0	42.8

ということで
時代を席巻している
BERTに入門してみようと思ひます

【参考】GLUEのタスク

下記の11つのタスクでスコアリングされる

#	Name	略称	Metric	概要
1	The Corpus of Linguistic Acceptability	CoLA	Matthew's Corr	文章として自然かどうかの2値分類
2	The Stanford Sentiment Treebank	SST-2	Accuracy	映画のレビュー文から、ポジネガの2値分類
3	Microsoft Research Paraphrase Corpus	MRPC	F1 / Accuracy	2つの文章が等価であるかの2値分類
4	Semantic Textual Similarity Benchmark	STS-B	Pearson-Spearman Corr	2つの文章の類似度を0-5のスコアで回帰
5	Quora Question Pairs	QQP	F1 / Accuracy	2つの質問文が等価であるかの2値分類
6	MultiNLI Matched	MNLI-m	Accuracy	2つの文章からcontradiction, entailment, neutralの3値分類 (trainセットに存在するgenreの文章)
7	MultiNLI Mismatched	MNLI-mm	Accuracy	2つの文章からcontradiction, entailment, neutralの3値分類 (trainセットに存在しないgenreの文章)
8	Question NLI	QNLI	Accuracy	Wikipediaの文章。質問に対して文章が答えを含んでいるかの2値分類
9	Recognizing Textual Entailment	RTE	Accuracy	2つの文章が等価であるかの2値分類
10	Winograd NLI	WNLI	Accuracy	1 of 2単語を変えると分が指し示す単語が大きく変わるものがある。それを当てる問題
11	Diagnostics Main	AX	Matthew's Corr	contradiction, entailment, neutralの3値分類 (train dataなし。GLUEで手動でキュレーションした)

アジェンダ

- 1 自己紹介
- 2 GLUE Leaderboard
- 3 BERTの工夫ポイント
- 4 学習方法の工夫
- 5 モデルの構造

BERTの工夫ポイント

BERTにおける工夫の概要は大きく学習の工夫と、モデルの工夫に分けられる。

1

学習方法の工夫

- ✓ Pre-training実施後にFine tuningする、という2段階の学習を行う設計により、NLPタスクの学習済みモデルを構築可能に
- ✓ Pre-trainingに
 1. Masked Language Model (Mask LM)
 2. Next Sentence Prediction (NSP)の2つのタスクを設定し、教師ラベルなしのpre-trainingが可能に

2

モデルの工夫

- ✓ TransformerのEncoder部分をベースにモデルを構築
- ✓ Transformerが6段stackだったブロックを
 - base : 12段
 - large : 24段に多段化
- ✓ Positional Encodingはsin/cosを使うものから通常のembeddingに

アジェンダ

1 自己紹介

2 GLUE Leaderboard

3 BERTの工夫ポイント

4 学習方法の工夫

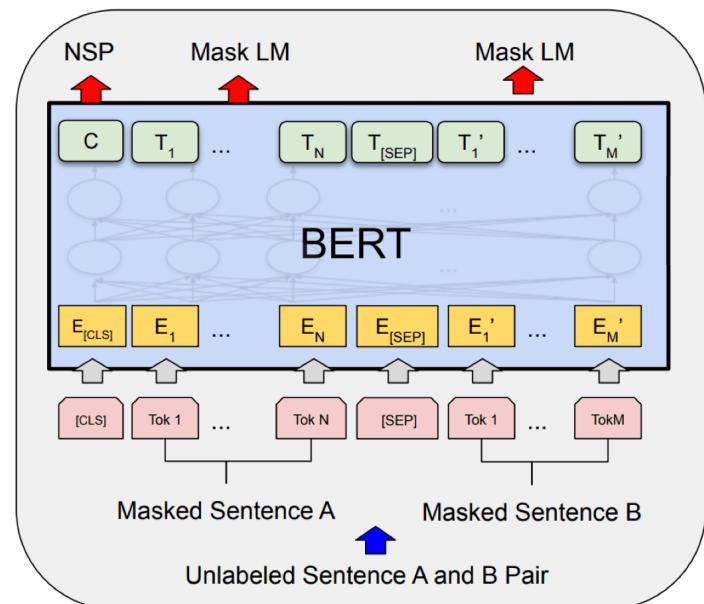
5 モデルの構造

BERTのPre-trainingとFine tune

Pre-trainingとFine tuneの2段階に学習を分けることで、大量のラベルなしデータを活用したPre-trainingで学習し、転移学習を可能にした。

Pre-training (unlabeled)

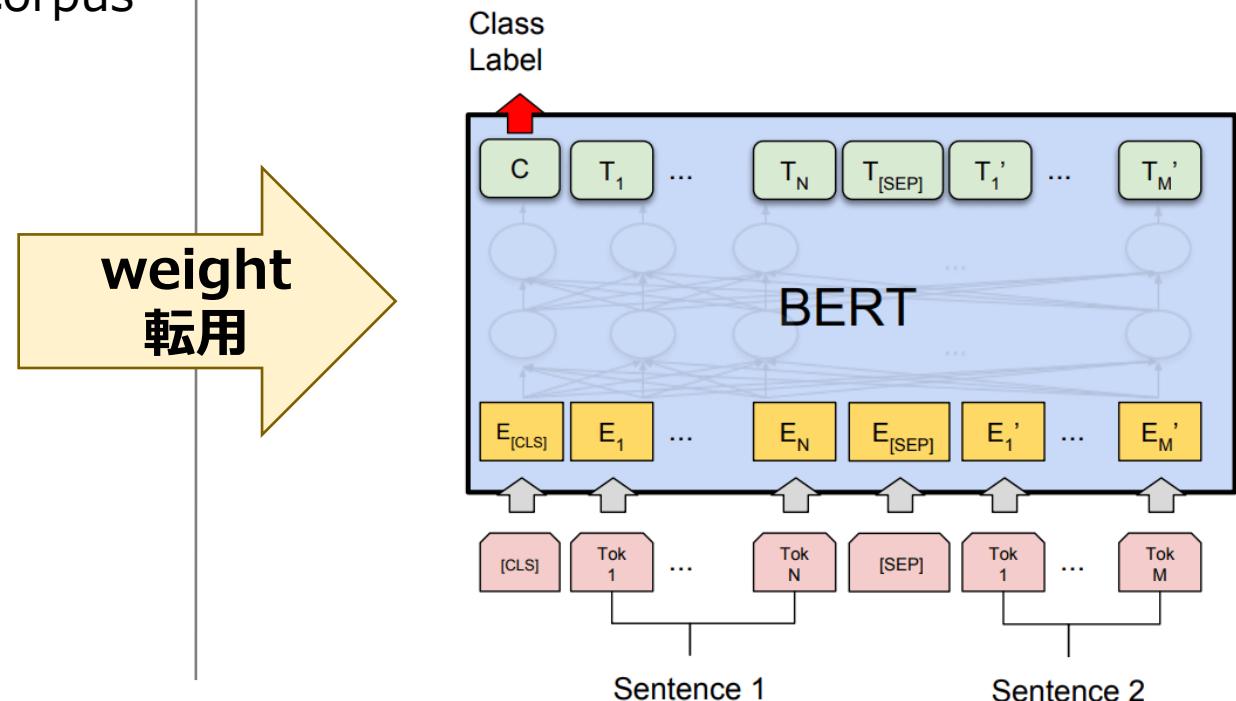
- ✓ ラベルなしで学習可能な、(1)Masked Language Model(Mask LM)と、(2)Next Sentence PredictionでPre-training(NSP)を行い、モデルのweightを保存。
- ✓ データセット： Wikipedia(25億語)、 Book Corpus(8億語)



Fine tune (labeled)

Pre-trainingで学習したweightを転移学習として利用。BERTモデルの上に追加的なLayerを設けて目的のタスクに適用することが可能。

2つの文章から分類を行う例



BERTの tokenize

pre-trainedのTokenizerで単語を分割。WordPieceを利用して低頻度語は分割して tokenize。
position_idもモデルにインプットする(後述)。 pre-trained tokenizerの語彙数は30522。

	position_id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
text 1	text	two	kids	are	playing	in	a	swimming	pool	with	a	green	colored	crocodile	float	.
	ids	2048	4268	2024	2652	1999	1037	5742	4770	2007	1037	2665	6910	21843	14257	1012
text 2	position_id	0	1	2	3	4	5	6	7	8	9	10	11	12		
	text	two	kids	push	an	in	##fl	##atable	crocodile	around	in	a	pool		.	
	ids	2048	4268	5245	2019	1999	10258	27892	21843	2105	1999	1037	4770		1012	

(2つのtextはSTS-B のindex 977より)
(ちなみに類似度は5段階の4.2)

WordPieceモデルを利用して低頻度語を分割して tokenize.
分割された単語は語頭に ## が付いている。
ex: "inflatable" = "in" + "##fl" + "##atable"
(inflatable : 膨らませることができる)

複数センテンス対応

文頭に[CLS]、センテンスの間と文末に[SEP]を入れ込み、複数センテンスを表現。

	position_id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
text 1	text	two	kids	are	playing	in	a	swimming	pool	with	a	green	colored	crocodile	float	.
text 1	ids	2048	4268	2024	2652	1999	1037	5742	4770	2007	1037	2665	6910	21843	14257	1012
text 2	position_id	0	1	2	3	4	5	6	7	8	9	10	11	12		
text 2	text	two	kids	push	an	in	##fl	##atable	crocod ile	aroun d	in	a	pool		.	
text 2	ids	2048	4268	5245	2019	1999	10258	27892	21843	2105	1999	1037	4770	1012		



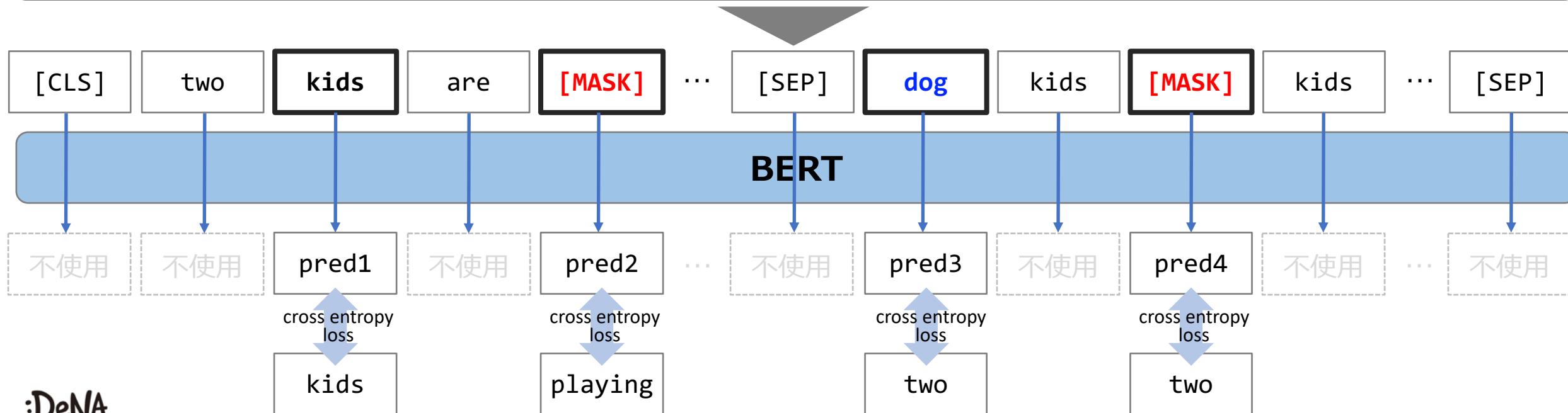
```
[ '[CLS]', 'two', 'kids', 'are', 'playing', 'in', 'a', 'swimming', 'pool', 'with', 'a',  
'green', 'colored', 'crocodile', 'float', '.', '[SEP]', 'two', 'kids', 'push', 'an',  
'in', '##fl', '##atable', 'crocodile', 'around', 'in', 'a', 'pool', '.', '[SEP]' ]
```

Pre-training 1 : Masked Language Model

指定の置き換えロジックにより文章を差し替え、オリジナル。単語を予測する問題。選択された単語の部分をcross entropyでlossをとる。これなら正解ラベルが不要(自動生成可能)。

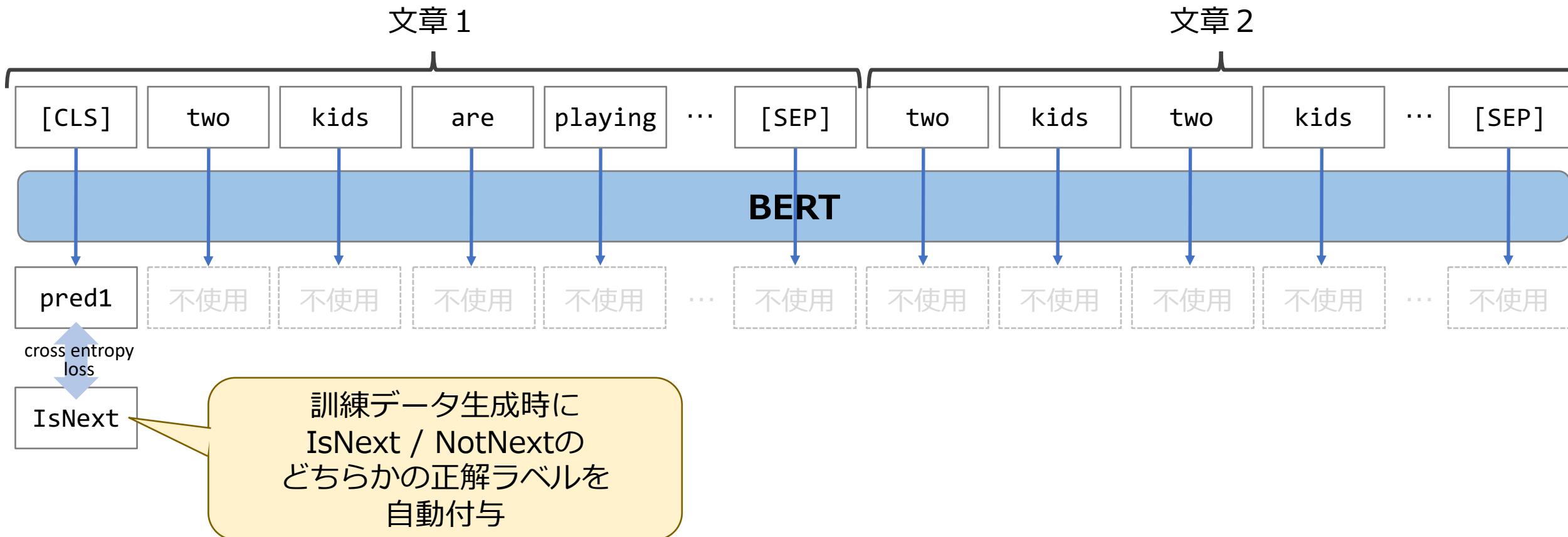


- (1) 15%の確率でランダムに選択, (2) そこから80%の確率で**[MASK]**に置き換え,
(3) そこから10%の確率で**ランダムトークン**に置き換え, (4) 10%の確率で**置き換えない**



Pre-training 2 : Next Sentence Prediction

2つの文章をインプットにして、文章2が文章1の次の文章であるかどうかの2値分類タスク。これも正解ラベルが不要(自動生成可能)。



【参考】BERTで使われる特殊文字

```
# 特殊文字
unk_token = "[UNK]",    # 未知の単語
sep_token = "[SEP]",    # 文章と文章の間の区切り
pad_token = "[PAD]",    # 末尾のパディング
cls_token = "[CLS]",    # 文頭を表す
mask_token = "[MASK]",  # pre-trainingに使うマスク
```

https://github.com/huggingface/transformers/blob/master/src/transformers/tokenization_bert.py#L141-L145

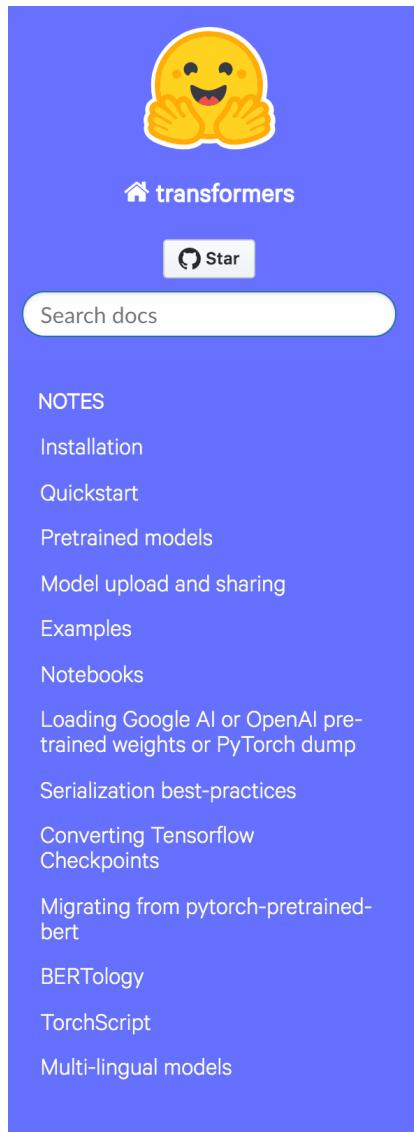
アジェンダ

- 1 自己紹介
- 2 GLUE Leaderboard
- 3 BERTの工夫ポイント
- 4 学習方法の工夫
- 5 モデルの構造

huggingface / transformers

非常に多くのTransformer系モデルの実装とpre-trained weightを提供しているパッケージ

<https://huggingface.co/transformers/index.html>



[View page source](#)

対応モデル

PACKAGE REFERENCE

AutoModels

BERT

OpenAI GPT

Transformer XL

OpenAI GPT2

XLM

XLNet

RoBERTa

DistilBERT

CTRL

CamemBERT

ALBERT

BERTも
含まれる

forwardするだけのデモコード

TF版BERTを動かしてみるための最小デモコード。pretrainedモデルのweightを読み込み、そのモデルでforwardする。PyCharmでステップ実行すると動作が確認できて便利。

```
import tensorflow as tf
from transformers import BertTokenizer, TFBertForSequenceClassification

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
text1 = "[CLS] Two kids are playing in a swimming pool with a green colored crocodile.
float. [SEP]"
text2 = "Two kids push an inflatable crocodile around in a pool. [SEP]"
tokenized_text = tokenizer.tokenize(text1 + " " + text2)
print(tokenized_text)

indexed_tokens = tokenizer.convert_tokens_to_ids(tokenized_text)
pos_sep = tokenized_text.index("[SEP]") + 1 # 上記の例では最初の[SEP]までが1st sentence
segments_ids = [0]*pos_sep + [1]*(len(indexed_tokens)-pos_sep)

tokens_tensor = tf.Variable([indexed_tokens])
segments_tensors = tf.Variable([segments_ids])

model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased')
print(model.summary())
outputs = model(tokens_tensor, token_type_ids=segments_tensors)
print(outputs)
```

tokenize用vocalファイル
がダウンロードされる

pre-trained weight
がダウンロードされる

【参考】直接vocabやweightのファイル入手 [tf_bertの例]

kaggleのkernelコンペでinternet OFF制限の時はこちらから

vocabファイル

https://github.com/huggingface/transformers/blob/master/src/transformers/tokenization_bert.py#L32-L52

```
PRETRAINED_VOCAB_FILES_MAP = {
    "vocab_file": {
        "bert-base-uncased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-uncased-vocab.txt",
        "bert-large-uncased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-large-uncased-vocab.txt",
        "bert-base-cased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-cased-vocab.txt",
        "bert-large-cased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-large-cased-vocab.txt",
        ...
        "bert-base-finnish-cased-v1": "https://s3.amazonaws.com/models.huggingface.co/bert/TurkuNLP/bert-base-finnish-cased-v1/vocab.txt",
        "bert-base-finnish-uncased-v1": "https://s3.amazonaws.com/models.huggingface.co/bert/TurkuNLP/bert-base-finnish-uncased-v1/vocab.txt",
    }
}
```

weightファイル

https://github.com/huggingface/transformers/blob/master/src/transformers/modeling_tf_bert.py#L32-L52

```
TF_BERT_PRETRAINED_MODEL_ARCHIVE_MAP = {
    "bert-base-uncased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-uncased-tf_model.h5",
    "bert-large-uncased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-large-uncased-tf_model.h5",
    "bert-base-cased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-cased-tf_model.h5",
    "bert-large-cased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-large-cased-tf_model.h5",
    "bert-base-multilingual-uncased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-multilingual-uncased-tf_model.h5",
    "bert-base-multilingual-cased": "https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-multilingual-cased-tf_model.h5",
    ...
    "bert-base-finnish-cased-v1": "https://s3.amazonaws.com/models.huggingface.co/bert/TurkuNLP/bert-base-finnish-cased-v1/tf_model.h5",
    "bert-base-finnish-uncased-v1": "https://s3.amazonaws.com/models.huggingface.co/bert/TurkuNLP/bert-base-finnish-uncased-v1/tf_model.h5",
}
```

【参考】PyCharmで動作確認：例) tensorのshape確認

The screenshot shows the PyCharm IDE interface during debugging. The code being run is `modeling_tf_bert.py`, specifically line 377:

```
hidden_states = layer_outputs[0]
```

A yellow callout bubble contains the instruction:

① ここで実行をstopさせて

In the bottom right corner of the code editor, there is a red box highlighting the expression `layer_outputs[0].shape` in the Watches tool window. This expression is shown to have the value `{TensorShape} (1, 31, 768)`. Another yellow callout bubble points to this entry with the instruction:

② `layer_outputs[0]`の
shapeを確認

The bottom left of the screen shows the stack trace in the Frames tool window, with the current frame being `call, modeling_tf_bert.py:377`.

BertConfig class

以降の解説は下記のパラメータが設定されているとして進める。

```
vocab_size=30522,  
hidden_size=768,  
num_hidden_layers=12,  
num_attention_heads=12,  
intermediate_size=3072,  
hidden_act="gelu",  
hidden_dropout_prob=0.1,  
attention_probs_dropout_prob=0.1,  
max_position_embeddings=512,  
type_vocab_size=2,  
initializer_range=0.02,  
layer_norm_eps=1e-12,
```

TFBertモデル class全体構成

huggingfacdeのtransformers実装のTFBertForSequenceClassificationを元に解説。

TFBertForSequenceClassification

TFBertMainLayer

TFBertEmbedding

TFBertEncoder

TFBertLayer

TFBert Attention

TFBertSelfAttention

TFBertSelfOutput

TFBertIntermediate

TFBertOutput

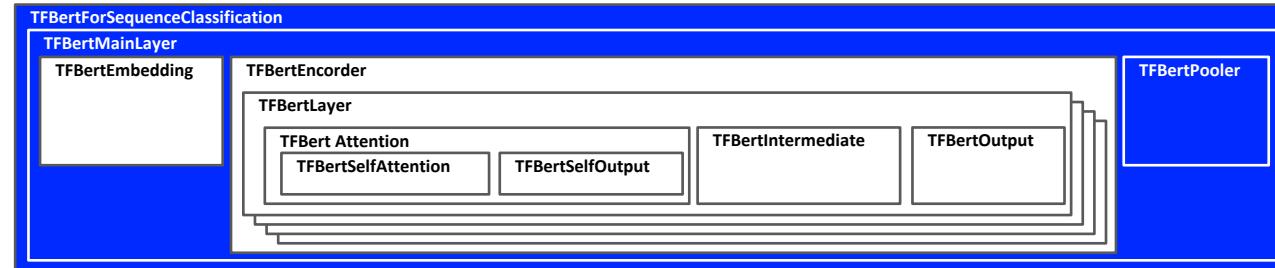
TFBertPooler

データの流れ

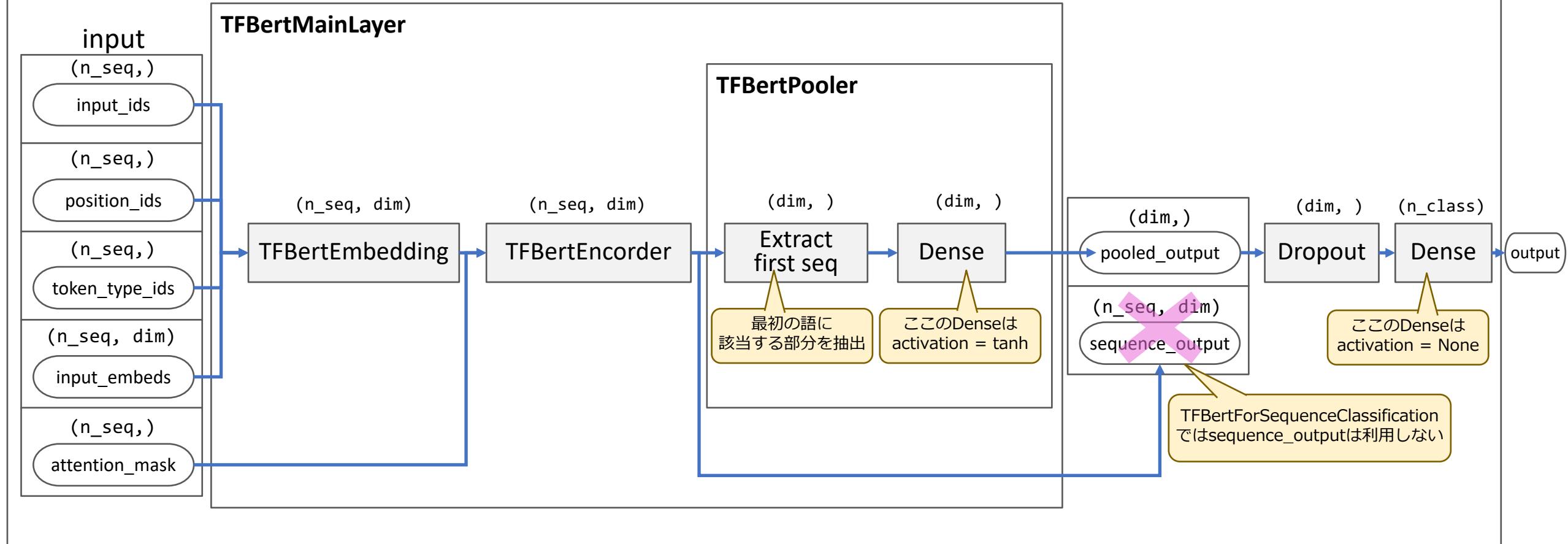
12段stackingする

TFBertForSequenceClassification

BERTの基本モデルにDropout, Denseを加えて分類タスクを解くモデルとしたもの。

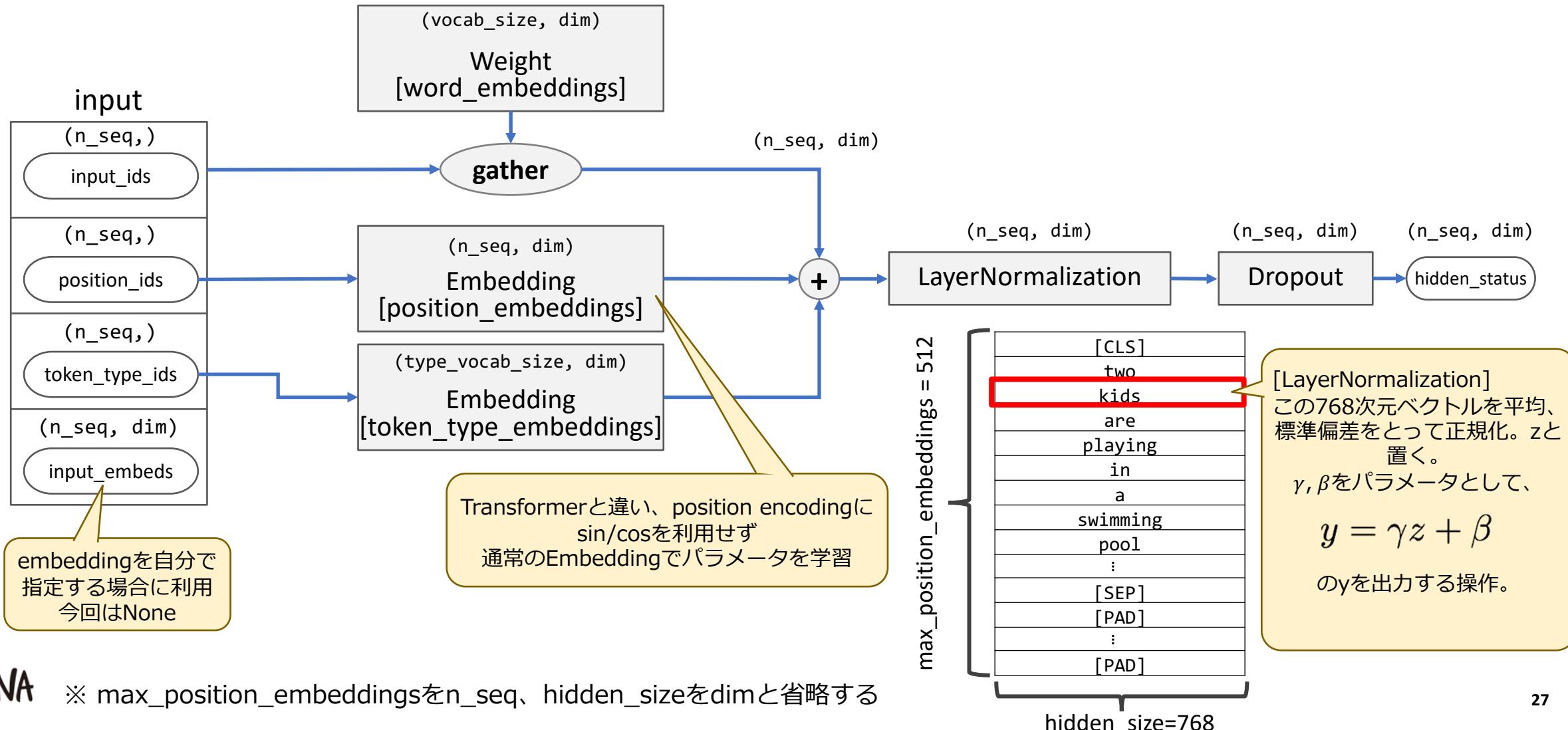
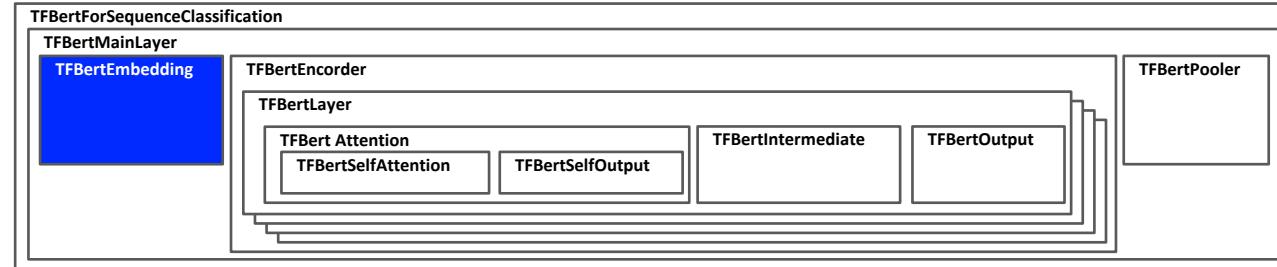


TFBertForSequenceClassification



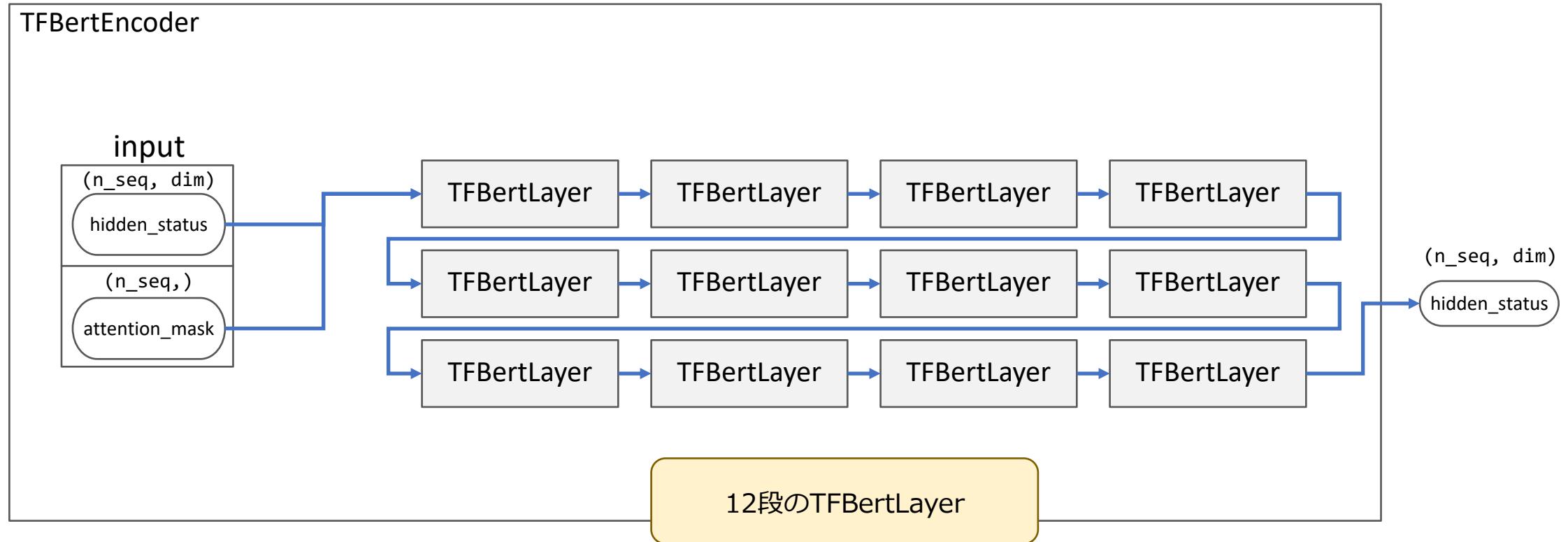
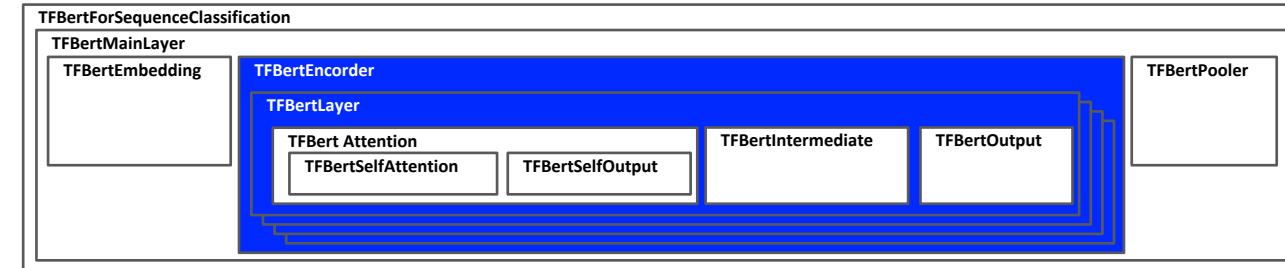
TFBertEmbedding

token, position, token_type(文章順序番号)をembeddingする。



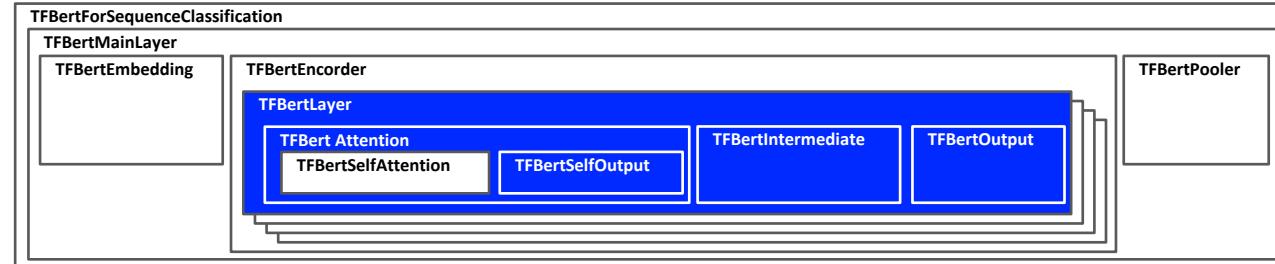
TFBertEncoder

Self Attentionを含むTFBertLayerを12段stackしている。

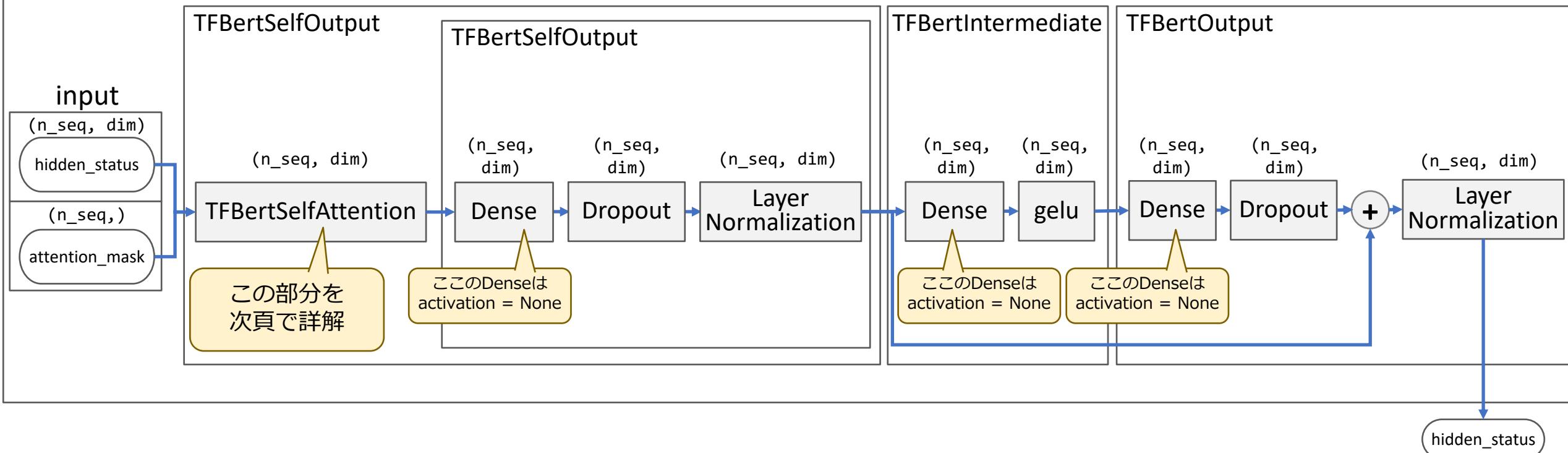


TFBertLayer

Self Attentionの出力をDense、Dropout、
LayerNormalizationなどをかけて変換。

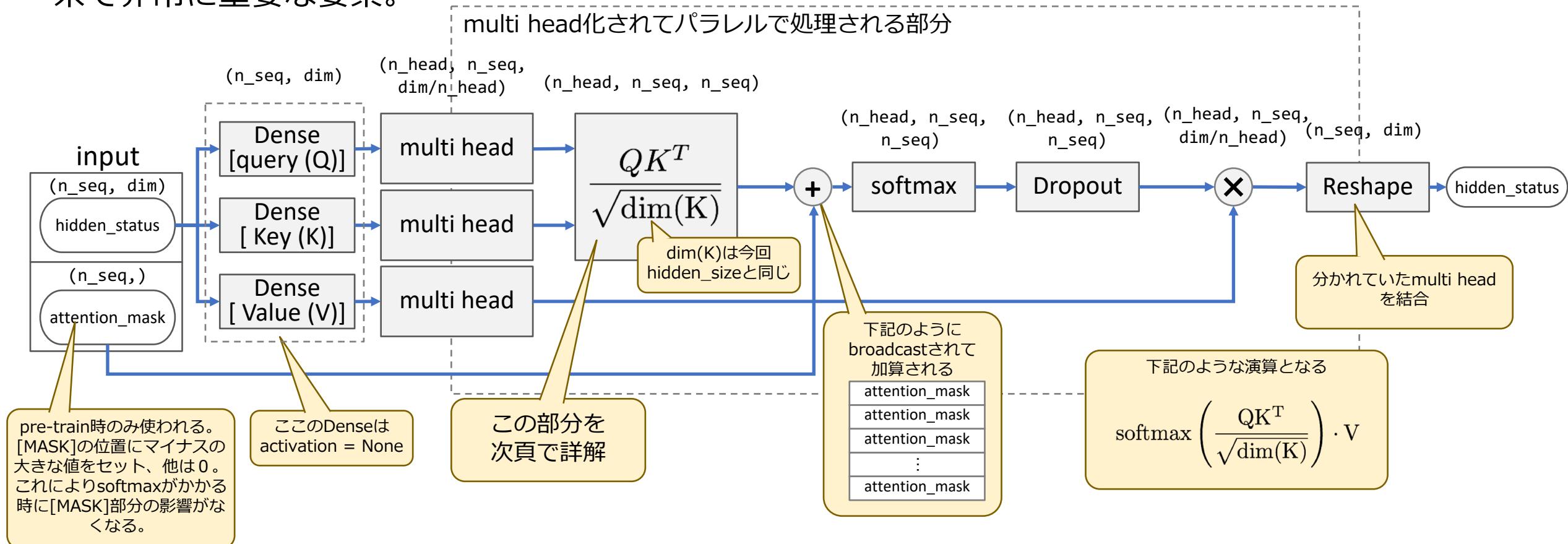
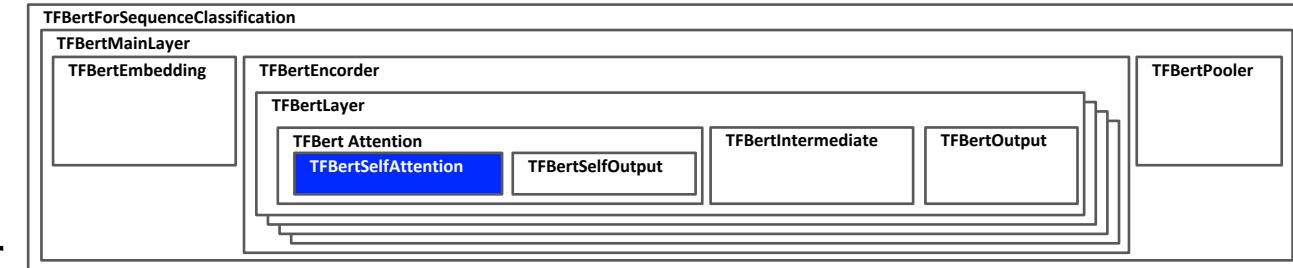


TFBertLayer



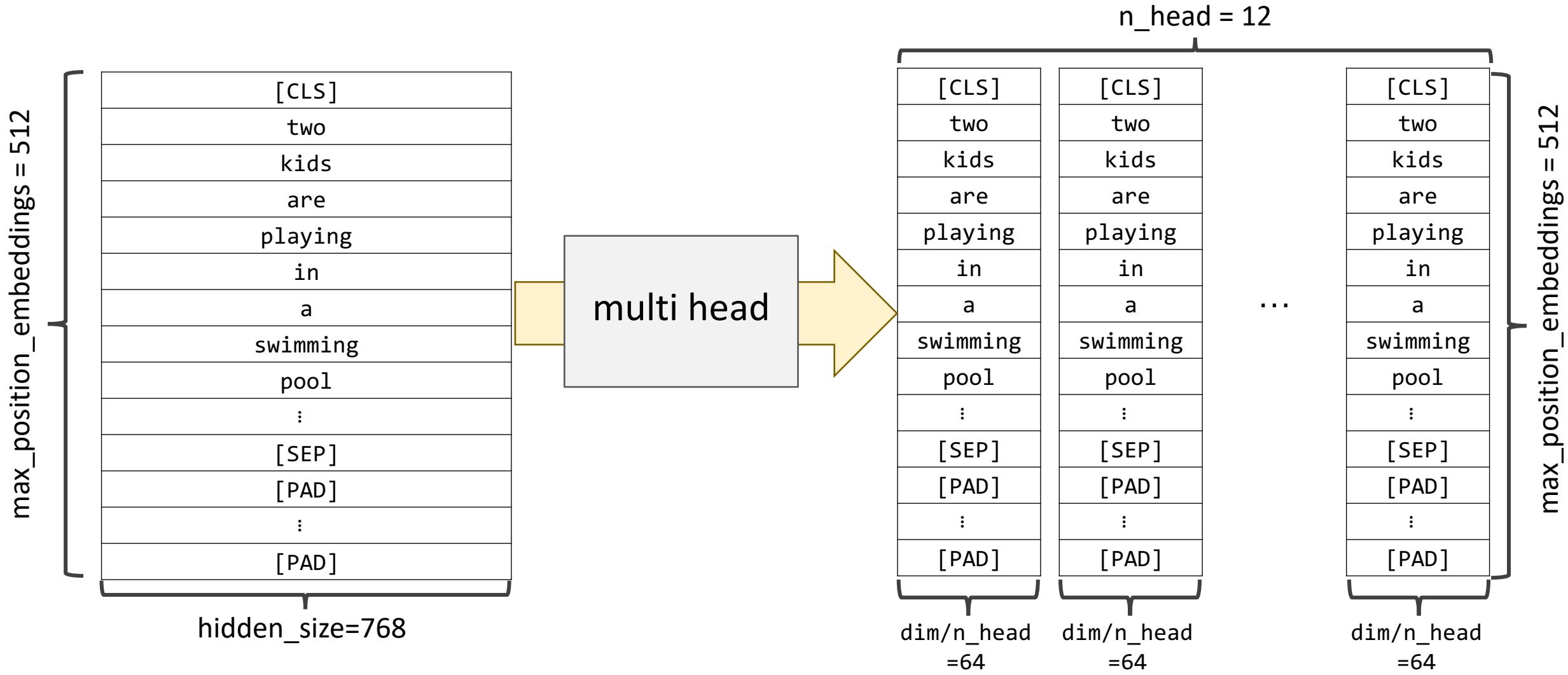
SelfAttention

QとKとVが同じ文章から構成される時、そのattentionをself attentionという。Transformer系で非常に重要な要素。



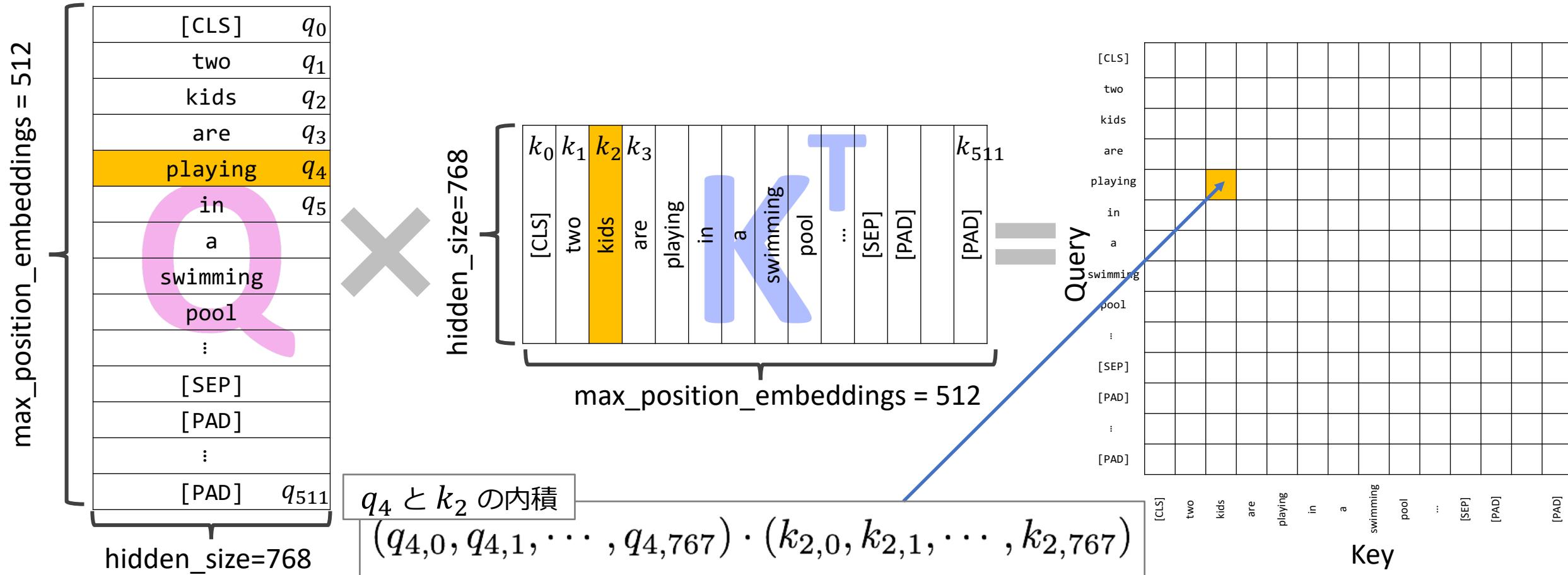
Multi head

`n_head = 12`の場合には下記のように分割され、パラレルに処理される



Self Attentionの仕組み (1)

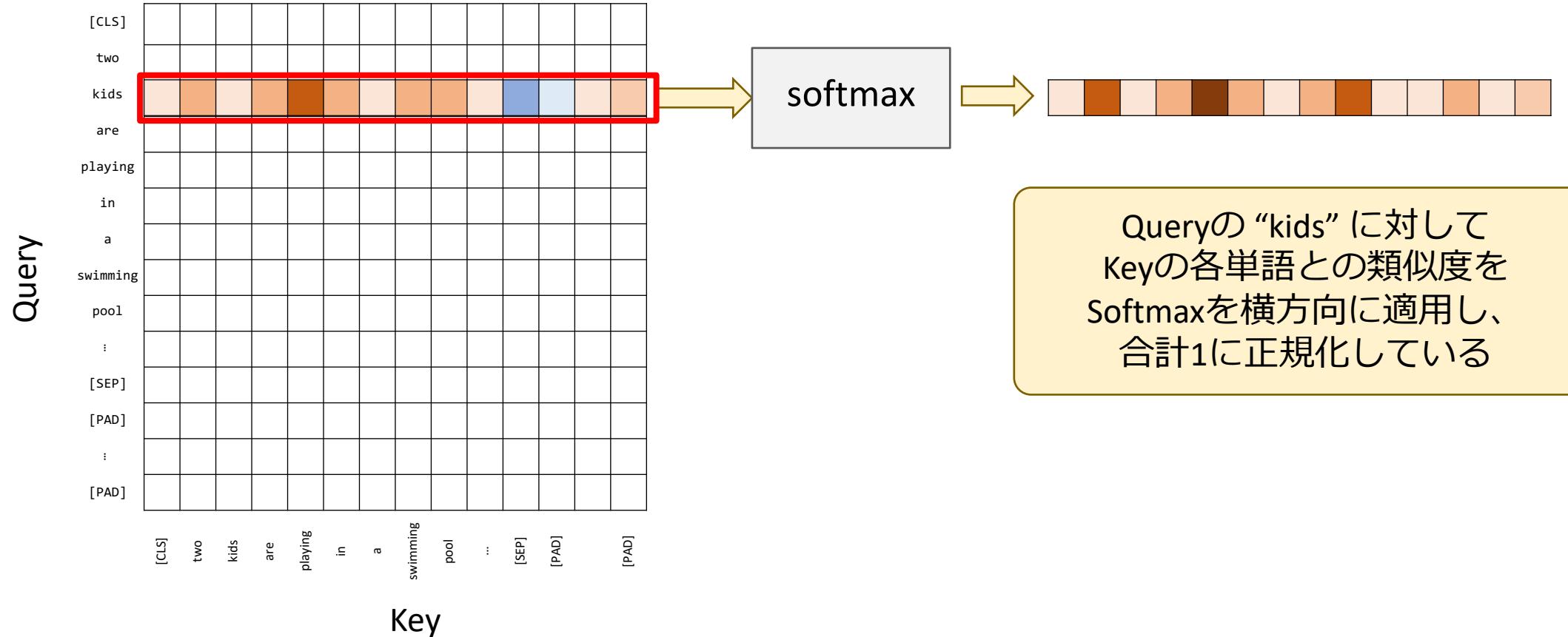
$\frac{QK^T}{\sqrt{\dim(K)}}$ の分子の部分はQの単語それぞれと、Kの単語それぞれの内積（つまり類似度）と考えられる※。



※ 上記では単語のembeddingの類似度のように記載しているが、実際はDense(linear)をかけてアフィン変換されていることに注意
 ※ multi head化で分割されてしまっているが、わかりやすさのためmulti headを省略した

Self Attentionの仕組み（2）

softmaxを横方向に適用し、加重平均のweightとして利用できるように変換。

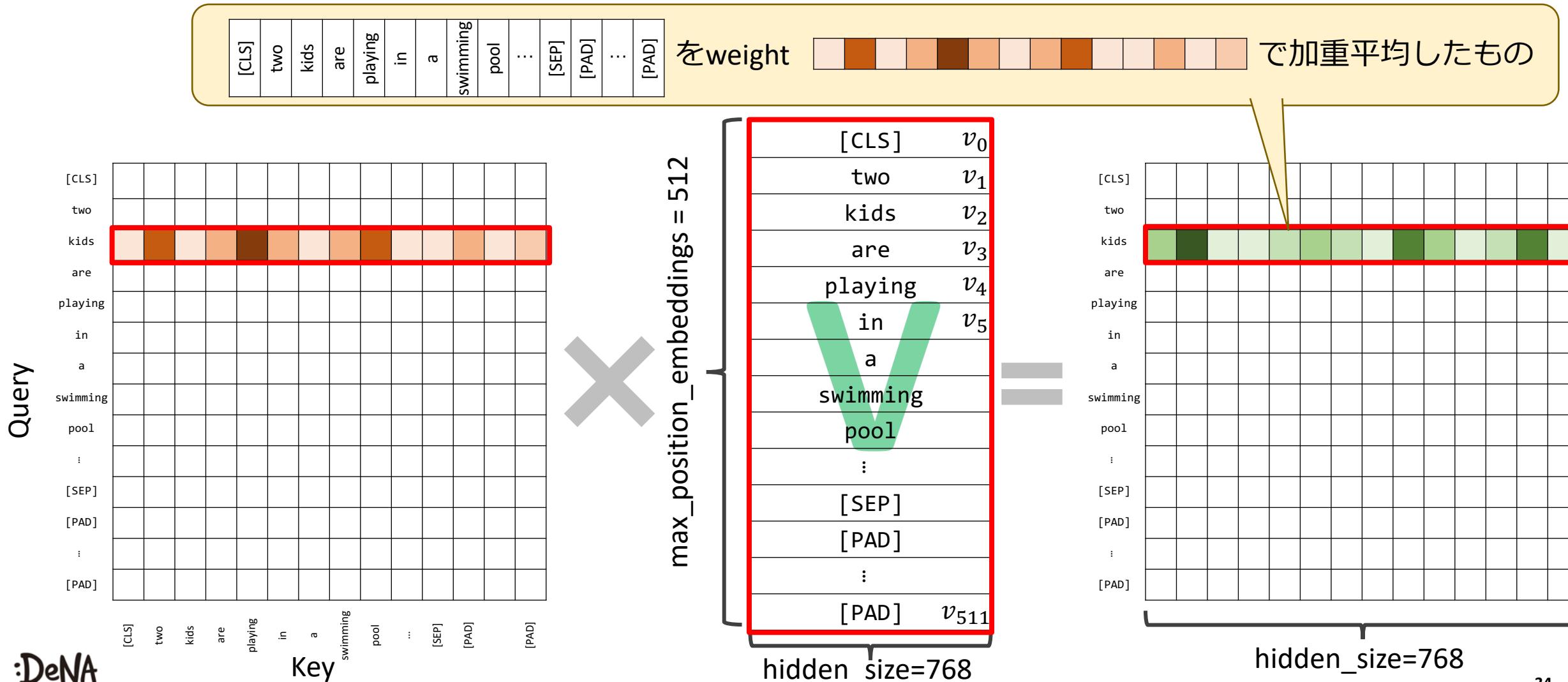


※ 上記では単語のembeddingの類似度のように記載しているが、実際はDense(linear)をかけてアフィン変換されていることに注意

※ 実際は $\text{sqrt}(\dim(K))$ で値を割っているが上記図では省略した。ある値が飛び抜けて高くならないようになまらす効果がある。

Self Attentionの仕組み (3)

Qのそれぞれの単語ごとにできたweightをValue(V)にかけて、1つの文章のembeddingに対する、加重平均を作る。



※ 上記では単語のembeddingの類似度のように記載しているが、実際はDense(linear)をかけてアフィン変換されていることに注意

Reference

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

<https://arxiv.org/abs/1810.04805>

Transformers

<https://huggingface.co/transformers>

3rd party含めた pre-trainedモデルリスト (日本語もあります)

<https://huggingface.co/models>

EOF