自然言語処理 —Attention, Transformer —

https://satoyoshiharu.github.io/nlp/

Attention、Transformerの位置づけ

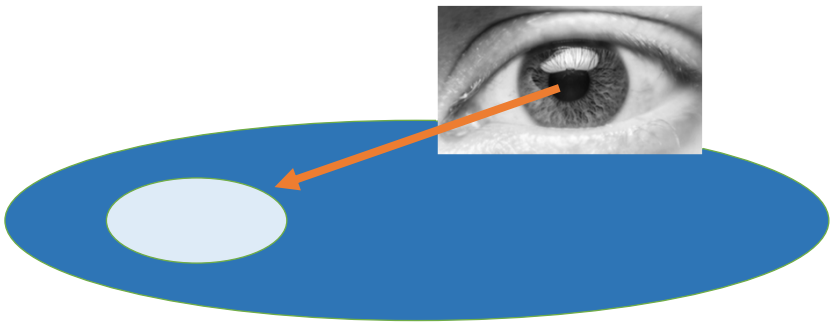
- 1000本ノック第9章では、最後の一問のみTransformerネタです。
- Attentionは、自然言語処理のみならず画像処理、信号処理など他分野へ影響を及ぼしている。
- Transformerは、Attentionをもとにしている。そして、自然言語テキストの処理は、Transformerを使った巨大なシステムが続々と生まれている。

自然言語処理
Attention

解説動画

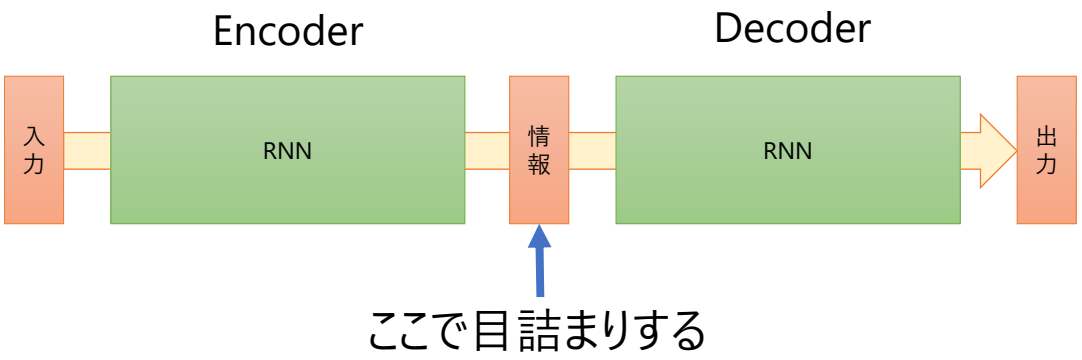
ここでは、Transformerの基礎となっているAttentionについてみていきます。

Attentionとは、



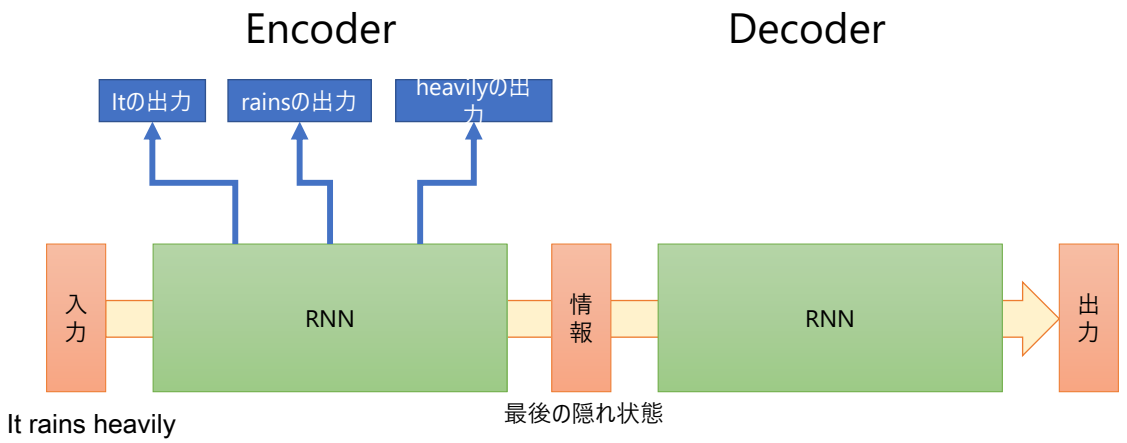
Attentionというのは、ある情報全体のどの部分に注目すれば、いい結果が得られるかを学習させて、推論に利用するアプローチです。過去のいろいろなテクニックもこれだったなと思われるくらい、汎用的で、応用範囲が広く、実装形態も多様です。
Attentionのパワーが認識されたきっかけが、自然言語処理の機械翻訳タスクなので、そこでの使われ方を見ていきます。RNNでの話なので、少し古い説明となりますが、
出てきた背景がわかりやすいので、RNNで説明します。

Seq2Seq：RNNへの入力が長くなると、



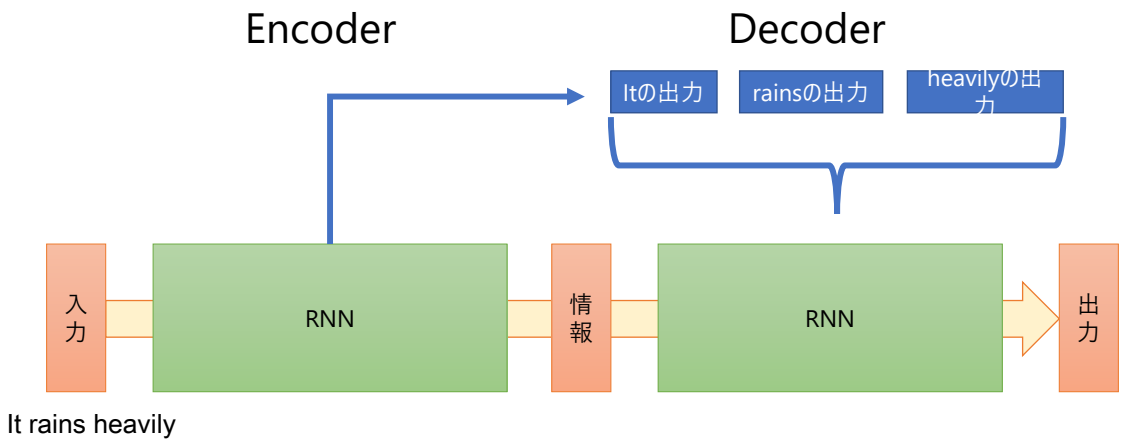
RNNのところで、Seq 2 Seq、Encoder-Decoderアーキテクチャを見ました。
今、英文を日本語文に翻訳しようとしているとします。
そこでは、Encoderの出力は最後の隠れベクトルで、保持できる情報が限られるという欠点があります。

エンコーダーの各時刻の出力も利用したい

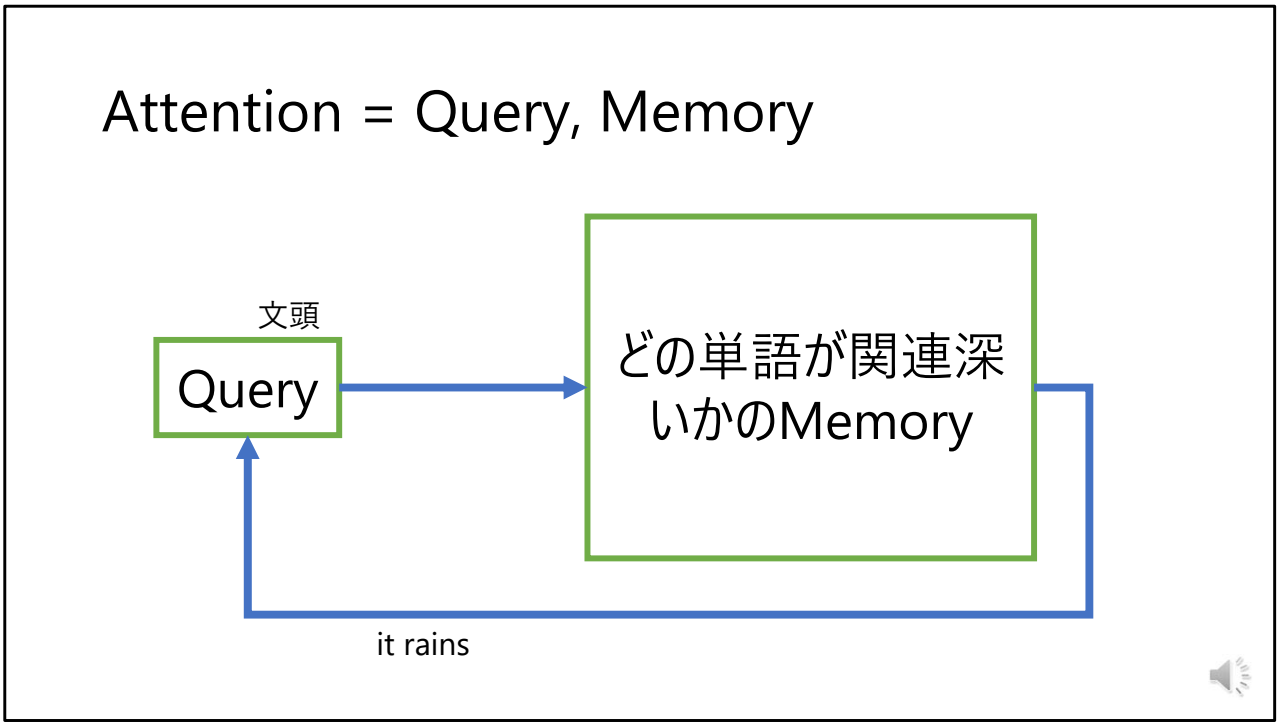


そこで、Encoderの最後の隠れ状態だけでなく、途中の各時刻の出力も見たいとなります。

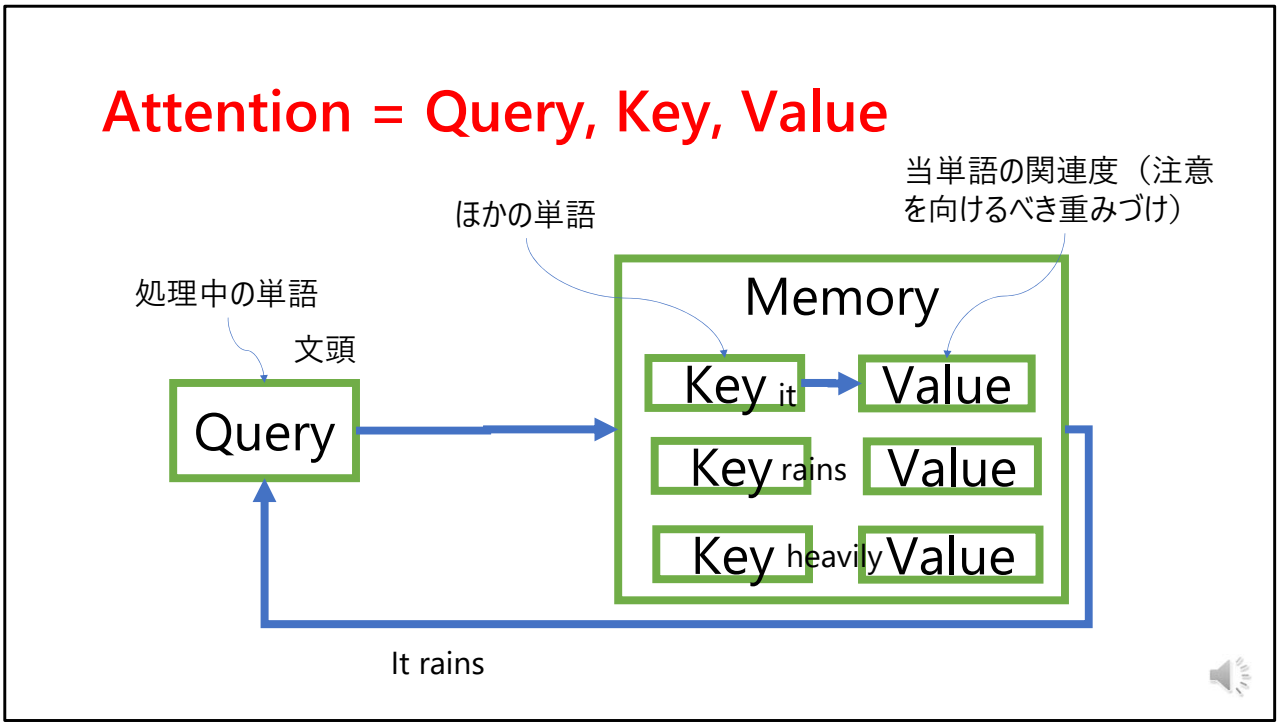
エンコーダーの出力は、1個づつ利用するのでなく、全部グローバルにみる



Decoderの処理の過程で、Encoderの出力を1個づつ見る必要はありません。
すでに全情報がそろっているので、情報全体を利用します。

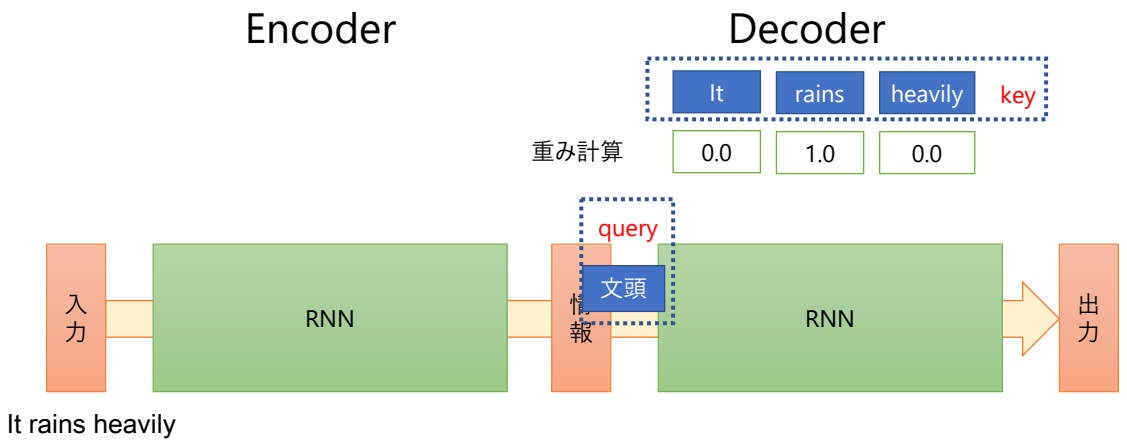


ここで、Attentionは、Query,Key,Valueからなるとして整理されることがあるので、
その観点を見ておきます。
Attentionは、QueryとMemoryからなります。
Queryは、今処理している単語です。
Memoryは、今処理している単語と関連性が深いほかの単語を記憶しています。
そのmemoryに問い合わせて、関連深い単語を得ます。



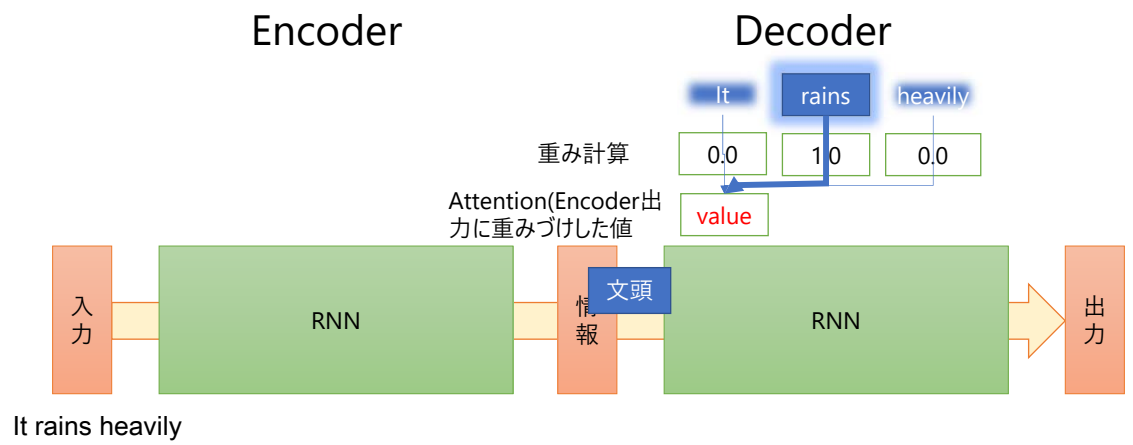
Memoryは、さらに、KeyとValueとに機能的に分割できます。
処理対象の単語をQueryとします。
Keyは、文中のほかの単語全部です。
翻訳のようにクロスで見る場合は、ターゲットのある処理単語がQueryで、
ソースの文の全単語がそれぞれKeyとなります。
要約のようにセルフで見る場合は、入力文のある処理単語がQueryで、
ほかの全単語がそれぞれKeyとなります。
Queryで問い合わせ、Keyで検索し、得られる情報が、記憶させていたValue
です。

エンコーダーの出力全体のうち、特定部分に注目する



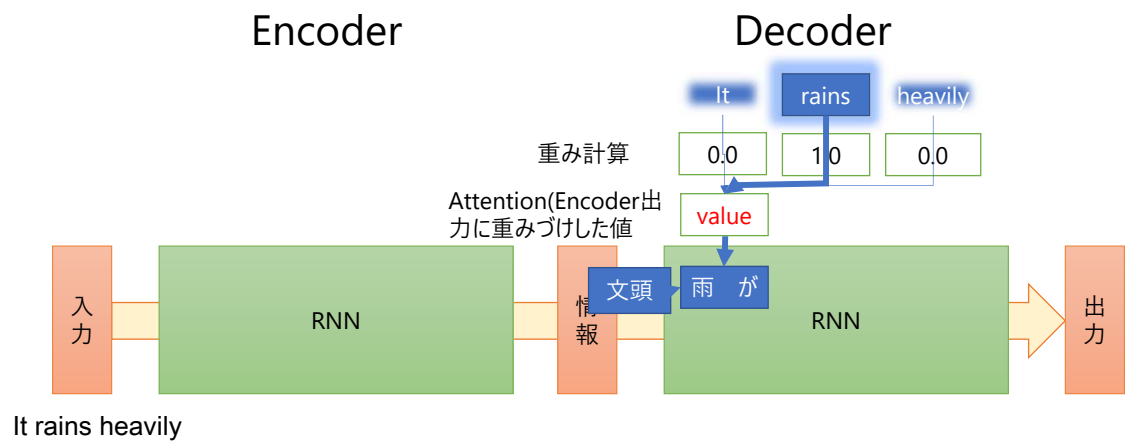
機械翻訳タスクの中でQKVの役割を見ていきます。
Encoderで入力文の情報を集約しました。
Decoderには、まず文頭というマークを入力します。
文頭マークをQueryとして、次単語を決めるため、
ソース文のEncoder出力のどこが一番ヒントになるかを問い合わせます。
ソース文のEncoderの各時刻の出力がそれぞれKeyとなりますが、
エンコーダーの各時刻の出力はほぼそこで処理した単語に対応します。

エンコーダーの出力全体のうち、特定部分に注目する



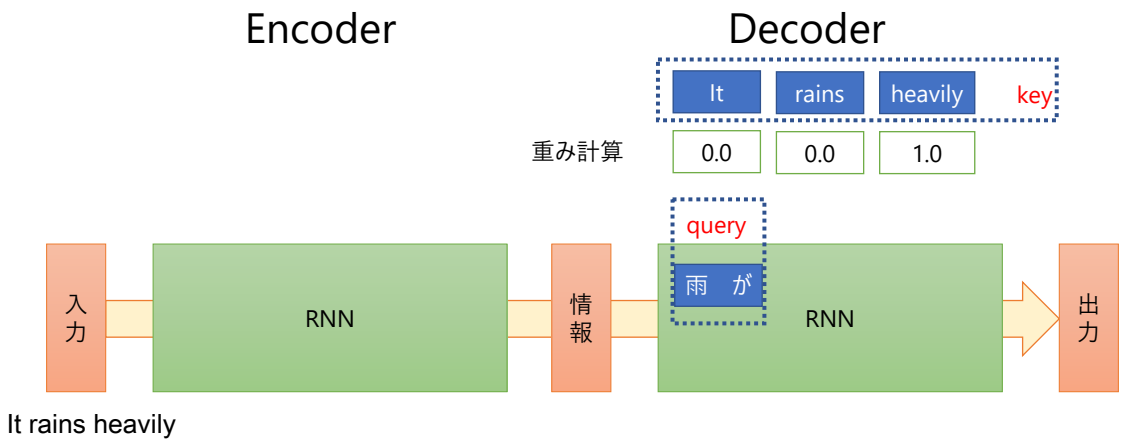
QueryとKeyの組み合わせで、問い合わせた結果、エンコーダー出力のそれぞれの重要度の重みづけが得られます。
重みづけを得たら、エンコーダー出力のそれぞれの重要度を示すValueを計算します。
ここでの重み計算やValue計算は、デコーダーの隠れ状態を加味した全結合などで計算するとします。

エンコーダーの出力全体のうち、特定部分に注目する



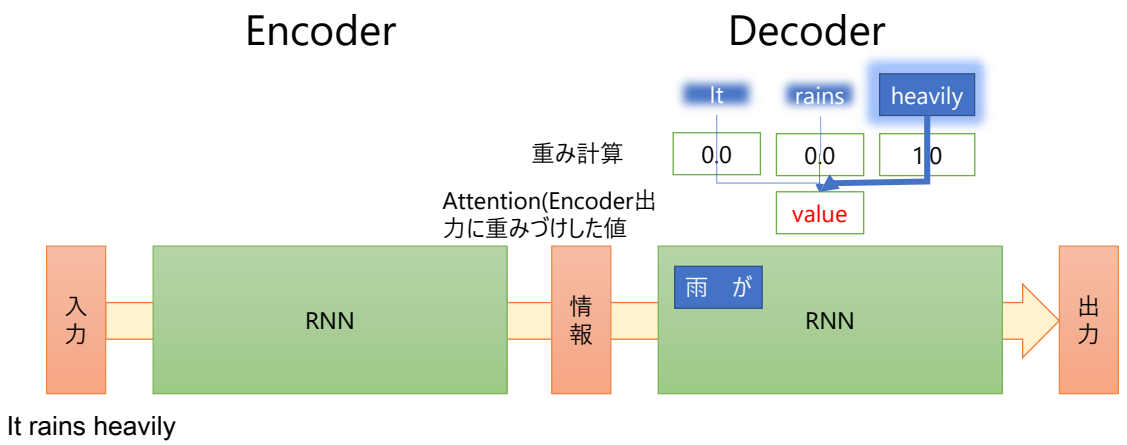
デコーダーは、Valueというヒントを得た結果、適切な単語を選びます。

エンコーダーの出力全体のうち、特定部分に注目する



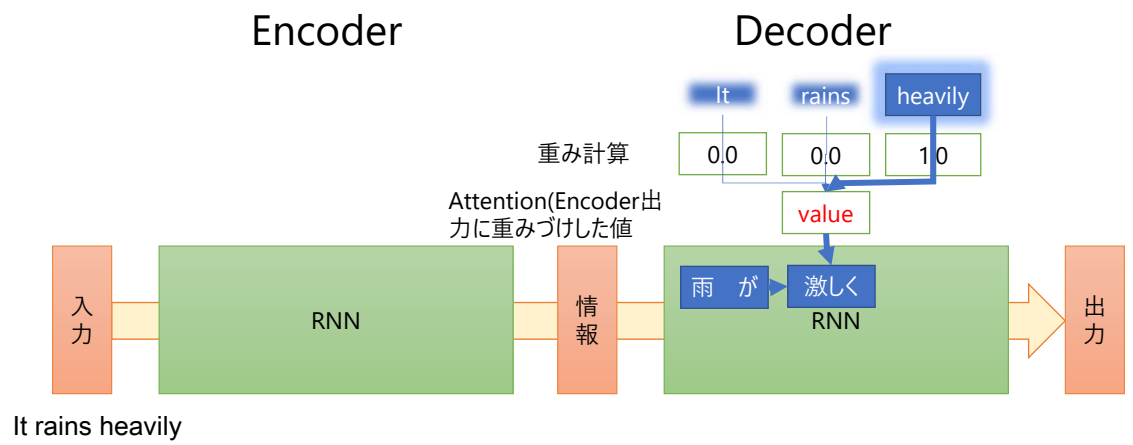
今度は、今選んだ単語をQueryとして、デコーダーの隠れ状態とエンコーダー出力であるKeyと組み合わせて、
重みづけを得ます。

エンコーダーの出力全体のうち、特定部分に注目する



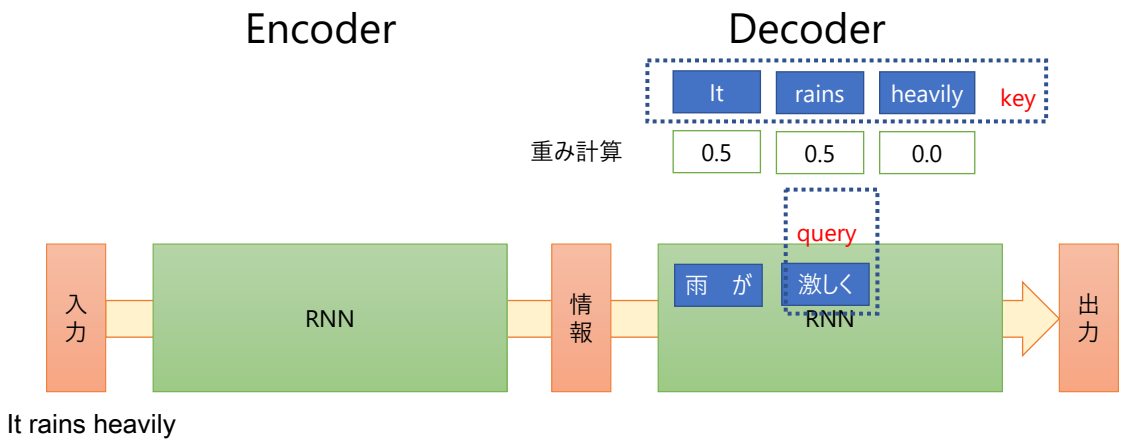
重みづけに基づいて、Valueを得ます。

エンコーダーの出力全体のうち、特定部分に注目する



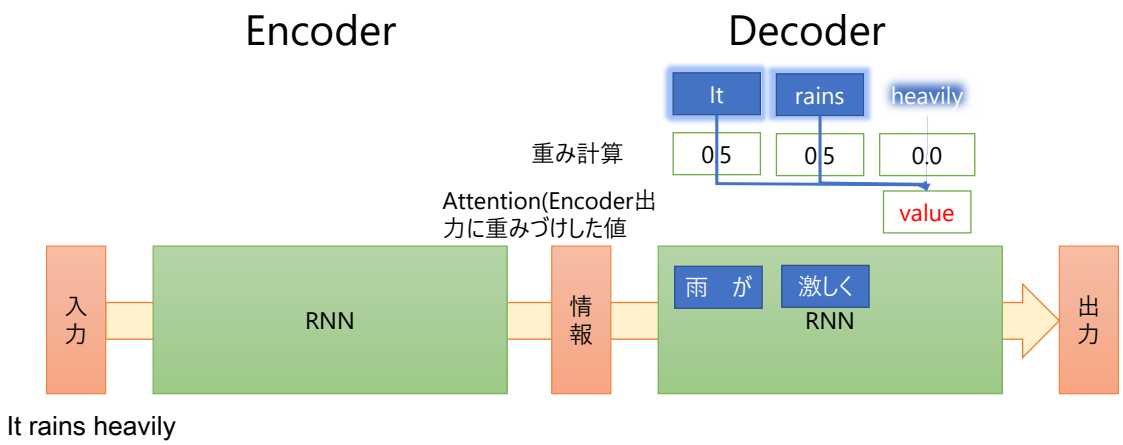
Valueに基づいて、次の単語を決めます。

エンコーダーの出力全体のうち、特定部分に注目する

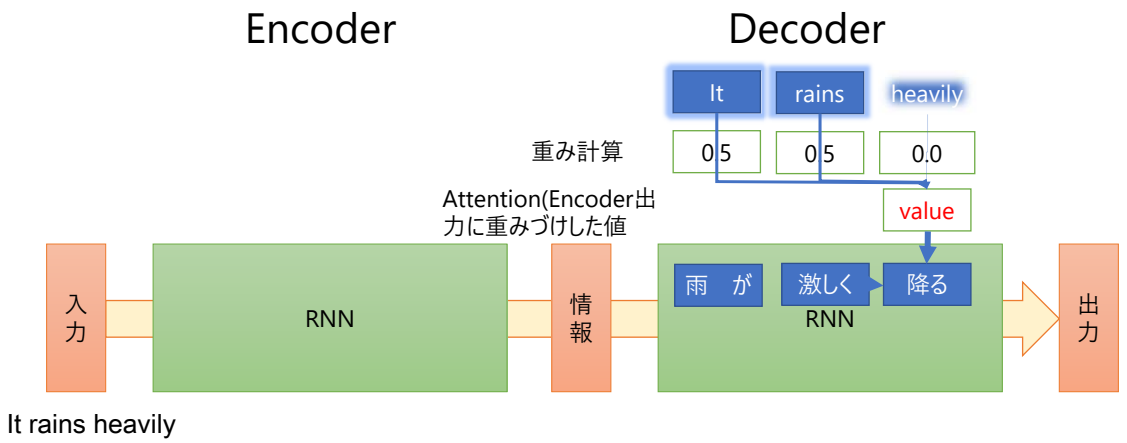


このように、デコーダは、次の単語を決める際に、Queryして、
ソース文の全体を見渡すためにKeyを取り出し、全体の中でどこに注目したらいいかを示す
Valueを計算し、それをもとに次単語を決めていきます。
Queryはターゲット言語内埋め込み表現です。
Keyは、エンコーダーの各時刻の出力で、入力文の各単語に対応します。
重みやValueの計算は、デコーダーの隠れ状態とQ,Kを利用した全結合などの学習で獲得されます。

エンコーダーの出力全体のうち、特定部分に注目する



エンコーダーの出力全体のうち、特定部分に注目する



Attentionの学習

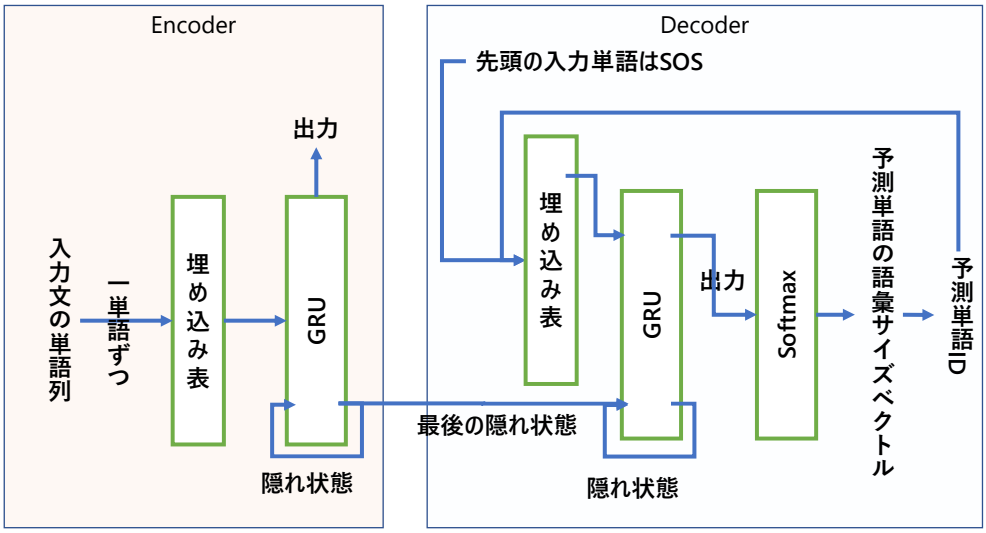
- Attentionは、ネットワーク構成中に、あるところに注目する仕組みを入れるというアプローチ。
- あるところに注目した結果、あるロスになった。それを踏まえて、逆伝播学習の中で、よりよい結果が得られるように注目個所を繰り返し最適化する。

このように、Attentionは、Deep Learningの学習の仕組み中に、あるところに注目する、という仕組み（計算と結合線）を入れることで実現します。
あるところに注目した結果、結局、あるロスになった。それを踏まえて、逆伝播学習の中で、よりよい結果が得られるように注目個所を繰り返し最適化することで、獲得されます。

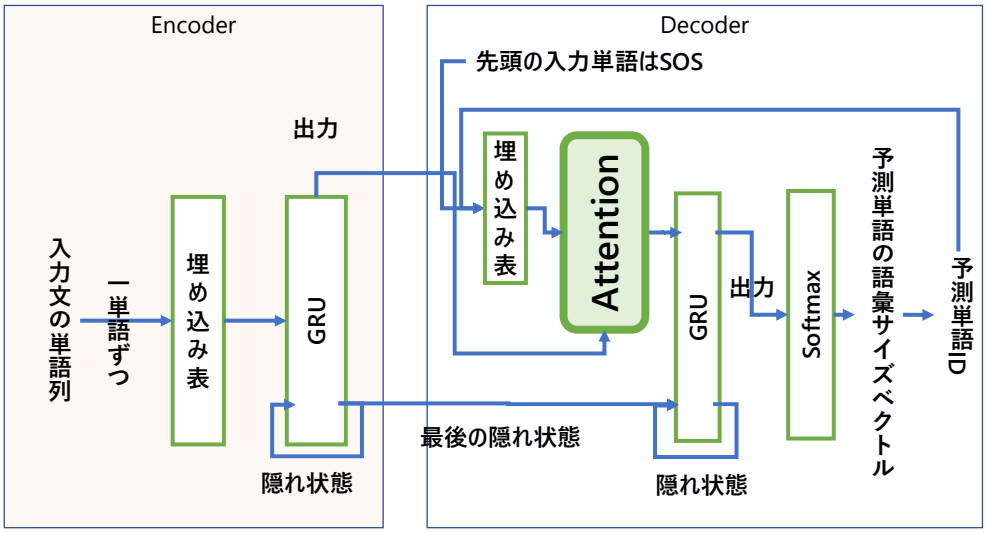
Attention 課題 1

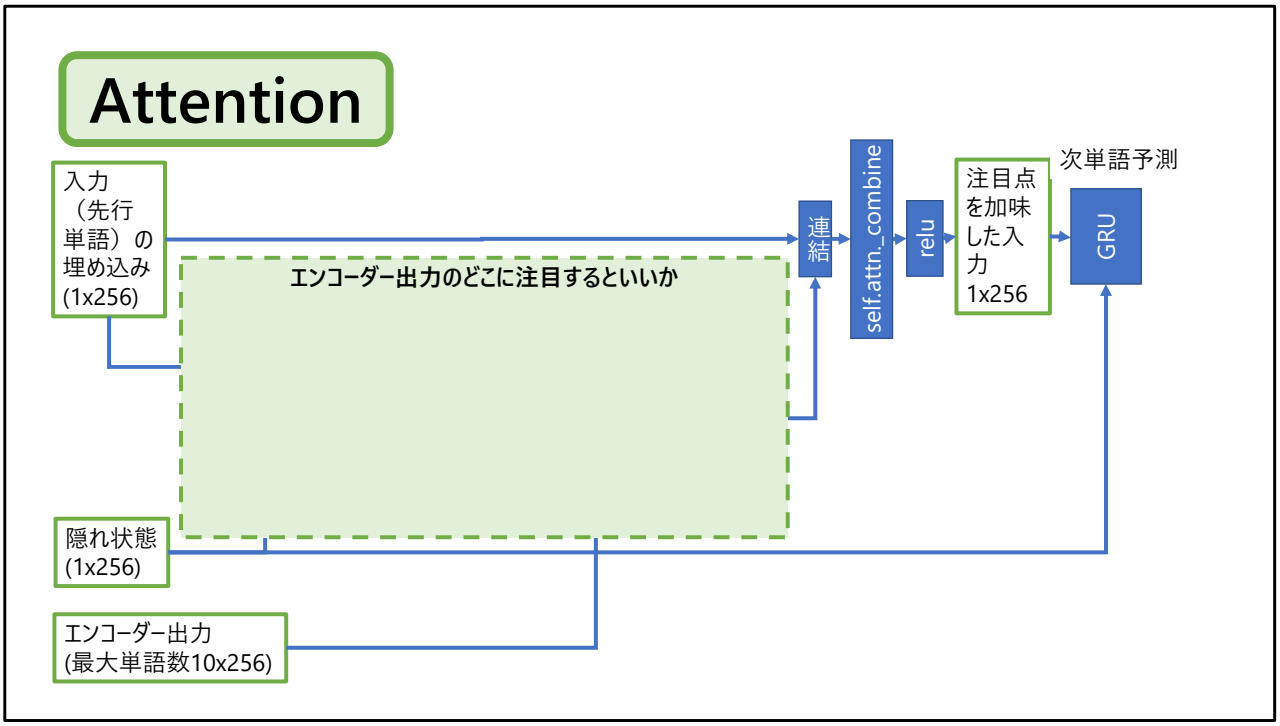
- attention_translation.ipynb があります。rnn_translation の Attention込み版です。これを読解します。

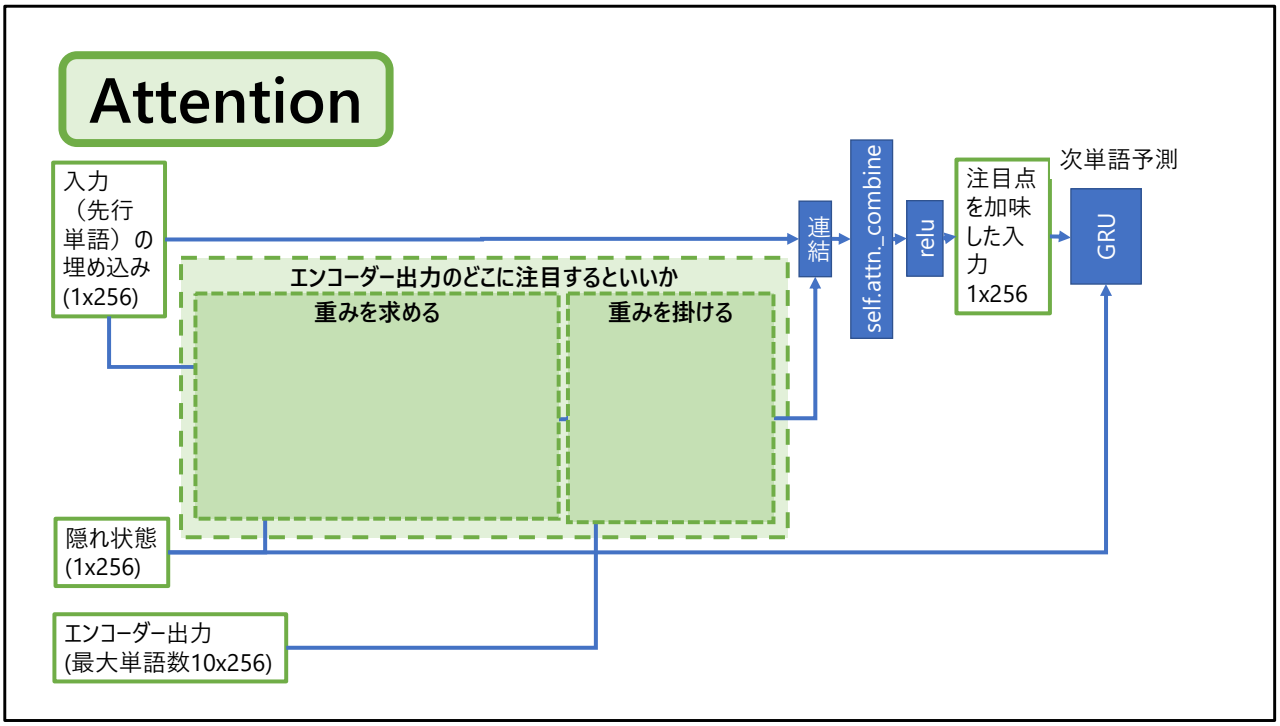
Attention入れる前のネットワーク構成

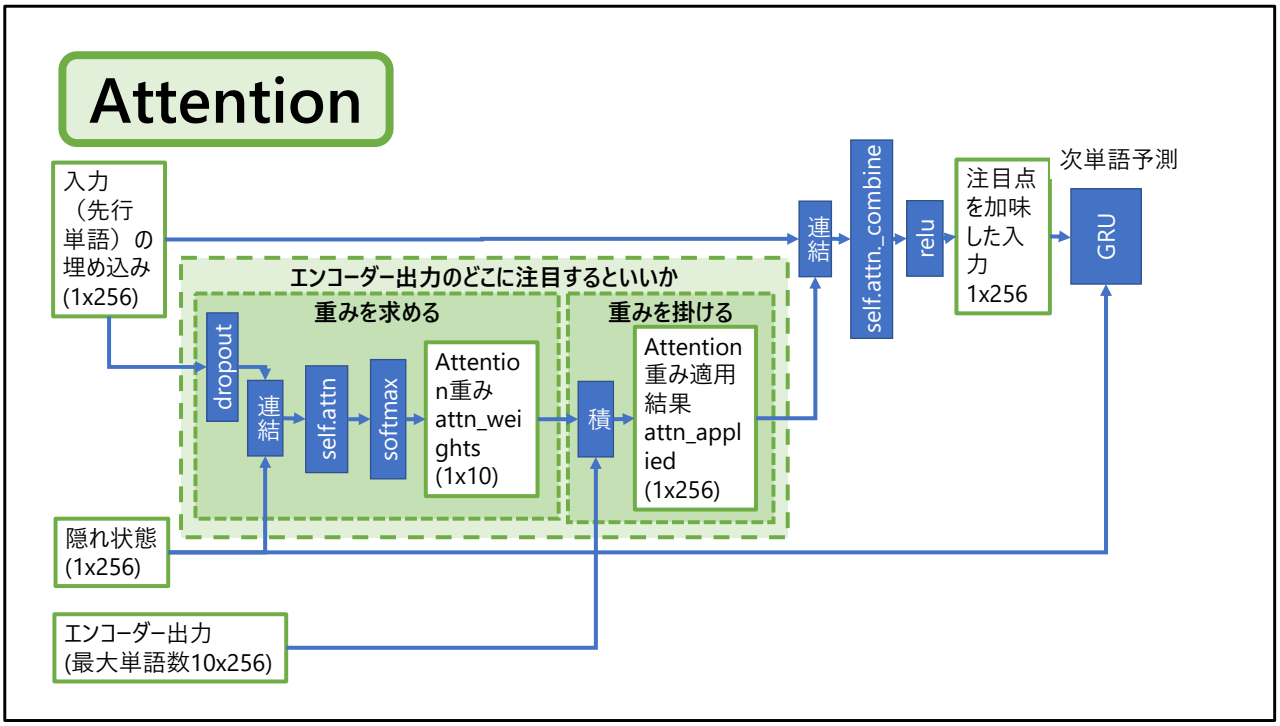


Attention組み込んだ後のネットワーク構成




Attention
入力
（先行
単語）の
埋め込み
(1x256)
エンコーダー出力のどこに注目するといいか
連結
self.attn_combine
relu
注目点
を加味
した入
力
1x256
GRU
次単語予測
隠れ状態
(1x256)
エンコーダー出力
(最大単語数10x256)


Attention
入力
（先行
単語）の
埋め込み
(1x256)
エンコーダー出力のどこに注目するといいか
重みを求める
重みを掛ける
隠れ状態
(1x256)
エンコーダー出力
(最大単語数10x256)
連結
self.attn_combine
relu
注目点
を加味
した入
力
1x256
GRU
次単語予測


Attention
入力
（先行
単語）の
埋め込み
(1x256)
エンコーダー出力のどこに注目するといいか
重みを求める
dropout
連結
self.attn
softmax
Attentio
n重み
attn_wei
ghts
(1x10)
重みを掛ける
積
Attention
重み適用
結果
attn_appl
ied
(1x256)
連結
self.attn_combine
relu
注目点
を加味
した入
力
1x256
GRU
次単語予測
隠れ状態
(1x256)
エンコーダー出力
(最大単語数10x256)

参考資料

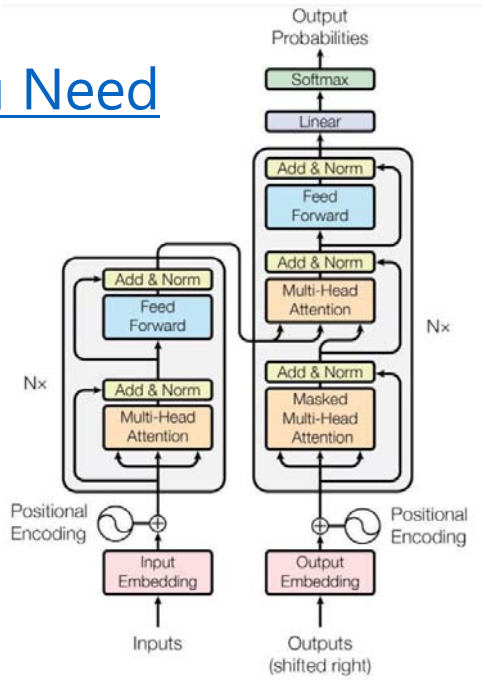
- A Brief Overview of Attention Mechanism
- チュートリアル動画：Deep Learning入門：Attention（注意）、Sony 小林さん
- An introduction to Attention

自然言語処理
Transformer

解説動画

ここでは、BERTあるいはその中核をなすTransformerについてみていきます。

Attention Is All You Need



BERTは、Googleが、2017年から18年にかけて発表したもので、
自然言語用のBERTネットワークをプリトレーニングしたものが、それまでの様々な記録を塗り替えました。
その中心となるネットワーク構成はTransformerと呼ばれます。

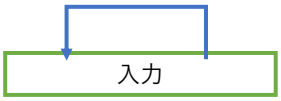
CNNは、周りを見るとはいえ限定的。RNNは、逐次にデータを見ていくので並列処理ができない。
そこで、CNNもRNNも全く使わないで、全体をグローバルにみ、並列処理も可能なものとして、Transformerが出てきました。

BERTあるいはTransformerは、とても複雑なので、最初から実装の詳細を見てしまうと
全体がよく見えません。
ここでは、実装から入るのではなく、
パーツの意味、機能・役割を押さえることによって、全体を理解するというアプローチをとります。
実装はオリジナル論文や論文に沿った解説記事をごらんください。

まず、前提として、Attentionと言語モデルに関する4つの概念を押さえます。

そのあとで、Self-AttentionというBERTの中心的なパーツを見ます。
そして、BERT全体が、パーツからどう構成されているかを見ることで、全体を把握します。

Self-Attention



「鳥がナク」

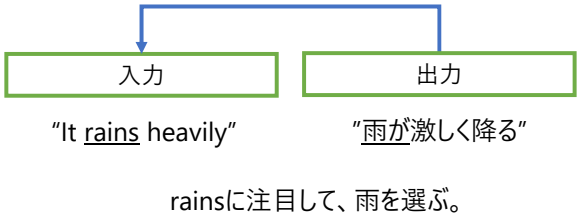
「ナク」は鳥に注目すれば、
「泣く」でなくて「鳴く」。

ある入力文の中で、ある単語とほかの単語の関連度

まず、前提として、AttentionとLanguage Modelに関する4つの概念を押さえます。
4つの概念は、2つづつの対立概念となっています。

最初の対立概念を見ます。
Attentionは、大きく、Self-Attentionとクロスなものに分けられます。
Self-Attentionというのは、注目する箇所を選ぶ対象全体と、注目箇所とが、同一データであるものです。
一つのデータの中で、ある特定の処理単位を見るときに、そのデータ全体の中でほかのどこに注目するか、を問います。
ある入力文の中で、ある単語とほかの単語の関連度を学習します。

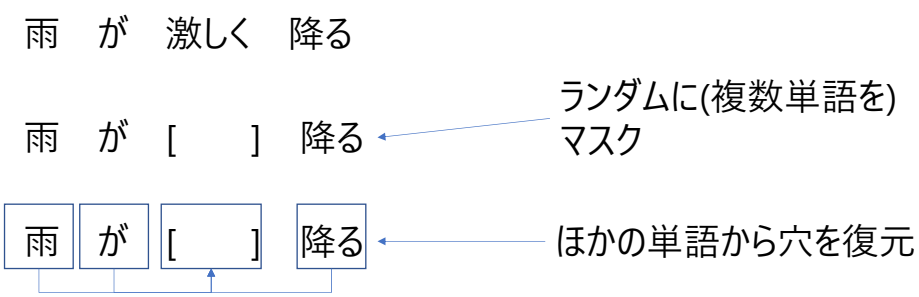
Source-Target Attention



ある入力文と変換後の出力文の間で、ある出力単語と入力単語の関連度

Source-Target　Attentionというのは、注目する箇所を選ぶ対象全体と、注目箇所とが、別一データからきているものです。
二つのデータの中で、一方のある特定の処理単位を見るときに、他方のデータ全体の中のどこに注目するか、を問います。
ある入力文と変換後の出力文の間で、ある出力単語と入力単語の関連度を学習します。

Masked Language Model



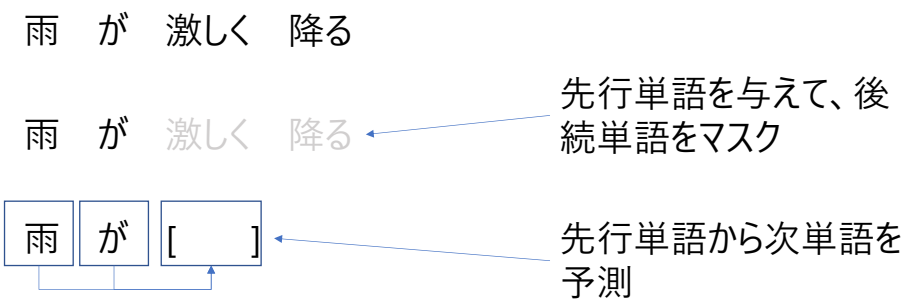
文章内の単語の相互の関連度を学習

二つ目の対立概念です。
Language Model、言語モデルというのは、言語をモデル化して、
言語に関するさまざまな予測タスクに利用できるものをいいます。

ここでは、Hugging Faceの分類用語を取り上げます。
言語モデルは、トレーニングする方法の観点から、Masked LMとCausal LMとに分かれます。

Masked LMというのは、BERTのトレーニングで採用されて効果が実証されました。
Masked LMというのは、入力文が与えられたときに、一部の単語ランダムにマスクし、
それを復元するようにトレーニングするものです。
マスクした状態を入力し、マスクされた単語を正解ラベルとして、学習します。
その学習の中で、正解を与えるために関連度が高い周辺単語が、Attentionとして学習されます。
そのトレーニングの結果、文章内の単語の相互の関連性が学習されます。

Causal Language Model



先行単語との関連度から次の単語を予測する

Causal LMというのは、BERTのデコーダーのトレーニングで採用されいるものですが、
機械翻訳や音声認識など伝統的なタスクで以前から行われてきたものです。
Causal LMというのは、入力文が与えられたときに、先頭から見て行って、先行単語をもとに、次に来る単語を予測します。
先行単語列を入力し、次単語を正解ラベルとして、学習します。
BERTのデコーダーでは、後続単語をマスクしてトレーニングするというやり方を取りました。
この学習の中で、正解を与えるために関連度が高い先行単語がAttentionとして学習されます。
結果として、単語を予測するために重要な先行単語との関連性が学習されます。

以上の4つの概念をベースに、BERTを把握します。

Transformer

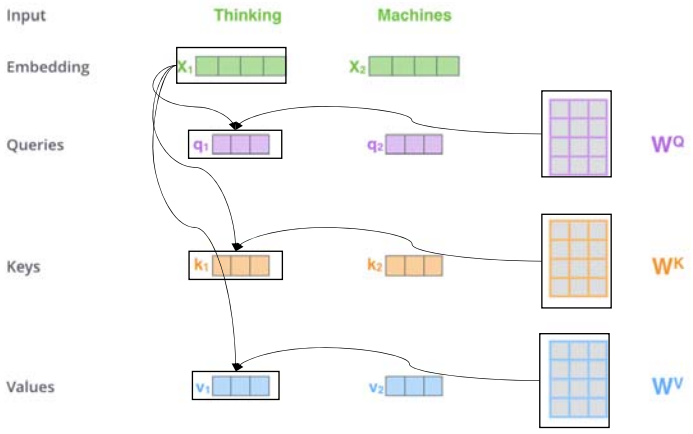
huggingfaceの定義

self-attention based deep learning model architecture

BERTの中核部分はTransformerと呼ばれますが、その定義は
Self-AttentionだとHuggingFaceではみなしています。

そこで、最初に、BERTのSelf-Attentionのパーツを見ていきます。

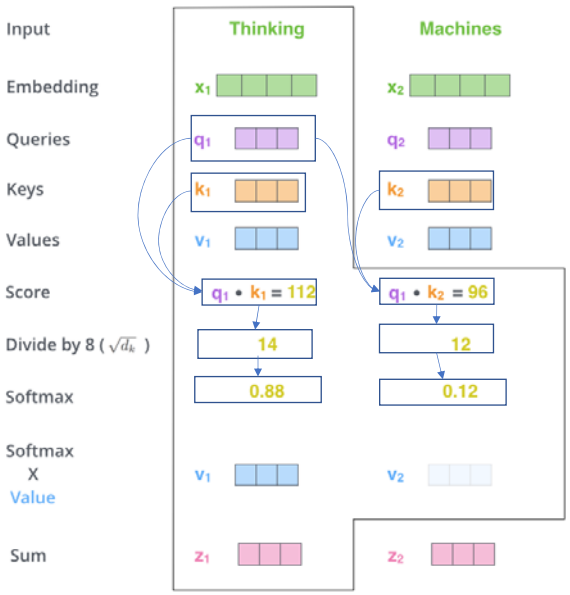
Query, Key, Value



図はThe Illustrated Transformerから

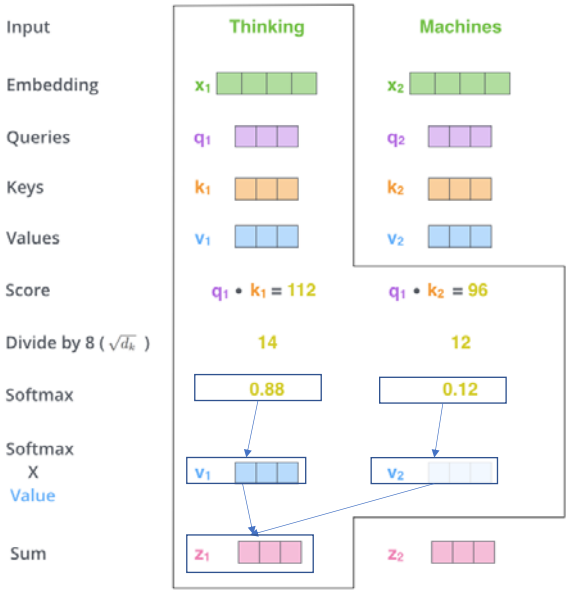
Attentionは、問い合わせQueryと相手Keyの組み合わせによって記憶内容Valueを取り出して
Queryに対応する単語と、ほかの単語の関連度を計算します。

入力となる単語埋め込みベクトルに、それぞれQuery, Key, Valueというデータを準備します。
それらは、埋め込みベクトルを重み行列Wq、Wk、Wvにかけて、
各入力単語のQueryベクトル,Keyベクトル,Valueベクトルとなります。
このWq,Wk,Wvも、逆伝播の中で学習するパラメータになります。



図はThe Illustrated Transformerから

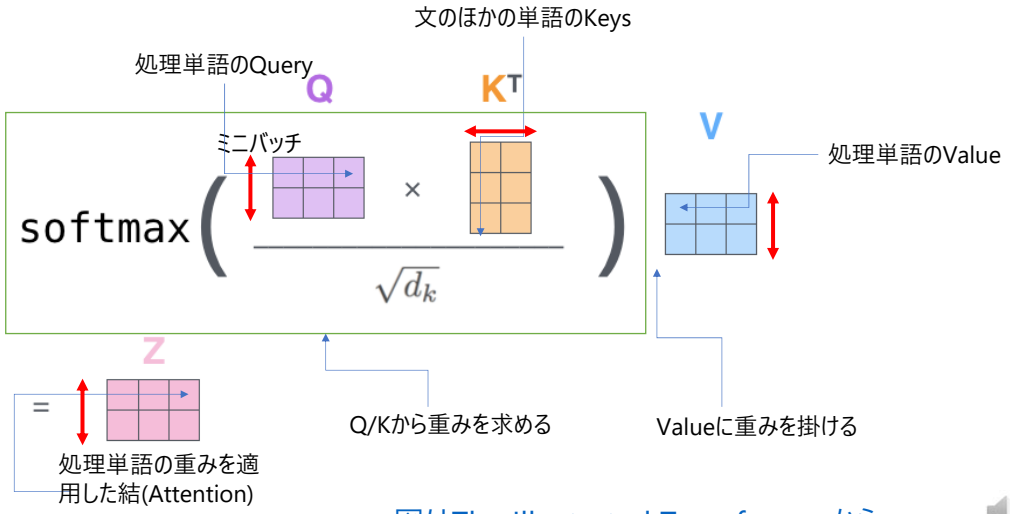
まず、重みづけの計算です。
処理単語のQueryベクトルと、入力文のほかの単語のKeyベクトルをかけた結果をScoreとします。
Scoreを正規化し、Softmaxに通し、重みづけを得ます。



図はThe Illustrated Transformerから

次にvalueベクトルをつかったAttentionの計算です。
すべての単語の重みづけとvalueベクトルとを掛けて、和を取ります。それが、現在の処理単語に関する、当入力文内の注目個所を示すAttentionベクトルとなります。

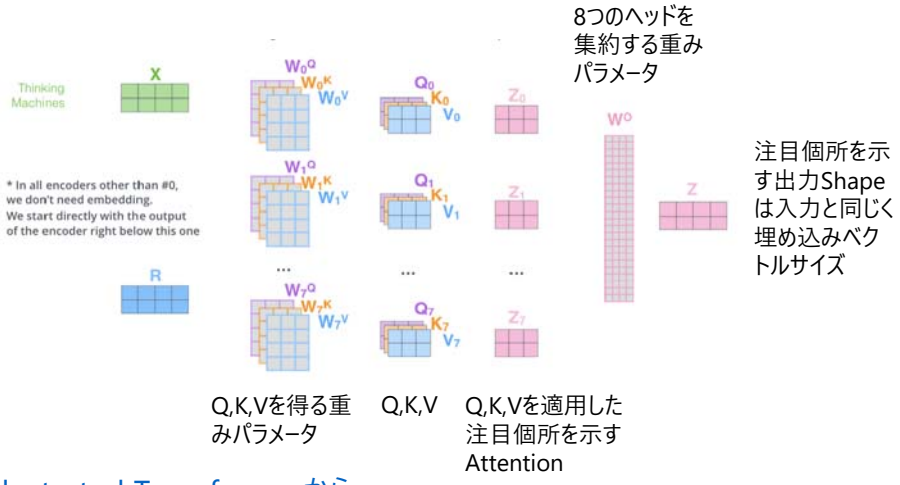
Self-Attention



図はThe Illustrated Transformerから

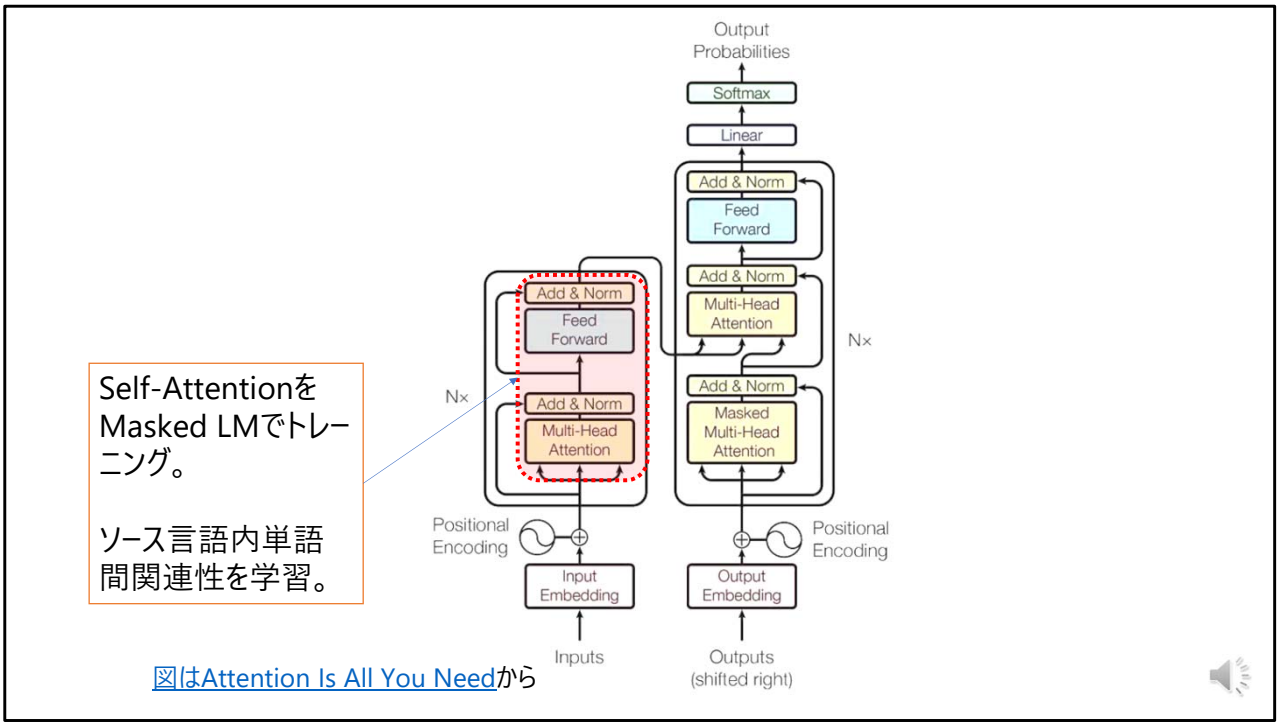
式でまとめるとこうなります。
QueryとKeyをかけて、正規化し、Softmaxをとる。
それをValueにかけて、Attentionを得る。

マルチヘッドAttentionで複数の関連性を捕捉する

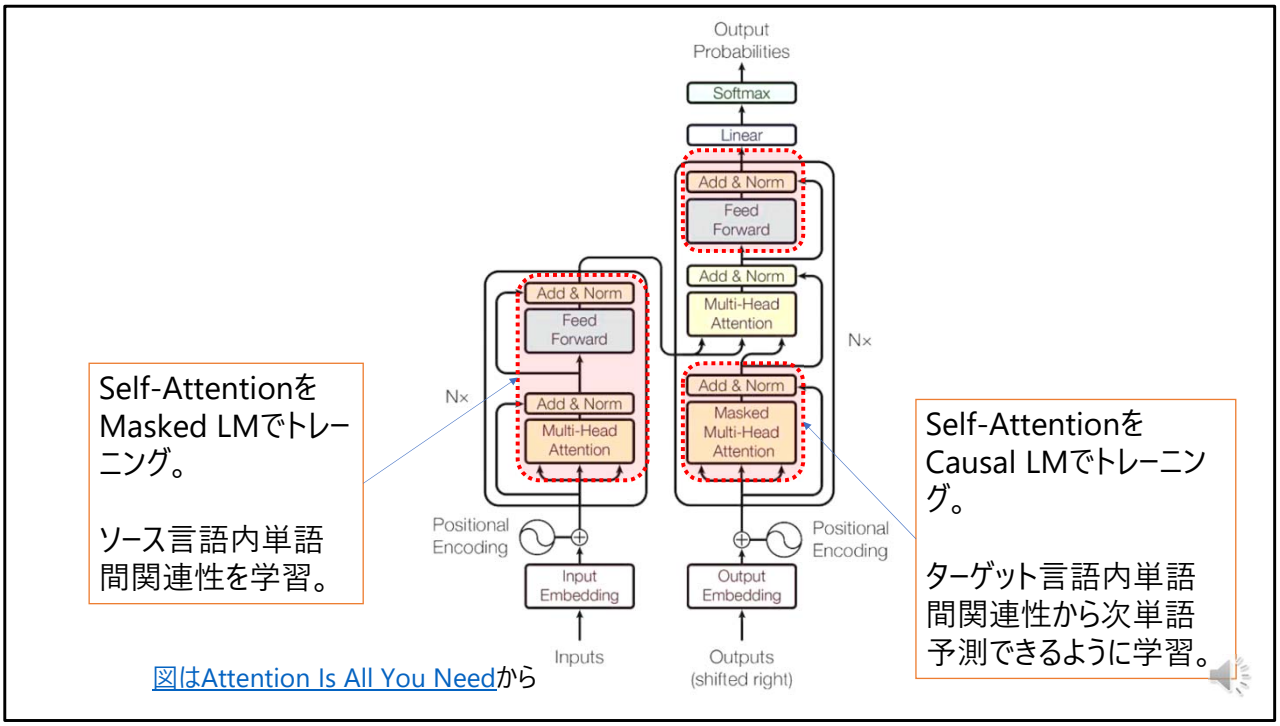


[図はThe Illustrated Transformerから]

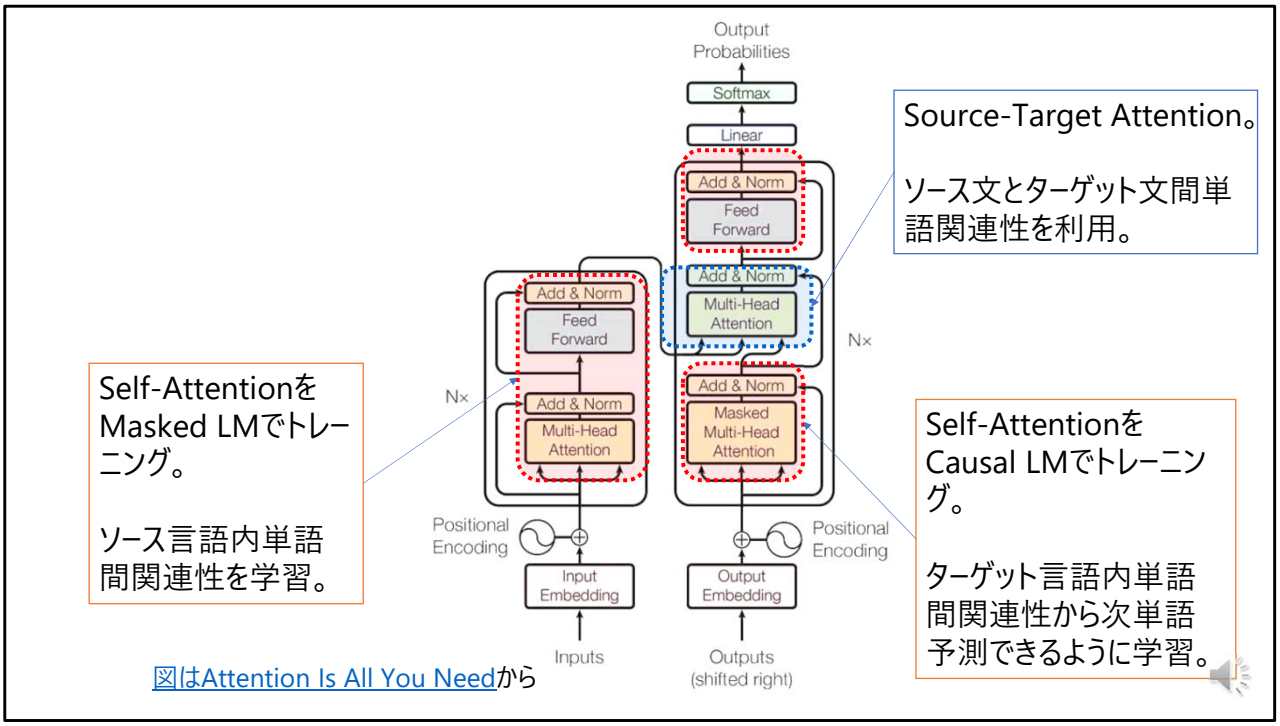
BERT/Transformerは、このようなSelf-Attentionを、複数設けて、総合し、単語間の関連度をもれなく補足します。



以上のようなマルチヘッドAttentionブロックを配置します。
BERTでは、エンコーダー、デコーダーアーキテクチャを取ります。
エンコーダー部で、入力文を、Self-Attention機構をMasked LMでトレーニングします。
Masked LMというのは、入力文の一部単語をランダムにマスクしそれを復元するように学習させます。
それによって、エンコーダーのSelf-Attentionブロックに、ソース言語内の単語間関連性が学習されます。

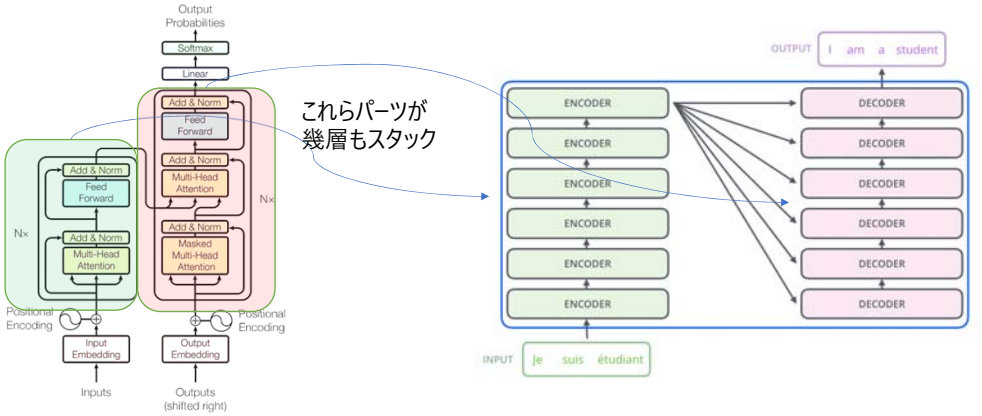


次に、デコーダー部では、ターゲット言語に関して同様にSelf-Attentionブロックを用いますが、
今度はCausal LMでトレーニングします。Causal LMというのは、後続単語をマスクし、
先行単語から次の単語を予測するようにトレーニングするものです。
それによって、ターゲット言語内の単語の関連性から次単語を予測する知識が学習されます。



デコーダーでは、さらに、ソース文内単語とターゲット文内単語との間の関連性を利用するため、
Source-Target　Attentionブロックを挟み、ターゲット文の単語の選択で、ソース文の全単語のAttention情報を利用します。

BERT全体構成：レイヤのスタック



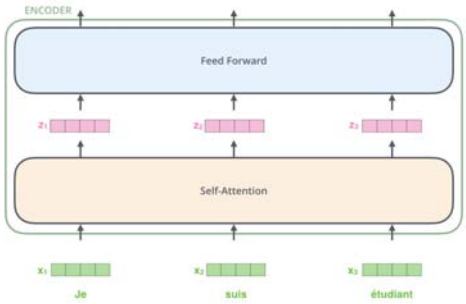
図はAttention Is All You Need から

図はThe Illustrated Transformer から

先のエンコータとデコーダの構成は、同じブロックを何層も重ねる構成をとります。
それによって、複雑な構造を補足しようとしています。

これが、BERTの重要パーツの意味・役割と、それらでくみ上げている全体構成です。

単語の埋め込みベクトルが、順次上のレイヤに

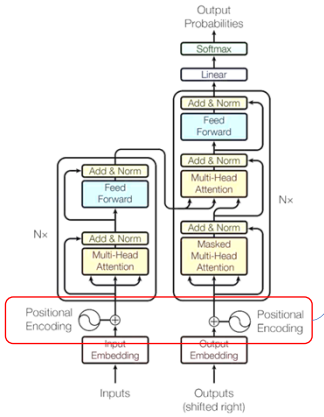


単語ごとに全結合

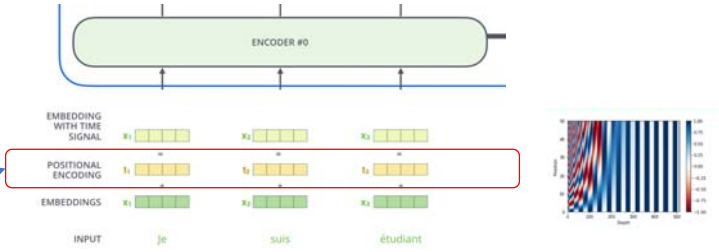
図はThe Illustrated Transformer から

入力に関し、補足します。
最下層レイヤの入力は単語の埋め込みベクトルです。
あるSelf-Attentionレイヤでは、QKVを通して、Attentionベクトルが出力されます。
そのあとPoinwise全結合と呼ばれ、単語ごとに独立した全結合で、情報選別します。
その結果が、上のブロックへの入力となります。
こうして順次、上のブロックに情報が上がっていきます。

Positional Encoding：系列情報を埋め込みベクトルに加える



図はAttention Is All You Need から

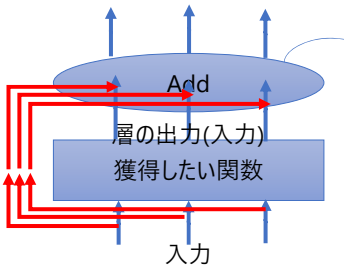


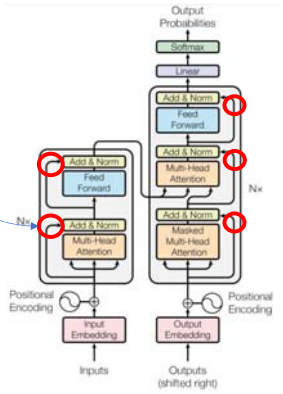
図はThe Illustrated Transformer から

Attentionは、RNNを使わないので、系列・順序情報がありません。
そこで、最下層で、単語埋め込み表現に、位置を示すあるパターンを加えることで、
順序情報を組み込みます。これをpositional encodingと呼びます。
ニューラルネットの学習は、その順序情報をちゃんと把握するそうです。

Residual Connection

層の出力(入力) = 獲得したい関数(入力) – 入力
⇒
獲得したい関数(入力) = 層の出力(入力) + 入力





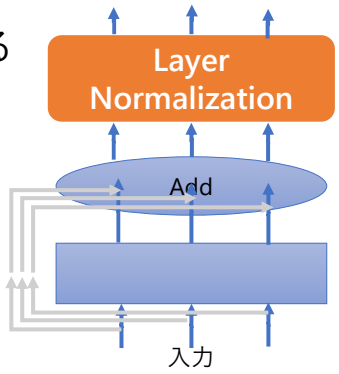
層を単純に重ねると学習しにくくなる。一方、層を深くして複雑な構造を学習させたい。
⇒ Residual Connectionで層を深くしても学習するようにする

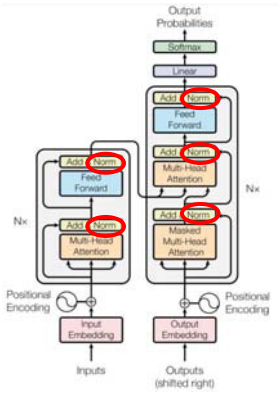
各ブロックでは、Attentionや全結合レイヤの後に、Residual Connectionというテクが施されます。
これは、図に示すように、ブロックへの生入力をレイヤをバイパスして、出力に加える手法で
残余ConnectionとかスキップConnectionとか呼ばれます。
層を深くしても学習してくれる効果があるそうです。

Layer Normalization

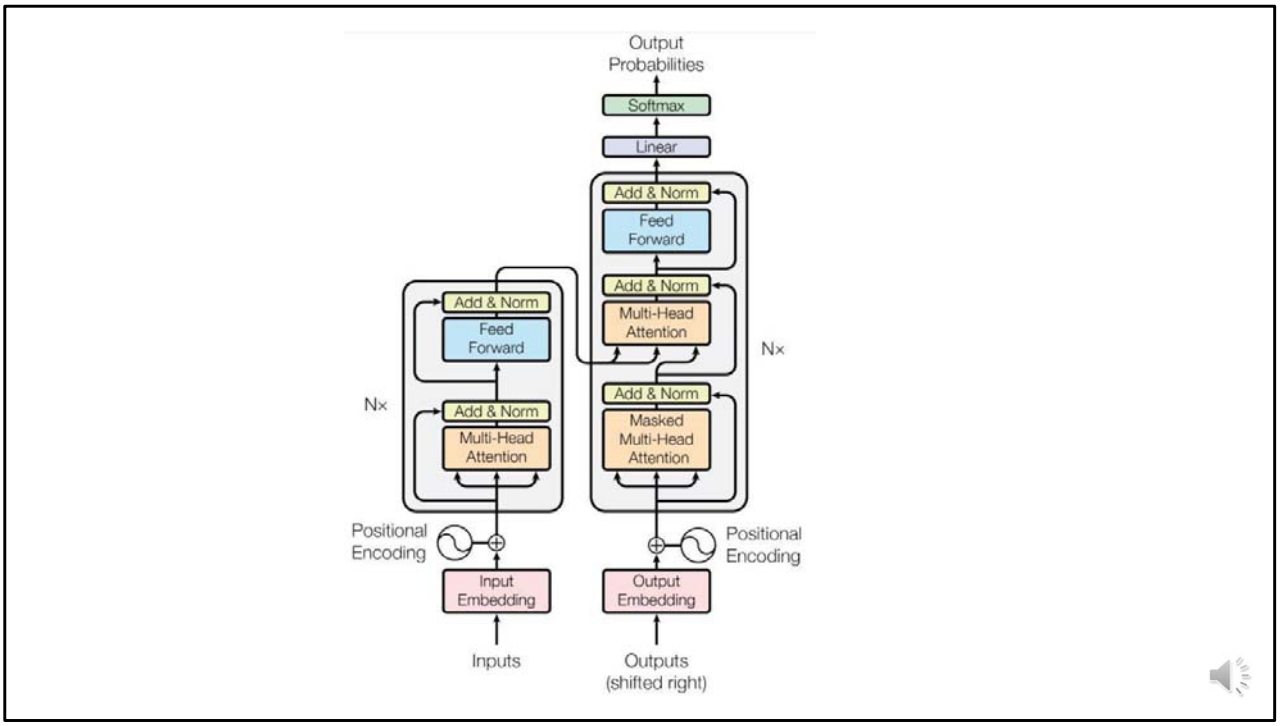
平均と分散を見て、ばらつきを抑える

⇒学習の高速化、過学習抑制





また、各ブロックの、Residual Connectionのあとに、レイヤNormalization というテクが
施されます。
データの平均と分散を使って、データのばらつきを抑える手法です。
バッチNormalizationと同様に、学習の高速化や過学習の抑制効果があるそうです。



以上で、Self-Attention、Source-Target Attention、Masked LM, Causal LMという4つの概念をもとに、BERTの主要ブロックの意味・役割と全体構成を見てきました。

効果：SoTA(State-of-The-Art)記録を塗り替えた

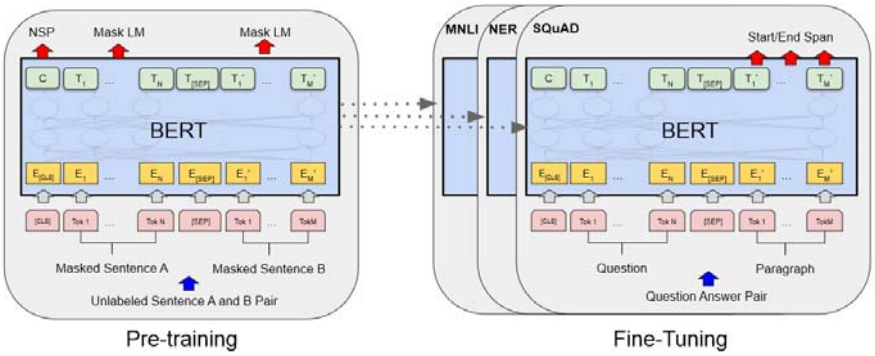
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Attention Is All You Need, 2017

論文によれば、BERTは、いろいろなSoTA記録を塗り替えたそうです。
これ以降、今は、自然言語処理はTransformerが主役となった感じです。
また、画像処理の分野も影響を受け始めています。

Fine-Tuning：事前にトレーニングしたBERTの出力を、タスクに応じたデータで再度Tuning

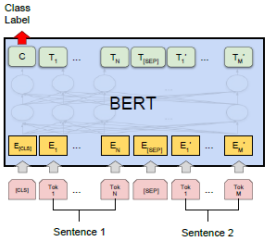


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding、2018

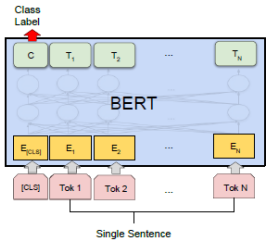
BERTは、あらかじめトレーニングしてある汎用システムです。
具体的なタスクに落とすには、Fine-Tuningといって、後付けでタスク特殊な学習を行います。

取り出す情報を変えて、タスクに合わせる

出力の先頭ラベルで相次ぐ文の関係の分類



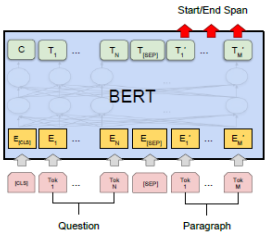
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



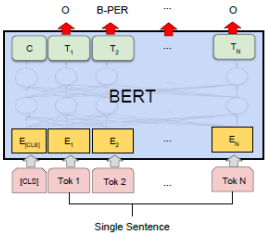
(b) Single Sentence Classification Tasks: SST-2, CoLA

出力の先頭ラベルで文の分類

次文の開始マーカ・終了マーカを取り出して、次文予測



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

単語にタグ付け（場所表現取り出しとか）

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding、2018

例えば、BERTの出力は、一文の先頭にカテゴリーラベルというのが出力されます。
文の分類タスクでは、そこに文の分類ラベルを投入してFine-Tuningします。

HuggingfaceのTransformerの分類

自動回帰モデル	自動符号化モデル	SEQ2SEQモデル
• Causal LM • 応用：文生成 • 例：GPT、BERTのデコーダー	• Masked LM • 応用：文分類、単語タギング • 例：BERTのエンコーダー	• エンコーダー・デコーダー構成のもの • 応用：機械翻訳、要約、Q&A • 例：BERTの翻訳タスク版、T5

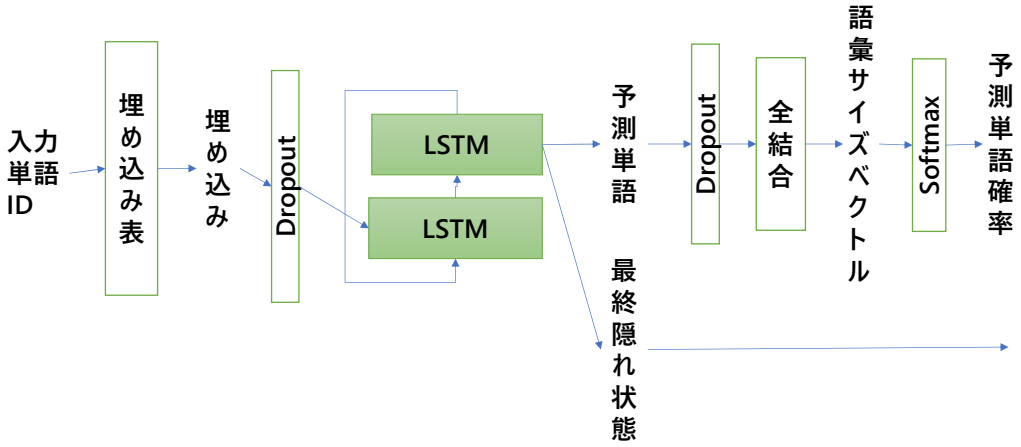
https://huggingface.co/transformers/model_summary.html

ここで、HuggingFaceサイトでのTransformerの分類を見ます。
HuggingFaceでは、TransformerはSelf-Attentionベースのものと定義しています。
それらは自動回帰モデル、自動符号化モデル、Seq2seqモデルなどに分類されています。
自動回帰モデルは、Causal LMを使うもので、文生成に利用されます。GPTが有名です。BERTのデコーダもこれです。
自動符号化モデルは、Masked LMを使うもので、文の分類や単語タギングや固有名詞抽出などに利用されます。
BERTのエンコーダーが典型です。
SEQ2SEQモデルは、エンコーダー・デコーダ構成をとるもので、機械翻訳、Q＆Aなど異質なものの対応をとるタスク向きです。BERTのオリジナルやT5が有名だそうです。

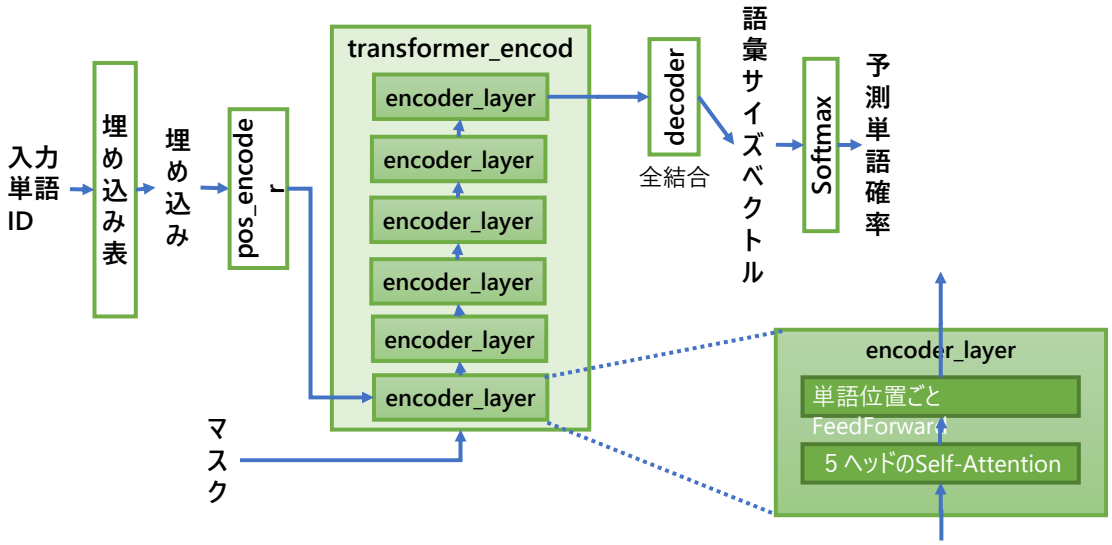
Transformer 課題 1

- attention_languagemodel.ipynb があります。rnn_languagemodel のサンプルの Transformer 版です。これを読解します。
- 実行ログを追加て、提出しなさい。

振り返り：LSTMを使った言語モデルのネット構成



Transformerを使った言語モデルのネット構成



参考資料：Transformer,BERT

- Original Paper
  - Attention Is All You Need, 201
  - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018
- The Illustrated Transformer
- 作って理解する Transformer / Attention
- Harvard NLP, The Annotated Transformer
- 深層学習界の大前提Transformerの論文解説！

参考資料：画像処理でのAttention

- Self-Attentionを全面的に使った新時代の画像認識モデルを解説！
- Exploring Self-attention for Image Recognition

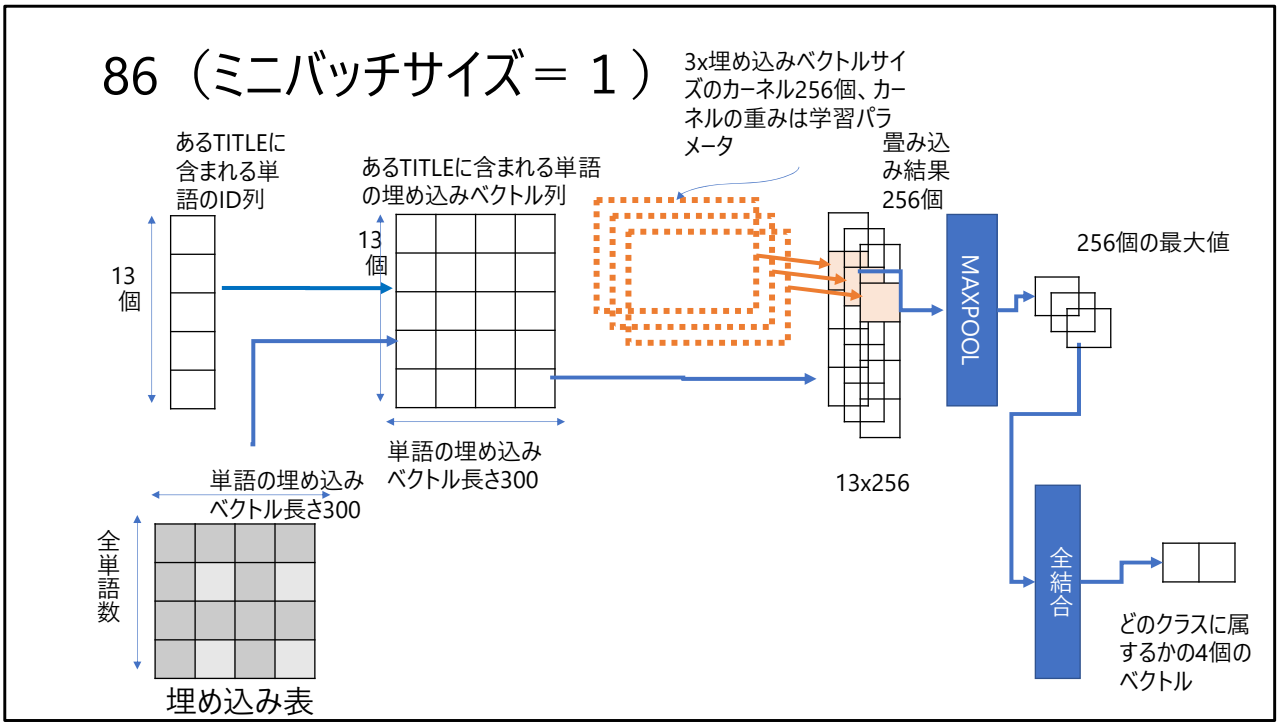
100本ノック第9章課題89

- ニューラルネットのコードに慣れてきました。「100本ノック」の9章の課題 の89(Transformer)の回答例を読みましょう。
- 「CNNRNNTransformer.ipynb」というノートをコピーし、89の回答例コードを読解してください。80をやってから実行ログを残してください。

RNN,CNN,Transformerを利用するコードが理解できたら、すでに、高度なテーマでも、NLPとDeep Learningを自力で深めていく力がついています。

確認クイズ

- スタログの確認クイズをやってください。


86（ミニバッチサイズ＝１）
3x埋め込みベクトルサイズのカーネル256個、カーネルの重みは学習パラメータ
あるTITLEに含まれる単語のID列
13個
あるTITLEに含まれる単語の埋め込みベクトル列
13個
畳み込み結果256個
MAXPOOL
256個の最大値
単語の埋め込みベクトル長さ300
単語の埋め込みベクトル長さ300
13x256
全単語数
全結合
どのクラスに属するかの4個のベクトル
埋め込み表