学生・若手研究者のための BERTワークショップ

質問解答編

RIKEN AIP 中山功太

バッチ入力に関して

Q. バッチ入力の「学習を安定させる効果」とはどういうことでしょうか?
A. 「モデルが学習中に好ましくないパラメータに陥るのを防ぎ、より理想的なモデルを得られる」といった効果を意味します。

バッチ入力だとなぜ学習が安定するのか?

深層学習モデルの性能を上げるためには、希少な問題より頻出する問題の方が良く解けていることが好ましい。

そのため、毎回のパラメータ更新はなるべく入力分布にとって最適なものである必要がある。

バッチ入力は、入力分布をある程度近似でき、誤差の平均から得られる勾配も入力分布を反映した方向となる。
⇨学習が安定する。

入力テキスト	正解ラベル
吾輩は猫である。 名前はまだ無い。	夏目漱石
私は、その男の写真を三葉、見たことがある。	太宰治
親譲の無鉄砲で小供の時から損ばかりしている。	夏目漱石
…	…

間違ったラベルがつけられたデータを中和してくれる効果もある。

サブワード分割に関して

Q. サブワード分割の前にMeCab等で単語分割をおこなっているのでしょうか?
A. はい。MeCabなどの形態素解析器で単語分割をおこなった後、BPE(バイトペアエンコーディング)などの手法を用いてサブワードに分割しています。

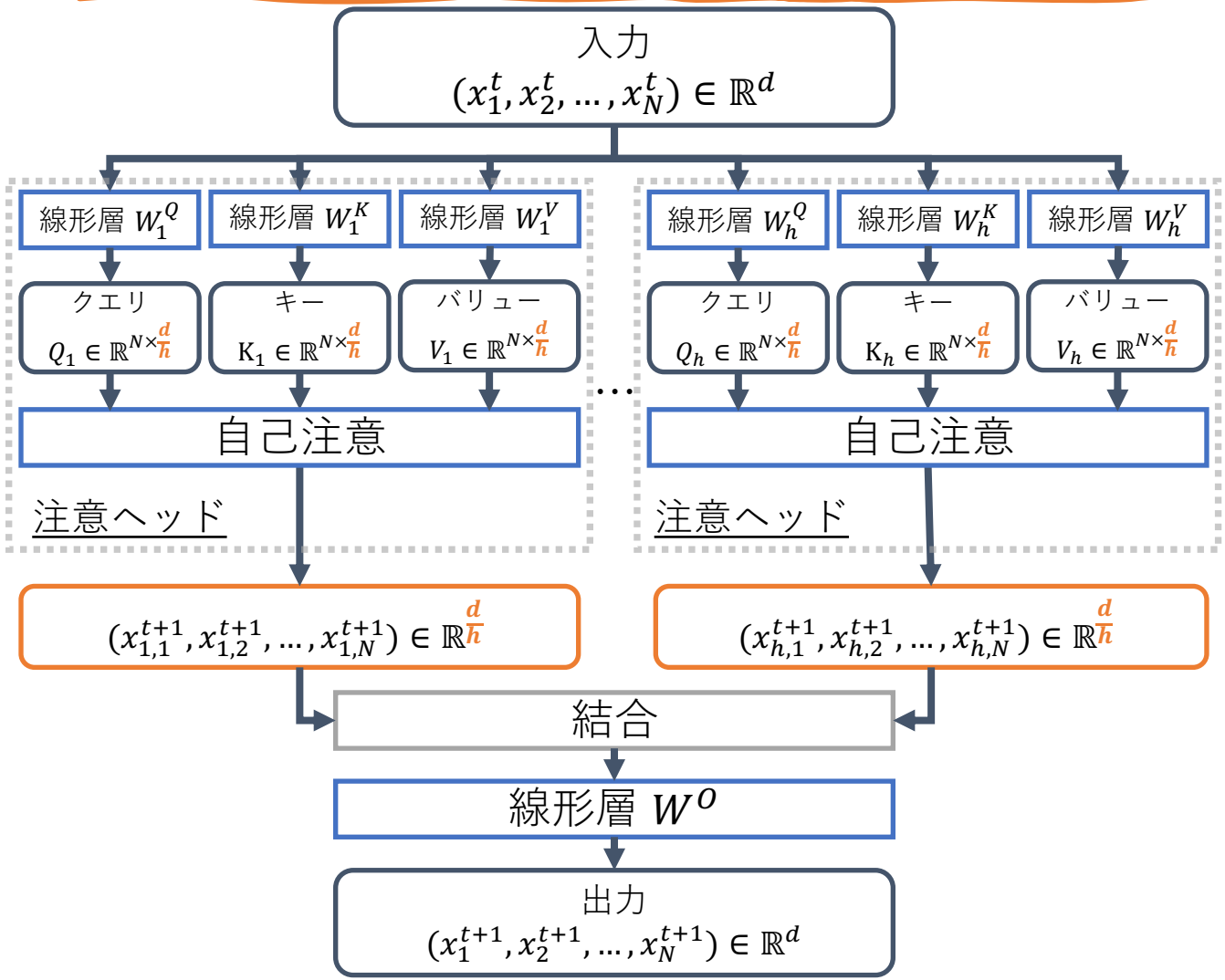
入力文章　　本日は晴天なり。

↓ 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, 。]

ちなみに、サブワードレベルへの分割を行うと、全体の語彙数を大幅に圧縮することができます。
サブワードの細かさは調節可能ですが、事前学習を行う前に決めておく必要があります。
分割を細かくすると全体の語彙数をより圧縮できますが、逆に分割後のトークン列が長くなります。

各注意ヘッドの出力の意味



Q. 注意ヘッドの出力は何かを意味しているものなのでしょうか?
A. 注意ヘッドの出力は、各注意ヘッドで捉えた文脈を考慮した単語の意味を表していると考えられます。また、それぞれの注意ヘッドは別々のパラメータを持っているため、文章のさまざまな側面から捉えられた意味を表してくれていると考えられます。

次文予測(Next Sentence Prediction)とは

Q. 次文予測とはなんでしょうか。なぜ役に立たないと言われているのでしょうか。
A. [SEP]トークンで連結された文章ペアが流れとして正しいかどうかを判定する分類タスク。
本タスクによりモデルが長期的な依存関係を学習することが期待される。
50%の確率で次の文章を大規模コーパスからサンプリングした文章に置き換えてデータセットを作成する。

正例

[CLS]吾輩は猫である。名前はまだ無い。[SEP]どこで生れたかとんと見当がつかぬ。[SEP]

[CLS]親譲りの無鉄砲で小供の時から損ばかりしている。[SEP]小学校に居る時分学校の二階から飛び降りて …[SEP]

負例

[CLS]吾輩は猫である。名前はまだ無い。[SEP]小学校に居る時分学校の二階から飛び降りて …[SEP]

[CLS]親譲りの無鉄砲で小供の時から損ばかりしている。[SEP]どこで生れたかとんと見当がつかぬ。[SEP]

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

RoBERTa: A Robustly Optimized BERT Pretraining Approach(2019)であまり効果がないことが指摘された。
次文予測を並列して用いた場合、マスク言語モデリングでは50%の確率でしか文章全体を参照できないデメリットが存在する。次文予測タスクが簡単すぎるため、このデメリットの方が目立ってしまい効果が低いと思われる。

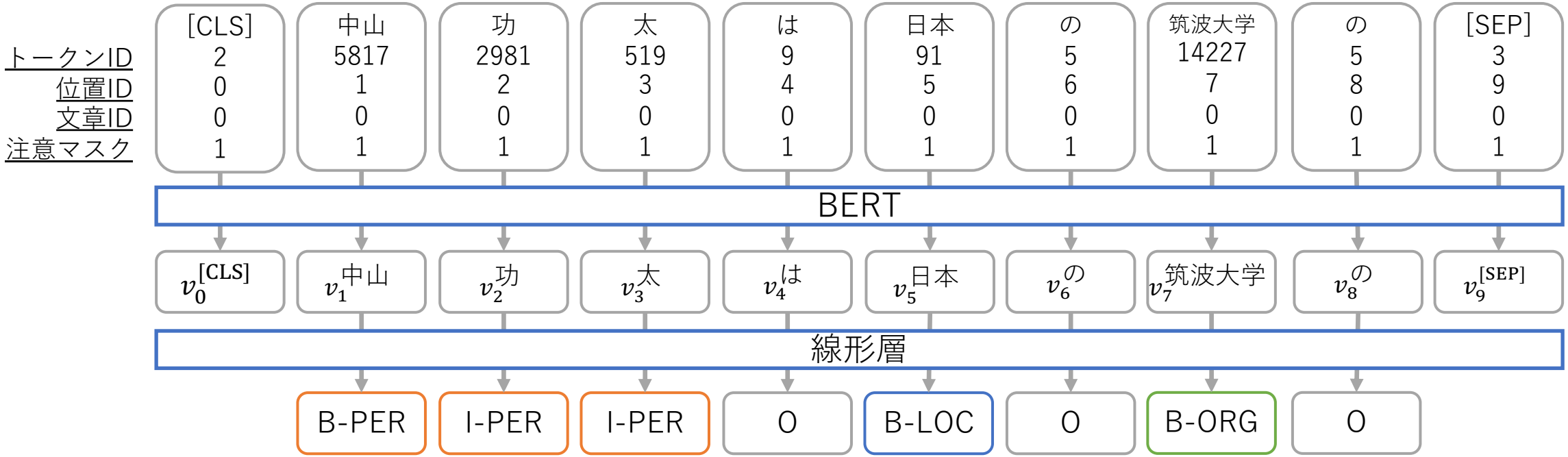
BERTを固有表現抽出に適用したい

Q. BERTは固有表現抽出のようなタスクに適用できますか?
A. できます。トークン分類タスクとして解くことになります。

固有表現抽出タスクとは
文章中から固有表現に関する言及を抽出するタスクである。
どこまでを固有表現とするかはタスクごとに決められている。
人名(PER)、**施設名(ORG)**、**地名(LOC)**を対象とするタスクの場合、
「**中山功太**は**日本**の**筑波大学**の学生である。」となる。
PER LOC ORG

BERTを固有表現抽出に適用したい

BERTを固有表現抽出に適用する場合、各トークンを
人名(B-PER, I-PER), 組織名(B-ORG, I-ORG), 地名(B-LOC, I-LOC), それ以外 O
の7クラスに分類するタスクとして解くことができる。
ここでB-は範囲の先頭を示し、I-は範囲の継続を示す。



今回のコードを感情分類に適用したい

感情分類とは
入力文章がどの感情に属するのかを判定するタスク。
簡単なものではポジティブ、ネガティブ、ニュートラルに分類する。
飲食店のレビューを対象とする場合「スープが美味しかった」「店内が汚かった」「いい天気でした。」
ポジティブ　ネガティブ　ニュートラル

Q. 感情分類に適用したいが今回のコードを適用できる?
A. 以下の場所を書き換えれば、動くかと思います。
1. データの読み込みの部分を書き替える。
2. データセットのテキストを取り出す部分と、クラスラベルを作成する部分を書き換える。
3. 評価が正答率ではない場合は、評価関数を適したものに書き換える。

今回のコードは、文章に対して複数のラベルがつかないタスク全般に流用できます。

```
class ClassificationDataset(Dataset):
  def __init__(self, data, labels, ene_id_list):
    self.tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME) # 東北大BERTで使用されてい
    self.data = data
    self.labels = labels
    self.ene_id_list = ene_id_list # 学習データに存在するラベルのリスト
    self.num_labels = len(ene_id_list) # ラベル種類の総数

  def __len__(self): # データセットの長さ(記事総数)を返す。
    return len(self.data)

  def __getitem__(self, i):
    page_id, text = self.data[i] # i番目の文章取り出し
    d = self.tokenizer(
      text,
      max_length=MAX_LENGTH,
      truncation=True,
      padding="max_length"
    ) # MAX_LENGTHまでの長さのBERTの入力を自動作成

    # 深層学習モデルに入力する配列はテンソルに変換されている必要があります。
    d["input_ids"] = torch.LongTensor(d["input_ids"]) # テンソルに変換(int64)
    d["token_type_ids"] = torch.LongTensor(d["token_type_ids"]) # テンソルに変換(int64)
    d["attention_mask"] = torch.BoolTensor(d["attention_mask"]) # テンソルに変換(bool)

    d["page_id"] = page_id # どのデータを処理したかわかるようにpage_idを渡しておく。

    if self.labels is not None: # ラベルが存在する場合
      label = self.labels[page_id][0] # ラベルを取り出し
      d["labels"] = self.ene_id_list.index(label) # ラベルリストからインデックスを取得する

    return d
```

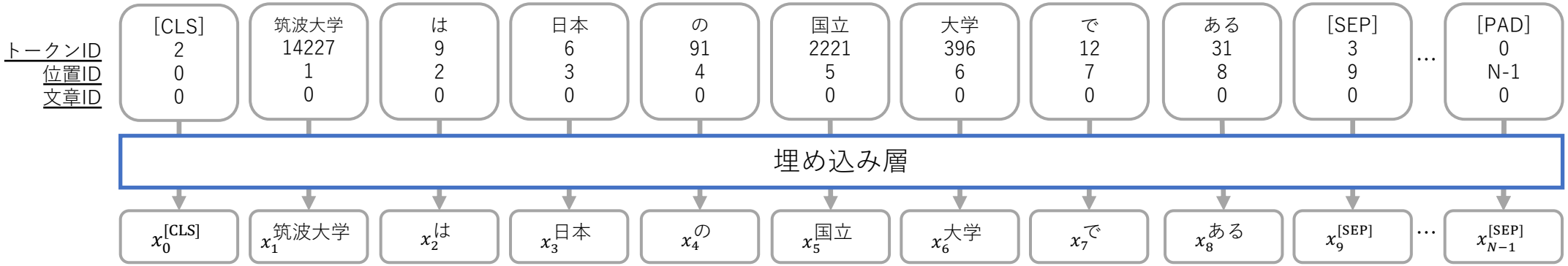
長いトークン列を使いたい⓪

Q. 長いトークン列を使いたいのですが方法はありますか?
A. 知っている限りにはなりますがいくつか方法をご紹介します。

BERTの計算で最も重い部分は自己注意のQK^Tであり、入力長に対する計算量は$O(n^2)$である。
⇨入力長に対してシビアに計算量が増える。

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

そのため、BERTの事前学習では入力長を512トークン前後に制限する場合が多い。
また、BERTでは位置IDに対する埋め込みを採用しており、事前学習時に使用されたトークン長以下の入力しか扱うことができない。

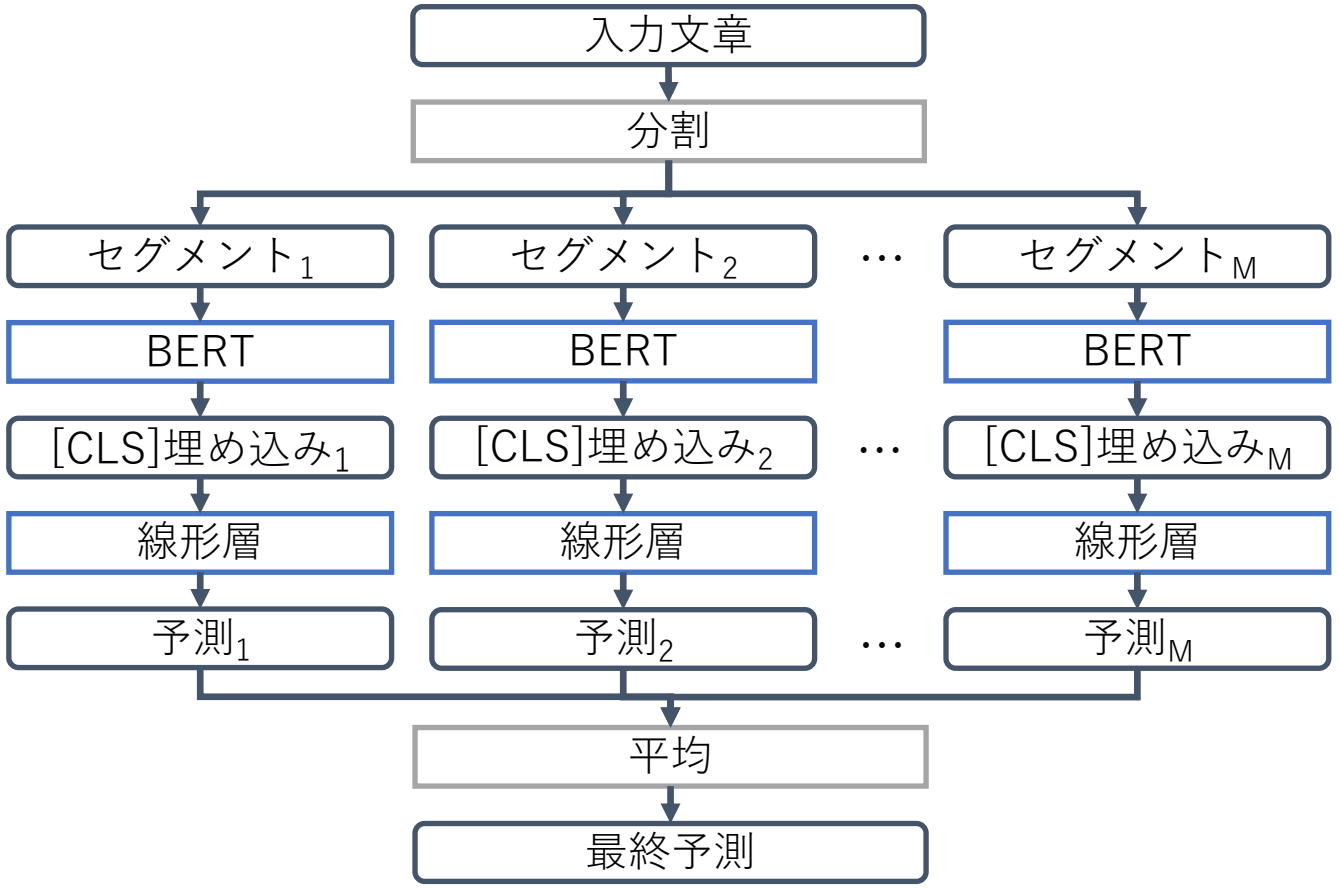


長いトークン列を使いたい①

入力文章を入力長以下のセグメントに区切り、
BERTと線形層を用いてそれぞれの予測を出力する。
最終予測はそれらの平均によって得る。

勾配計算による学習は、
①各予測と正答ラベルとの誤差を独立して逆伝播
もしくは、
②予測平均と正答ラベルとの誤差を逆伝播
することで行うことができる。

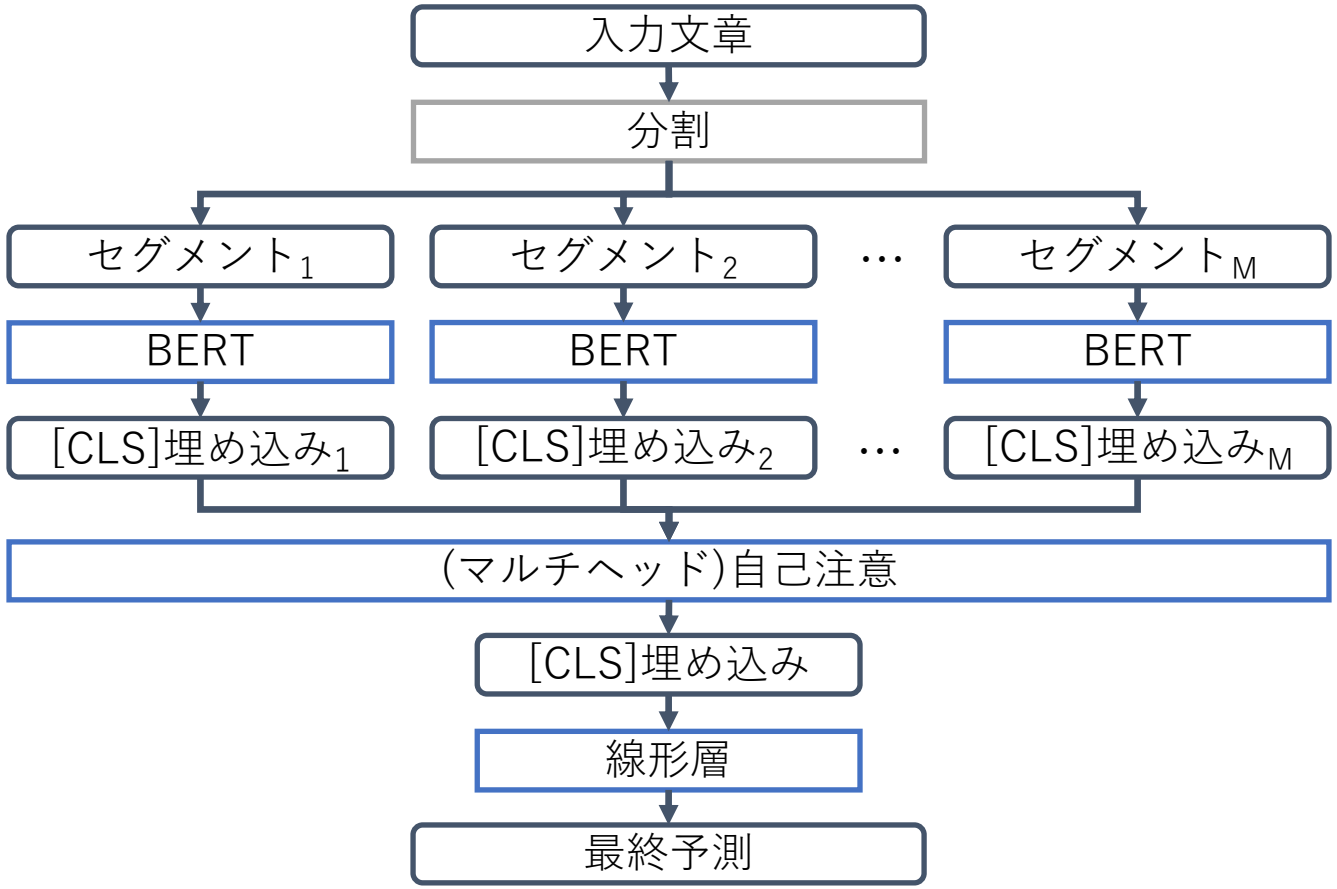
①の場合は、いずれかのセグメントにおいて正答ラベルに関するヒントが含まれない場合に学習に矛盾が発生し、②の場合は全てのセグメントの計算グラフを保持しておく必要があるため膨大な計算メモリーを必要とする。



長いトークン列を使いたい②

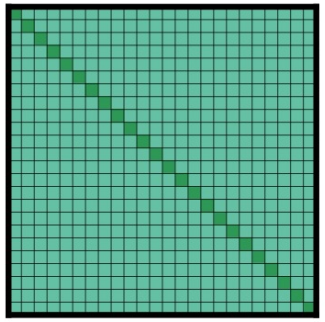
入力文章を入力長以下のセグメントに区切る部分は同じだが、BERTから得られた[CLS]埋め込みを自己注意と線形層に入力することで最終予測を得る。
自己注意によりセグメント間の関係を捉えて最終予測を行うことができる。

全てのセグメントの計算グラフを保持しておく必要があるため膨大な計算メモリーを必要とする。

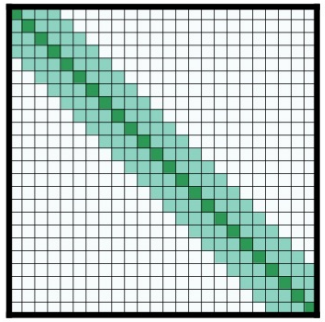


長いトークン列を使いたい③

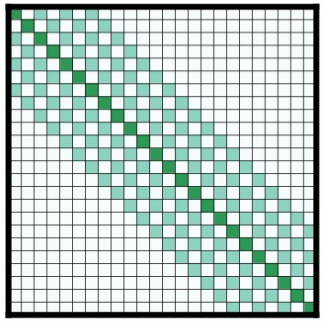
手法ベースでの改良も行われている。
Longformer: The Long-Document Transformer (2020)で提案された手法の場合、
近隣単語のみとの注意を計算することで計算コストを削減している。
[CLS]トークンなどは文章全体を見る必要があるため全トークンとの注意を計算する。



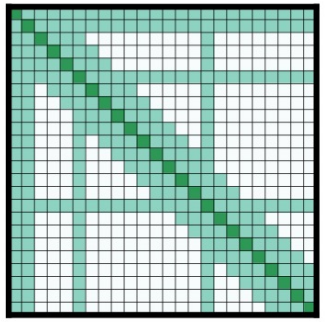
(a) Full n^2 attention



(b) Sliding window attention



(c) Dilated sliding window



(d) Global+sliding window

デメリットとしては、日本語で事前学習されたモデルが存在しないことが挙げられる。

BERTの選び方

Q. ファインチューニング(事前学習後に別タスクで学習することを意味する)に用いる事前学習済みモデルは予測タスクによって適切なものを選ぶ必要はあるか。
A. あります。

事前学習済みBERTを選ぶ際に特に重要になる点は2つ

1. トークン分割の適性
   分類のような文章全体から1つの予測を導くタスクでは強く影響しないが、固有表現抽出のように各トークンを分類するような場合、トークン分割が粗いモデルだと固有表現の境界線とトークンの境界線が合わない場合があるため、分割が細かいモデルを選ぶ必要がある。
2. 事前学習に使用されたコーパスの適性
   経済、医療、化学などのタスクに適用する場合、それぞれの分野のコーパスで事前学習されたモデルを用いた方が良い。ちなみにワークショップで使用した東北大BERTはWikipediaで事前学習されている。

XLNetとの違い

XLNet: Generalized Autoregressive Pretraining for Language Understanding(2019)で提案されたモデル。
BERTにおける[MASK]のデメリットを克服しているが、計算コストがBERTより重い。

New, York, is, a, city といった入力を考える

- BERTのマスク言語モデリングの場合
  [MASK]された単語を予測する。
  New, York, [MASK], a, city ⇦**[MASK]という通常は使用されない単語が使用されている。**
  [MASK], [MASK], is, a, city ⇦**複数単語マスクされた場合2つの単語の間の予測を活用できない。**
- 通常の言語モデリングの場合
  現在までの単語列から次の単語を予測する。
  New, York, ? ⇦**後半に出てくる単語の情報が使用できない。**
- XLNetのモデリングの場合
  語順を入れ替えて言語モデリングを行う。(各単語は元の位置を保存している)
  is, New, city, a, ?
  以上をすべての順列で行う。**後半に出てくる単語を使用しつつ他の予測結果も使用できる。**

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city}),$$

$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{New}, \text{is a city}).$$

意味のクラスタリングは可能?

Q. BERTが文脈を加味しているなら単語の意味でクラスタリングできるのでは?
A. 可能であり、実際に多くの研究がなされています。

質問で挙げられていた「甘い」という単語の意味に関して実際に東北大のBERTで試してみる。

以下の4つの文章をBERTでエンコーディングし、
「甘い」に対応する出力ベクトルのコサイン類似度を計算した。

P1：砂糖が**甘い**。
P2：カレーが**甘い**。
N1：詰めが**甘い**。
N2：考えが**甘い**。

	P1	P2	N1	N2
P1	1.000	**0.892**	0.876	0.850
P2	**0.892**	1.000	0.879	0.865
N1	0.876	0.879	1.000	**0.902**
N2	0.850	0.865	**0.902**	1.000

味覚的の一つを示すP1とP2、足りなさを示すN1とN2のペアの類似度が高い、
つまり超球面上で集まっていることがわかる。

使用したColabは[こちら]。