@Kosuke-Szk (/Kosuke-Szk) 2018年11月27日に更新
(/Kosuke-Szk)

# 汎用言語表現モデルBERTを日本語で動かす(PyTorch)

NLP(/tags/nlp) DeepLearning(/tags/deeplearning) Python3(/tags/python3) PyTorch(/tags/pytorch) bert(/tags/bert)



今DL for NLP界で、BERTというモデルが話題です。PyTorchによる実装が公開されていたので、日本語Wikipediaコーパスに適用してみました。

コードはこちら (https://github.com/Kosuke-Szk/ja_text_bert)に公開しております。

2018/11/27
作成したBERTのモデルを使って内部動作の観察とその考察を行いました。単語の潜在表現獲得の部分で感動的な結果を見せてくれました。ご興味あればご覧ください↓
https://qiita.com/Kosuke-Szk/items/d49e2127bf95a1a8e19f (https://qiita.com/Kosuke-Szk/items/d49e2127bf95a1a8e19f)

この記事ではBERTのポイントの解説と、ポイントごとの実装を紹介します。
尚、記事の執筆にあたってこちらのリポジトリを参考にさせていただきました。
https://github.com/codertimo/BERT-pytorch (https://github.com/codertimo/BERT-pytorch)

本記事は以下の4つで構成されています。
・BERTとは
・BERTのキモ
・BERTの弱点
・実装
・考察

## BERTとは

(https://camo.qiitausercontent.com/daf9623770c4143218bc63ceb7c1a8386a031ac8/68747470733a2f2f71696974612d696d6167652d73746f72652e73732e616d617a6f6e6177732e636f6d2f302f3132333538392f313063373461366632d393531312d303962612d653165622d366130383439343463313538 2e706e67)

## 概要

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[論文 (https://arxiv.org/abs/1810.04805)]
著者:Jacob Devlin, Ming-Wei Chang, Kenton Lee,
Kristina Toutanova(Google AI Language)

2018年10月11日にGoogleからArxiv公開された論文です。双方向Transformerで言語モデルを事前学習することで汎用性を獲得し、転移学習させると8個のベンチマークタスクでSOTAを達成したそうです。

| タスク | 概要 | 前SOTA | BERT |
|---|---|---|---|
| GLUE | 8種の言語理解タスク | 75.2 | 81.9 |
| 1. MNLI | 2入力文の含意/矛盾/中立を判定 | 82.1 | 86.7 |
| 2. QQP | 2質問文が意味的に等価か判定 | 70.3 | 72.1 |
| 3. QNLI | SQuADの改変．陳述文が質問文の解答を含むか判定 | 88.1 | 91.1 |
| 4. SST-2 | 映画レビューの入力文のネガポジを判定 | 91.3 | 94.9 |
| 5. CoLA | 入力文が言語的に正しいか判定 | 45.4 | 60.5 |
| 6. STS-B | ニュース見出しの2入力文の意味的類似性をスコア付け | 80.0 | 86.5 |
| 7. MRPC | ニュース記事の2入力文の意味的等価性を判定 | 82.3 | 89.3 |
| 8. RTE | 2入力文の含意を判定 | 56.0 | 70.1 |
| SQuAD | 質疑応答タスク．陳述文から質問文の解答を抽出 | 91.7 | 93.2 |
| CoNLL | 固有表現抽出タスク．単語に人物/組織/位置のタグ付け | 92.6 | 92.8 |
| SWAG | 入力文に後続する文を4つの候補文から選択 | 59.2 | 86.3 |

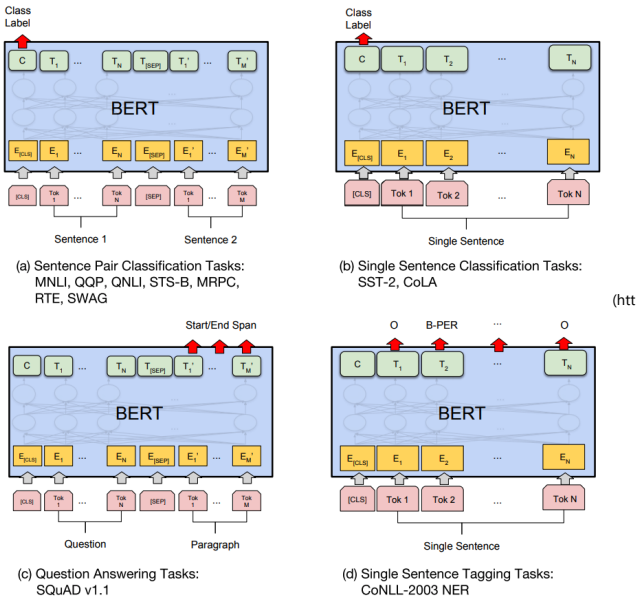
(https://camo.qiitausercontent.com/7fde3dae3649c60c051b23ae17a3da25b8d9caf4/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f313233353839 2f303837343537646 42d653662302d316461642d373639362d623539633561366665343239 2e706e67)
(引用元: https://twitter.com/_Ryobot/status/1050925881894400000 (https://twitter.com/_Ryobot/status/1050925881894400000))
ComputerVisionではImageNetで事前学習したVGGやResNetがありますが、今回BERTによって言語でそのようなポジションのモデルが作られました。

## 事前学習（Pre-Training）

BERTがここまでさわがれている理由のひとつに、BERTが自然言語処理における汎用的な事前学習モデルであることがあります。

そもそも、何かのデータを統計的に処理したい時に、画像処理の場合はピクセル、音声処理の場合は音声信号といったように基本入力が決まっているのに対し、自然言語では処理対象とするテキストをどのように表現すべきか決まった方法がなく、タスクに応じて決定されます。例えば、情報抽出の場合はテキストを単語の集合(Bag of words)として扱うのが主流で、機械翻訳のようなより高度なタスクでは品詞解析、係り受け解析などより複雑な表現方法が用いられることがあります。統計的自然言語処理では、テキストをどのような特徴量を使って表現するかの選択が重要です。
自然言語処理では単語の組合せ(n-gram)を古くから特徴量として用いてきましたが、組合せの種類が膨大になり、十分な訓練サンプルが確保できなくなるなどの問題がありました。
「表現学習」はこれらの有効な特徴の組み合わせを自動的に学習させる手法です。深層学習では層を深く重ねることで単純な特徴量の組み合わせからより複雑な特徴の表現を獲得できます。
大規模なデータを元に表現学習を行うと、汎用的な特徴量抽出器となることが期待されます。
様々なタスクへの応用を前提としたこの表現学習を事前学習(Pre-Training)と呼びます。
単語の出現は前後の文脈に依存して決まるので、単語や文章同士の一般的な依存関係が事前に与えられていれば、あるタスクを解くために必要な特徴が入力に出現していない場合でもそれを補うことができます。特に、特徴の出現がスパース(疎)な自然言語の場合は、事前学習は重要な役割を持つイメージが付きやすいと思います。
そして、事後学習(Post-Training)では事前学習で得られた有効な特徴の組合せを使って目的とするタスクを学習します。

BERTは、大規模コーパスから事前学習させたモデルに対して、タスクごとに事後学習を行わせた結果、あちらもこちらも最高得点(State of the Art)を叩き出したので、あらゆるタスクはこのBERTを土台にして取り組めるのではという機運が高まっているわけですね。
以上が、BERTが汎用的な言語モデルとして注目を集める所以です。




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

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

(c) Question Answering Tasks: SQuAD v1.1

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

(htt

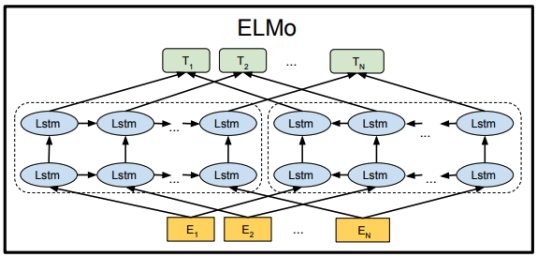
ps://camo.qiitausercontent.com/46ce0feed4f5f5fbe753dc02910ca045bf49b27b/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f313233353538392f623838353503864612d613961662d383165392d333530382d663864623265333336626465 2e706e67)

## BERTを作るモチベーション

上記の背景から、様々な自然言語処理タスクにおいて事前学習が有効なことが示されています。
・文レベルのタスク→文同士の関係性を学ぶ
・トークンレベル→トークン同士の関係性を学ぶ
事前学習のアプローチには大きく二種類あります。

### Feature basedアプローチ

N-gramモデルやWord2Vecなどがこれに当たります。独自の分散表現で、様々なNLPタスクの素性として活用されます。
最近ではELMo[Peters et al., 2017, 2018]が有名です。獲得した素性をNLPタスクの入力層及び隠れそうに連結するだけで、性能の向上が図れるというものです。

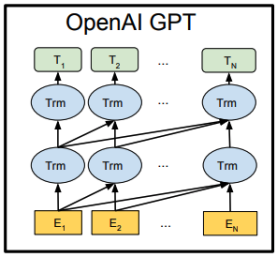



(https://camo.q

iitausercontent.com/ec45d166a80a63367acd5d97ef332ccfbefe3969/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f313233353538392f343231623232323939642d363165352d396261342d323435302d306531373338653564306 4352e706e67)

### Fine tuningアプローチ

言語モデル(次の単語を予測するモデル)を生成する形で事前学習を行います。その後に教師ありタスクでFineTuningすることでタスクにフィットさせるアプローチです。
最近ではOpenAI GPT[Radford et al. 2018]が有名です。



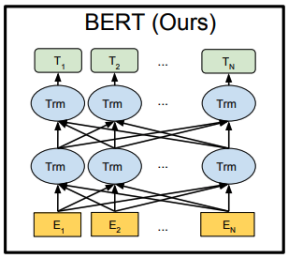

(https://camo.qiitausercontent.com/8c0eb4352edfad5

43803c1a126ed21b17ca90824/68747470733a2f2f71696974612d696d6167652d73746f72652e733
32e616d617a6f6e6177732e636f6d2f302f3132333538392f373966396466353924383739332d3061
66312d663638632d383637386339626433313939 2e706e67)

**もっと良い事前学習モデルが欲しい！**

BERTは従来の事前学習の有効性を踏襲しつつ、弱点を克服する形でより汎用性の高い事前学習モデルを学習できたところが革新的です。

## BERTのモデル

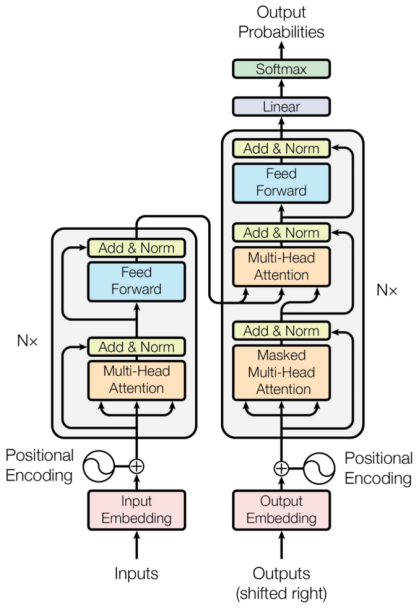



(https://camo.qiitausercontent.com/dce8f663b7760b

fae4faed56bd2cd1658f13140c/68747470733a2f2f71696974612d696d6167652d73746f72652e733
32e616d617a6f6e6177732e636f6d2f302f3132333538392f646337353565353 12d363035642d3731
31612d646437322d3265366130613262643131372e706e67)
BERTは双方向Transformerです。
です。と言われてもよくわからないので、そもそもTransformerとは何で、従来と何が違うのかから探ってみましょう。

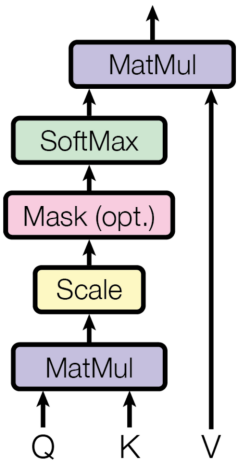
### Transformerとは

Transformer( 論文 (https://arxiv.org/abs/1706.03762))は、Attentionのみを使用したニューラル翻訳モデルで、わずかな訓練でSOTAを達成しました。Attention機構は、加法注意・内積注意・ソースターゲット注意・自己注意に分類されます。この中でも、自己注意(Self-Attention)は汎用的かつ強力であり、言語処理のあらゆるタスクにおいて高い性能を発揮します。
Transformerに関してはこちらのブログ (http://deeplearning.hatenablog.com/entry/transformer)がとても参考になります。




(https://camo.qiitausercontent

nt.com/eecd1e53ad2e662a0cdf32a3661c96942306b9c6/68747470733a2f2f71696974612d696d61
67652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3132333538392f356166616164
66652d343932392d333266372d393161332d373139383630346266666393 22e706e67)
BERTではこのTransformerのEncoder部分をユニットとして活用しています。
ユニットの中身を拡大するとこのようになっています。Q:Query, K:Key, V:Valueにそれぞれ対応しています。Q,K,V３つのテンソルを放り込むとよしなに動いてくれるのがSelf-Attentionです。
詳しく見てみましょう。



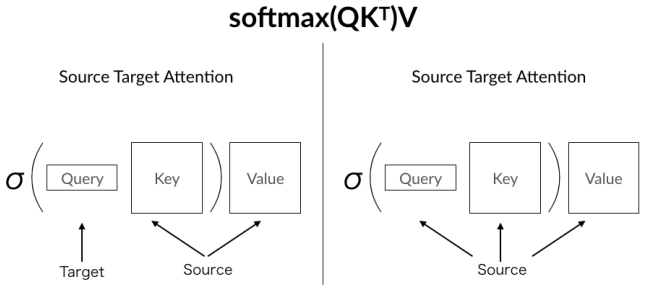

(https://camo.qiitausercontent.com/adb85a42d6729fd617f1adc50e4ace06e92375be/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f313233353839 2f666365396636616 22d366238382d303163362d623866302d666434356239313933364336 52e706e67)

### そもそもAttentionとは

Attentionにおいて、Query, Key, Valueと名前がついているのにはわけがあります。Attentionでは、Query(検索クエリ)に一致するKeyを索引し、対応するValueを取り出す操作と見ることができます。Encoder-Decoderモデルにおける Attentionでは、隠れ層のの内部状態をValueとして、Queryに関するvalueのみをAttentionの重みの加重和として取り出します。必要な情報にだけ"注意"しているわけです。

### Self Attentionとは

Attentionは、Key-Valueがどこをソースとするかで二種類に分類されます。



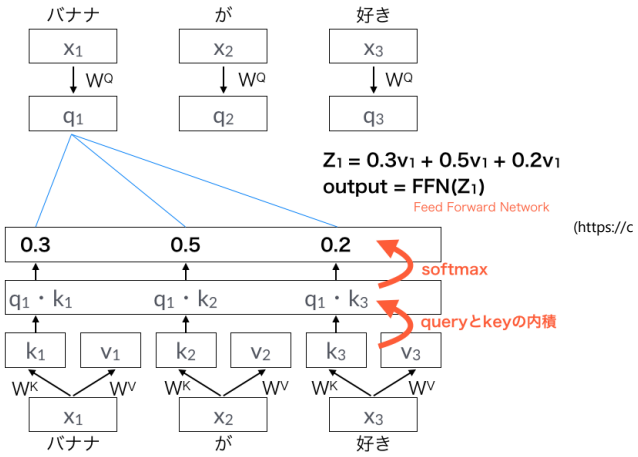

(https://camo.qiitausercontent.com/6dca36528bdc08339336ea9bde23ab6b226df6d3/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f313233353839 2f313233393626164612d613636392d636462322d663963352d353336646633643932 3638662e706e67)

#### Source-Target-Attention

KeyとValueはEncoderの隠れ層から、QueryはDecoderの隠れ層からきます。Decoderの要請に応じて注意を払う場所を決めるのがこちらのAttentionです。

#### Self-Attention

一方で、Query、Key、Valueが全て同じ場所からくるのが自己注意です。下の層からの同じ入力がQ,K,VとしてAttentionに流されます。

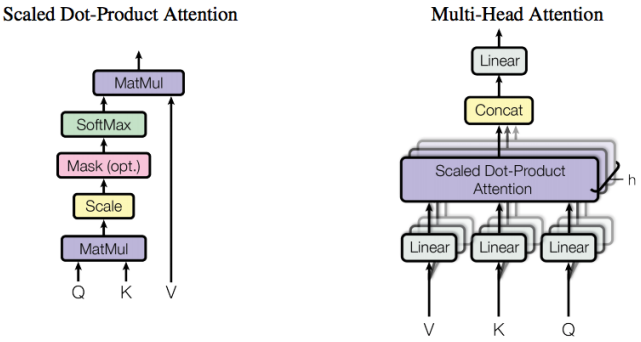



(https://camo.qiitausercontent.com/c0357c70af7308f9be5bb30ad4e69fa2f7a00629/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f313233353538392f373535666343237302d336331302d653134612d343033362d356162663930656566313731 2e706e67)

Self-Attentionは、下の層全ての情報の全てを参照してある位置の出力を自己定義することができます。
下の層の全てから、どの情報に注意して(重みを付加して)自己の出力を定義するかを学習的に決定できるのが、局所的な情報から出力を決定するCNNとの差異です。

#### Multi-Head-Attention

Multi-Head-AttentionはSelf-Attentionを並列に複数並べたものになります。
Scaled Dot-Product AttentionがSelf-Attentionのことで、この出力をConcatして上の全結合層に流します。



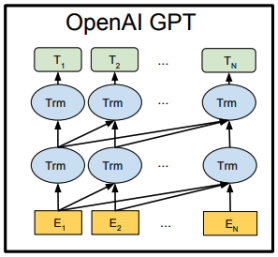

(https://camo.qiitausercontent.com/0c71ab10a88d718eedb3ffadfc9a3c3339b9f7f2/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f313233353538392f6363396633396333864302d633934632d326334372d366466662d323536623931616662343131302e706e67)

このMulti-Head-Attentionをひとつのユニット(BERTモデル図中のTrm)として、全結合的に接続したものがBERTモデルです。

### 従来との違いは？

事前学習を行う際、言語モデルを学習させることが一般的です。しかし、言語モデル(次の単語を予測するモデル)を事前学習に用いる場合、現在の予測プロセスの際に、予測するべき未来の単語情報を用いてはいけません。(カンニングになってしまうからです)

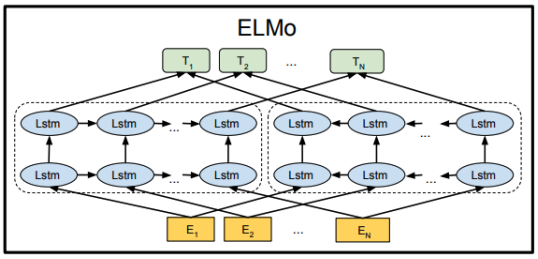
#### OpenAI GPT




(https://camo.qiitausercontent.com/8c0eb4352edfad543803c1a126ed21b17ca90824/68747470733a2f2f71696974612d696d6167652d73746f72652e733332e616d617a6f6e6177732e636f6d2f302f313233353839 2f373966396466353 92d383739332d306166312d663638632d383637386339626433331393 92e706e67)

OpenAI Transformerは未来の単語位置のネットワークにマスクをかけねばならず、単方向Transformerになります。

**ELMo**




(https://camo.qiitausercontent.com/ec45d166a80a63367acd5d97ef332ccfbefe3969/68747470733a2f2f71696974612d696d6167652d73746f72652e733332e616d617a6f6e6177732e636f6d2f302f313233353839 2f343231623232393964 2d363165352d396261342d323435302d306531373338653564 3064352e706e67)

ELMoでは順方向LSTMと逆方向LSTMを同時に学習することができません。

しかし、left-to-rightモデル(OpenAI GPT)や双方向の連結モデル(ELMo)より、更に強力な双方向モデル(BERT)を利用したいところです。

そこでBERTでは、次の単語を予測するのではなく、ランダムにマスクされた単語を周辺情報から予測することを事前学習タスクとしました。
従来、モデルの中に暗黙的に組み込まれていたマスクを、学習データのほうで直にマスク(または別単語に置換)したことで代替したのです。
**これによって双方向Transformerを使うことができるようになりました。**

## BERTのキモ

### 事前学習のタスク選択

#### 事前学習① マスク単語の予測

系列の15%を[MASK]トークンに置き換えて予測します。
そのうち、80%がマスク、10%がランダムな単語、10%を置き換えない方針で変換します。

<モチベーション>
片方向しか情報が伝播しないネットワークよりも、両方向に情報が伝播したほうが強力に学習できそうです。
しかしながら、双方向モデルは、一般的な条件付き言語モデルでは学習できません。
一般的な条件付き言語モデルというのは、

$$p(x_n|x_1, x_2, \ldots, x_{n-1})$$

のように、n-1番目までの情報からn番目の単語を予測するタスクです。前述したように、このタスクを解くときは現在の予測プロセスの際に、予測するべき未来の単語情報を用いないようにする制約が必要です。したがって、モデル自身は前後の文脈を考慮できず、前の文脈情報のみで次を推測しなければいけませんでした。
そこで、モデル側ではなく入力データ側を穴埋め問題形式的に制約をかけることにしました。

```
[CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
```

(https://camo.qiitausercontent.com/cbfb2acc5e5e006f0afc9976bf6da430bab04baf/68747470733a2f2f71696974612d696d6167652d73746f72652e733332e616d617a6f6e6177732e636f6d2f302f313233353839 2f363436303931393962 d653836302d616436612d623035612d373931383535623632363134312e706e67)

MASKされた単語を予測するタスクにおきかえることで、前後の文脈を考慮するモデル(双方向)で学習させることができるようになりました。

**事前学習② 隣接文の予測**

二つの文章を与え、隣り合っているかをYes/Noで判定します。
文章AとBが与えられた時に、50%の確率で別の文章Bに置き換えます。

<モチベーション>
QAや自然言語推論では２つの文章の関係性を理解できることが重要です。
しかし、文章同士の関係性は単語の共起性から単語の発生確率をモデリングしている言語モデルのみではとらえきれない特徴量を含んでいると考えられます。
隣接文予測を解かせることで、文章の単位での意味表現を獲得できます。
word2vecが単語表現モデルであったのに対し、BERTはより広範的な言語表現モデルとして機能できる理由のひとつです。

## BERTの弱点

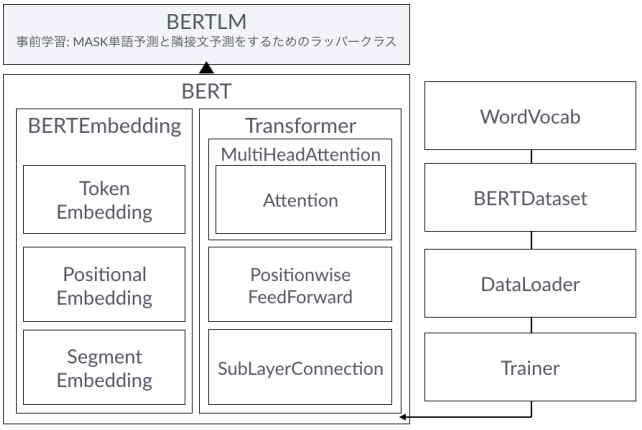
BERTは学習に時間がかかります。これは学習の際に予測するべき単語が学習データの中で15%しか存在しないためです。
一方で、従来の言語モデルでは毎単語を予測対象としていたので、学習のペースに大きな差があります。

ちなみにGoogleはBaseとLargeという二種類のモデルを用意しています。後者のLargeはGoogleの本気モードです。お高くつきます。
Baseの学習：4TPUS(16TPUチップ) x 4日
Largeの学習：16TPUS(64TPUチップ) x 4日
同様の学習をGPUで行うと40~70日かかるとか・・・

## 実装

今回は日本語版BERT実装するにあたって新規性のある部分だけピックアップして解説しようと思います。
コードはこちら (https://github.com/Kosuke-Szk/ja_text_bert)にjupyter notebook形式で公開してありますのでご参照ください。

jupyter notebookに全部載せているので構造がわかりにくくなっていますが、BERTの全体構成は以下のようになっています。




(https://camo.qiitausercontent.com/348980102b722b9ab05ed175aa63f452af8ee1b0/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f313232333538392f663035323433866642d663737662d613164652d383336392d3036643363313735646139362e706e67)

### データセットの準備

日本語Wikiからコーパスを作成します。今回は100Mbyte程度のコーパスを作成します。
基本的に `text_preprocess.ipynb` のコードを上から実行しけばデータセットが完成します。

**コーパス作成の流れ**

1. コーパスを作るもとになるWikipediaの日本語ダンプデータをダウンロードします
2. Wikiextractor (https://github.com/attardi/wikiextractor)で不要なマークアップを取り除きます
3. htmlタグ削除、空白・改行削除、大文字小文字変換などの前処理を施します
4. 単語単位で文章を分割します
5. 使いたいデータ量分だけ切り出す

6. Wikipediaの一項目の文書が、偶数個の文章から成るようにする(BERTのNext Sentence Prediction)のため)

## 事前学習

#### 隣接文予測

どちらもBERTからはきだされた内部状態テンソルをInputとして一層のMLPでクラス分類しているだけです。シンプルですね。
これとほぼ同じようにFine Tuningの際の追加層もくっつけられるってことなんですね。蛇口の出口にアダプター取り付けるだけであらゆるタスクに対応できる便利さ・・・。

NextSentencePrediction

```
class NextSentencePrediction(nn.Module):
    """
    2クラス分類問題 : is_next, is_not_next
    """

    def __init__(self, hidden):
        """
        :param hidden: BERT model output size
        """
        super().__init__()
        self.linear = nn.Linear(hidden, 2)
        self.softmax = nn.LogSoftmax(dim=-1)

    def forward(self, x):
        return self.softmax(self.linear(x[:, 0]))
```

#### マスク単語予測

MaskedLanguageModel

```
class MaskedLanguageModel(nn.Module):
    """
    入力系列のMASKトークンから元の単語を予測する
    nクラス分類問題, nクラス : vocab_size
    """

    def __init__(self, hidden, vocab_size):
        """
        :param hidden: output size of BERT model
        :param vocab_size: total vocab size
        """
        super().__init__()
        self.linear = nn.Linear(hidden, vocab_size)
        self.softmax = nn.LogSoftmax(dim=-1)

    def forward(self, x):
        return self.softmax(self.linear(x))
```

#### 実際にやってみると

Googleが実際にやった規模で動かすのはコスト高なので、実験的に100M程度のデータセットで事前学習をさせてみました。
経験則ですが、データセットが小さい場合は、モデルサイズも小さくしないとうまく動かないようです。(Lossが減少しても、Next Sentence PredictionのAccが向上しません。)
実はこのトピックは、BERT-pytorchのgithubリポジトリのIssueで最も盛り上がっているものの一つです。データサイズやハイパーパラメーター設定によってAccの向上スピードが全く異なってきます。手元の実験の際には、adam_weight_decayを0、dropoutを0にしてトレーニングデータにフィットしやすい状態にしました。
参考までに、こちらでAccが向上したときのハイパーパラメータは以下の通りです。

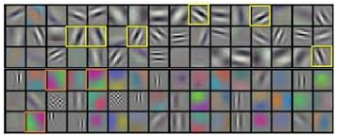
```
hidden = 256 # originally 768
layers = 8 # originally 12
attn_heads = 8 # originally 12
seq_len = 60
batch_size = 64
epochs = 10
num_workers = 5
with_cuda=True
log_freq = 20
corpus_lines = None
lr = 1e-3
adam_weight_decay = 0.00 # originally 0.01
adam_beta1 = 0.9
adam_beta2 = 0.999
dropout = 0.0 # originally 0.1
min_freq = 7 # これでvocabサイズを50,000程度に抑えた
```

# まとめ

日本語コーパスでBERT Pre-Trainedモデルを作成する方法を解説しました。
とにかく時間がかかります。GPUを使ってもなかなかAccuracyが向上しません。
どの程度の学習状況で、事前学習モデルとして性能を発揮するかは、実際に動かしてみるまで未知数ですね。
(この記事を書いている間にGoogleからPre-Trainedモデルが公開されました。リンク (https://github.com/google-research/bert))
ということで、次回以降はBERTモデルを実際の言語タスクに適用するFine Tuning以降を実装していこうと思います。

## 考察

BERTは深層自然言語処理のひとつの大きな転換点になるのでは、と思っています。ポイントは**「今後はネットワーク(脳)のチューニングから、学習タスク(教育)のチューニングに重きが移りそう」**ということです。
画像領域での深層学習の快進撃には目をみはるものがあり、もはや人間にできないことまでやり遂げています。一方で言語領域では根本的なブレイクスルーがなかなか起こらず、機械翻訳の発展などがありつつも、全体的にはいまひとつというラインで留まっていました。
CNNを筆頭とする画像分野での深層学習の革新の背景には、アルゴリズムを自然画像に適用すると、エッジまたは特定の色のエッジを検出する特徴量を人間のように学習することだったのではないかと考えています。下の画像のような特徴検出は、人間の一次視覚野に存在することが知られているガボール関数 (http://zellij.hatenablog.com/entry/20131003/p1)を想起させるものです。だからこそ、VGGやResNetのような大規模データで事前学習を行った画像モデルは、一般的な特徴量抽出器として性能を発揮するのでしょう。


(https://camo.qiitausercontent.com/c612b998afd4848581a3f620d077acb7df4207a7/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3132333538392f323466333356631612d363632612d396139612d623164302d336637656535643326634652e706e67)

一方で、自然言語処理では上のように人間らしい認知回路はなかなか出てきていないように思います。それは言語というものが、離散的情報であり、シンボルを基軸とした論理や構造を含む情報だからなのかもしれません。結果的に、統計的言語処理を用いて言語タスクを解く際にはタスクに特化させてネットワークを組み、タスクに特化させて特徴量をヒューリスティックに設計するアプローチが取られてきました。いわば、タスクに合わせて脳のネットワーク構造を改造するアプローチを取ってきたわけです。
深層自然言語処理以前の言語処理は、形態素解析、構文解析、意味解析、文脈解析など、人為的に言語論理を組み込んだフレームワークで処理を行ってきました。それが、深層学習によるEnd-to-Endの統計的言語処理により、単純なタスク解決においてはより効率的に解を示しました。しかし、タスクを変えてしまうと全く性能を発揮しなくなるフレーム問題は乗り越えられずにいました。
ここがBERTが転換点になると思っているポイントです。BERTはマスク単語予測と、隣接文予測というふたつのタスクを事前学習することにより、従来のタスク特化型のモデルに負けず、諸々のベースラインタスクで圧倒的な性能を示しました。憶測の域を出ませんが、より人間に近い形で、重層的な言語認知(注意)をしているのではないかと考えています。その中には構文解析的な認知状態や、まだ人間が定義できていない新しい認知状態も含まれるかもしれません。
以上が、今後は脳(ネットワーク)にどんな教育(事前学習タスク)を施すかの方法論に重きが移っていくのではないかと考えている理由です。
End-to-Endで自動的に汎用的な特徴量を抽出する表現学習を行えるとすると、過去に画像機械認識で起こった認知革命が言語にも起こるかもしれない！楽しみですね。

## 参考

・BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (https://arxiv.org/abs/1810.04805?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+arxiv%2FQSXk+%28ExcitingAds%21+cs+updates+on+arXiv.org%29)
・Attention is all you need (https://arxiv.org/abs/1706.03762)
・論文解説 Attention Is All You Need (Transformer) (http://deeplearning.hatenablog.com/entry/transformer)
・BERT-pytorch (https://github.com/codertimo/BERT-pytorch)
・日本語版text8コーパスを作って分散表現を学習する (https://hironsan.hatenablog.com/entry/japanese-text8-corpus)

## プロジェクト紹介

本記事の内容は、株式会社サイシード (https://www.wantedly.com/projects/251561)のシンボルグラウンディングプロジェクトの一貫として進めています。
このプロジェクトは、コンピューターサイエンスだけでなく、言語学、心理学、脳科学などの多角的な切り口から、実世界と記号空間のシステマティックな接続を実現する野心的な試みです。
サイシードではシンボルグラウンディングプロジェクトに一緒に取り組んでくれる熱意ある仲間を募集しています！



**Kosuke Suzuki (/Kosuke-Szk) @Kosuke-Szk (/Kosuke-Szk)** (/organizations/sciseed)
専攻は計算物理(プラズマ物理)です。 趣味と仕事でNLPもやります。
(/Kosuke-Szk)








この記事は以下の記事からリンクされています

過去の1件を表示する
**NLPのタスクを紹介するだけの簡単なページ(0) (/MeguruMokke/items/1201b5fa05ad6ec5efb9#_reference-736cade749a99cbaa65c)** からリンク 8ヶ月前
**BERT多言語モデルで日本語文章の二値分類を試す (/knok/items/9e3b4505d6b8f813943d#_reference-b3f5094cf0dfb2266433)** からリンク 7ヶ月前
**汎用言語表現モデルBERTの内部動作を解明してみる (/Kosuke-Szk/items/d49e2127bf95a1a8e19f#_reference-f84364e2458ca4273c57)** からリンク 7ヶ月前
**機械学習/ディープラーニング初心者が2018年にやったこと、読んだ論文 (/koshian2/items/bb4afec12e5dd50aae50#_reference-8a2e1f2de6305a36819d)** からリンク 6ヶ月前
**2018年〜19年のAIトレンドまとめ(Medium記事より) (/zumitan/items/5ca573b11ea1882420fb#_reference-b1347470b58c9d81865b)** からリンク 4ヶ月前

**tmitani (/tmitani)**
(/tmitani) 44 contribution
2018-11-06 00:42
いいね 0

丁寧な解説ありがとうございます。

self-attentionのところの図ですが、
q1*k1 q2*k2 q3*k3
となっている下から3層目のブロックは、

q1に対するattentionなので
q1*k1 q1*k2 q1*k3
な気がします。

**tmitani (/tmitani)** (/tmitani) 44 contribution — 2018-11-06 00:59 — いいね 0

あと、1点質問させてください。初歩的な質問で申し訳ありませんが、
token embeddingは、
nn.Embeddingsを継承しているので、ランダムに初期化されたのち、
このvocab_size*embed_size個のパラメータも、
訓練の際に学習されるという理解で正しいでしょうか。

**Kosuke-Szk (/Kosuke-Szk)** (/Kosuke-Szk) 1209 contribution — 2018-11-06 10:58 — いいね 0

図のご指摘ありがとうございます
修正させていただきました！

**Kosuke-Szk (/Kosuke-Szk)** (/Kosuke-Szk) 1209 contribution — 2018-11-06 11:01 — いいね 0

token embeddingのパラメータ更新に関するご理解もそれで間違い無いです。

ほぼ何もoverrideしていないので、通常のembeddingの働きと同様ですね。

**tmitani (/tmitani)** (/tmitani) 44 contribution — 2018-11-06 12:19 — いいね 0

ありがとうございます！

**Shunichirou (/Shunichirou)** (/Shunichirou) 763 contribution — 2018-11-07 23:43 — いいね 0

理解が合ってるか、教示ください。

モデルの変化

### 従来

n+1単語目は、n単語目までの単語から、次にくる単語を予想する、言わば前方向の予想（学習）しかできなかった。そのため、学習スピードは早くとも精度はイマイチ。

### BERT

文書からある単語をマスクし、前後関係からどういった単語が入るかを当てる、言わば双方向の予想（学習）ができるようになった。ただし学習が遅い。

上記理解があっている上で、以下質問させてください。

- BERTの学習では、15パーセントをマスクとして、その周辺の単語からマスク内容を当てることを「学習」と定義したと認識しました。その15パーセントを、さらに以下のように区分けしている理由を教えてください。（文中に述べられてなかったので...）

> そのうち、80%がマスク、10%がランダムな単語、10%を置き換えない方針で変換します。

## 以下所感

非常に読み応えがありました。おもしろかったです。記事にて述べているとおり、分散情報を学習させ、いかに人のように文書を理解させることができるかというのは、やはり難しい論点だと感じておりました。そんな中、学習モデルとしてBERTが出現したことは衝撃だと思います。

「ある単語と関係のある単語」を検索する際には、常に進化・変化し続ける「名詞」、「固有名詞」を常に学習させる必要があります。Googleのタスクにおけるニュース記事類似判定などはその良い例で、タスクとして多くの近代ニュースを教師データとしたのかと思われます。そのような、常に学習を繰り返すことが必須な中で、これまでは「どのようなモデルを使うか」という点が論点でしたが、「学習時のチューニングをいかにするか」が論点になり、今後が楽しみです。

長々と失礼いたしました。

**Kosuke-Szk (/Kosuke-Szk)** (/Kosuke-Szk) 1209 contribution — 2018-11-08 14:54 — いいね 1

ご質問ありがとうございます。

> 80%がマスク、10%がランダムな単語、10%を置き換えない方針で変換します。

これはGoogleが実際に取った方針で、論文中の記載をそのまま実装しております。
Googleがなぜこのようなmaskingの方針に落ち着いたのかの真意は正直私もわかりません。
論文中の解説によると、「TransformerのEncoderはどの単語が予測すべきマスクで、どの単語が別単語に置き換えられているのか知らないため、全ての入力単語に対して分散的な文脈表現を保持することを要請される」と説明されています。
その直後に「15% x 10%、つまり全体の1.5%の単語をランダムに置き換えても、文章データに影響を及ぼすことはないだろう。」とも記載されています。

この処理のコードの中での挙動は
・15%の確率でマークする単語を決める
・マークされた単語のIDをメモしておく
・マークされた単語のうち80%をMASKに置き換える、10%を別単語に置き換える、10%はそのままにする
・訓練の時は「15%の確率でマークされた単語ID」を教師として、それに対する予測結果との差分をLossとする
となっています。

すなわち、Lossの設計的には80%→10%の確率でそのままにされた(変化していない)単語に関しても、マークされていたと予測することを求めているわけですが、ちょっとそれは無理がある気もします。10%の確率で単語をそのままに放置した意図は不明です。

以上、参考になっていれば幸いです。記事を読んでいただいてありがとうございます。

**Tatsunaga (/Tatsunaga)**
(/Tatsunaga) 1 contribution
2018-12-26 16:25
いいね 0

投稿ありがとうございます。
只今コードを拝読し、私の環境上で実行しておりますが、
wikipedia全データで学習しようとするとmemory overになってしまいます。
御自身の実行環境(計算リソースなど)、御伺いしたいです。
宜しくお願い致します。

**Kosuke-Szk (/Kosuke-Szk)**
(/Kosuke-Szk) 1209 contribution
2018-12-29 13:49
いいね 1

ご返信遅くなりすみません。
こちらの計算環境としては、
- Tesla P100 GPU x ４枚
- メモリ100GB
でした。
バッチサイズでメモリに載せるデータ量は調整できるので、そちらを調整して検証していただければと思います。ちなみに私の環境でもバッチサイズはかなり低めに抑えないとmemory overになってしまった記憶があります。
また何かあればお気軽にご質問ください。

**shionhonda (/shionhonda)**
(/shionhonda) 1588 contribution
2019-01-11 16:39
いいね 1

コーパスの前処理含め事前学習の方法についてコード付きで解説していただき, 非常に助かりました. ありがとうございます.

私はこちらの記事と下のレポジトリを参考に, メモリ12GBというより厳しい制約のもとでミニBERTを事前学習しました.
ハイパーパラメータが気になっている方の参考になるかと思い, NSPのAccuracy上昇を確認できた時の設定をGitHubにコメントしました(英語コーパスを使っています)
https://github.com/codertimo/BERT-pytorch/issues/32#issuecomment-453382337 (https://github.com/codertimo/BERT-pytorch/issues/32#issuecomment-453382337)