

自己紹介

小川 晃

所属: UNICORN株式会社 エンジニア Div.

専攻分野: 自然言語処理(情報学修士修了)

主な業務内容(一例):

インターネット広告にて入札するキーワードの自動選定

BERT fine-tuning時 (固有表現抽出)

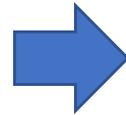
- 最大文長: 256
- 学習率: 5e-5
- 訓練エポック数: 16 epochs
- バッチサイズ: 32 (訓練), 64 (テスト)
- オプティマイザー: AdamW
- 重複範囲: 32

(その他の条件はデフォルトと同一)

試行1. 最大文長, バッチサイズなどを上げた

デフォルト

- 最大文長: 128
- 学習率: $5e-5$
- 訓練エポック数: 5
- 訓練バッチサイズ: 4



試行1

- 最大文長: 256
- 学習率: $5e-5$
- 訓練エポック数: 16
- 訓練バッチサイズ: 32

スコア: 85.9

スコア: 87.2

スコアが**向上**

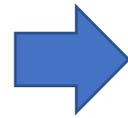
↑最大文長を学習可能なレベルで大きくしたことが
パフォーマンス向上に寄与したと推測

試行2. 最適化関数を変更し、重複範囲を拡大した

試行1

- 最適化関数: Adam
- 重複範囲: 16

スコア: 87.2



試行2

- 最適化関数: AdamW
- 重複範囲: 32

スコア: 87.5

スコアがわずかに**向上**

↑Adamの改良版であるAdamW & 重複範囲の拡大により文脈を捉えやすくなったことがパフォーマンス向上に寄与したと推測

試行 2.5. 末尾にさらに線形層を追加した

試行2

```
# BERTの埋め込みをIOB2タグに変換するための線形層
self.layer = nn.Linear(config.hidden_size, num_labels)

def forward(self, input_ids, attention_mask=None, labels=None):
    outputs = self.bert(input_ids, attention_mask=attention_mask) # BERTによる計算

    seq_outputs = outputs[0] # 各トークンのベクトルのみ取り出す

# 全結合層による分類結果
logits = self.layer(seq_outputs) # 各トークンのベクトルを、IOB2タグに変換
```

試行2.5



スコア：87.5

```
# BERTの埋め込みをIOB2タグに変換するための線形層
self.layer_1 = nn.Linear(config.hidden_size, config.hidden_size)
self.layer_2 = nn.Linear(config.hidden_size, num_labels)

def forward(self, input_ids, attention_mask=None, labels=None):
    outputs = self.bert(input_ids, attention_mask=attention_mask) # BERTによる計算

    seq_outputs = outputs[0] # 各トークンのベクトルのみ取り出す

# 全結合層による分類結果
logits = self.layer_1(seq_outputs)
logits = self.layer_2(logits) # 各トークンのベクトルを、IOB2タグに変換
```

スコア：87.1

スコアが低下

↑モデルが複雑になったため、
パラメータのより細かな調整が
必要と推測
&(線形層間で)dropout層の追加が
有効か？

1. **最大文長&エポック数の増加が、**
BERTのパフォーマンス向上につながった
2. **最適化関数の改善&サンプル間の重複範囲の拡大も、**
パフォーマンス向上につながった