ADL21 HW2 - M10915106

Q1 - Data processing

1. Tokenizer

a. Describe in detail about the tokenization algorithm you use. You need to explain what it does in your own ways.

在 Multiple Choice 中，將每一筆 question 弄成四筆相通的作為 tokenizer 的 text，將該筆資料的四個 paragraph 做成相對應的 text pair，再透過 tokenizer 將輸入轉成 BatchEncoding。其中設定 tokenizer 做 padding, truncation, max_length 設為 512。

在 Question Answering 中，在資料讀入後，會透過 relevant 把正確的 context 讀出，方便處理。把 question 做成 tokenizer 的 text，將該正確 context 做成相對應的 text pair，再透過 tokenizer 將輸入轉成 BatchEncoding。其中設定 tokenizer 做 padding, truncation, max_length 設為 512 之外，還有另外設 return_offsets_mapping, return_overflowing_tokens 為 True，以及 doc_stride = 192。因為在較長的 sequence 下，長度可能大於 max_length 512，context 就會被分長兩段，透過 doc_stride 來移動 context 視窗，斷句保留的文字會較完整且多。
為了要讓較長的 sequence 可以 mapping 回去，所以透過 overflow_to_sample_mapping 來達到。
offset_mapping 可以幫助我們將 character 位置 mapping 到 token 位置。\

遍歷 offset_mapping 時，因為長度超過 max_length 的問題，並不會每個 context span 都會有答案，可以透過 overflow_to_sample_mapping 來得知，如果沒有答案就將 start end position 填上 cls_index。 另外也處理 token 中的答案範圍超過實際 answer 範圍的情況，如果發生也將 start end position 填上 cls_index。都沒有的話就把 token 的答案移到正確答案的兩端。

2. Answer Span

a. How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?

在 tokenizer 設定 return_offsets_mapping, return_overflowing_tokens 為 True。藉由 return_offsets_mapping 可以幫助我們從 character 轉到 token 的 start end。
因為文字的 start end 轉成 token 後可能不一樣，舉例來說假設 applied deep learning is awesome. 的答案為 deep learning 那以文字的視角 deep learning 的 start=9 end=21，但是轉 token 時 deep 的位置是 1 learning 的位置是 2，我們沒辦法很好的轉換，這時候 offset_mapping 就能夠很好的幫我們解決，他會給定 token 在原本 character 的起始和結束位置。這樣我們就能很方便地把 token 和 character mapping。

較長的 sequence 會被切成多份，透過 return_overflowing_tokens 可以將 sequence mapping 回去。再額外做一些 start end position 的判斷就可以完成。

b. After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?

在模型輸出時，需要判斷 start end 不合理的狀況，以及不同的 start end 組合。將 start 和 end position 的分數由高到低排序後，找出兩兩配對最好的前幾名，並過濾掉長度不合理或是 end < start 的情況。

Q2 - Modeling with BERTs and their variants

1. Describe

a. your model (configuration of the transformer model)

Multiple Choice

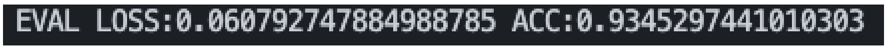
```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "directionality": "bidi",
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

Questoin Answering

```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "directionality": "bidi",
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

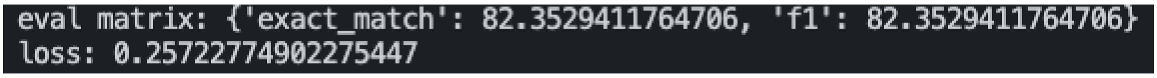
b. performance of your model.

Multiple Choice



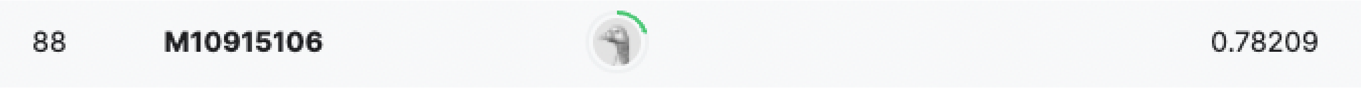
```
EVAL LOSS:0.060792747884988785 ACC:0.9345297441010303
```

Questoin Answering



```
eval matrix: {'exact_match': 82.3529411764706, 'f1': 82.3529411764706}
loss: 0.25722774902275447
```

Kaggle LeaderBoard



88	M10915106	0.78209

c. the loss function you used.

Multiple Choice
使用 Cross Entropy Loss。

Questoin Answering
使用 Cross Entropy Loss。

d. The optimization algorithm (e.g. Adam), learning rate and batch size.

Multiple Choice
Optimization 使用 AdamW
learning rate = 3e-5
batch size = 2
accumulation step = 4

Scheduler 使用 get_cosine_schedule_with_warmup
total_step = len(train_dataset) * num_epoch // (batch_size * accumlation steps)
warmup_step = total_step * 0.06

Questoin Answering
Optimization 使用 AdamW
learning rate = 3e-5
batch size = 4
accumulation step = 4

Scheduler 使用 get_cosine_schedule_with_warmup
total step = len(train_dataset) * num_epoch // (batch_size * accumlation step)
warmup step = total_step * 0.06

2. Try another type of pretrained model and describe

a. your model

Multiple Choice

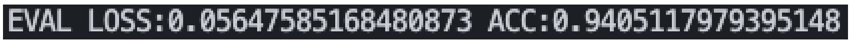
```
{
  "_name_or_path": "hfl/chinese-macbert-base",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

Questoin Answering

```
{
  "_name_or_path": "hfl/chinese-macbert-base",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

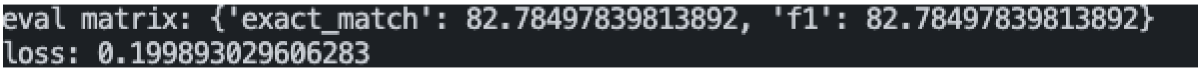
b. performance of your model

Multiple Choice



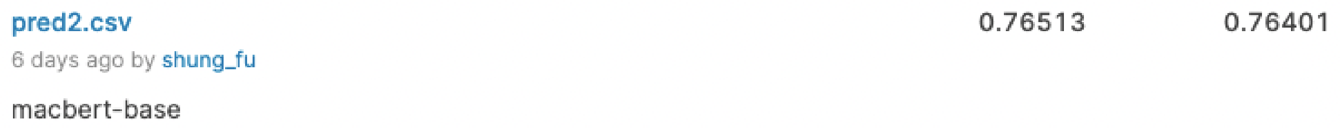
```
EVAL LOSS:0.05647585168480873 ACC:0.9405117979395148
```

Questoin Answering



```
eval matrix: {'exact_match': 82.78497839813892, 'f1': 82.78497839813892}
loss: 0.199893029606283
```

Kaggle



pred2.csv 0.76513 0.76401

6 days ago by shung_fu

macbert-base

c. the difference between pretrained model (architecture, pretraining loss, etc.)

chinese-roberta-wwm-ext

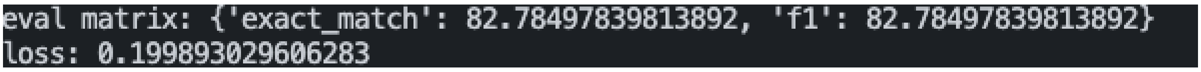
改進 BERT 模型，去除 NSP 預訓練任務，將 MLM 改進為動態 Masking。 使用 Whole Word Masking。

chinese-macbert-base

使用 Whole Word Masking、N-gram Masking：single token、2-gram、3-gram、4-gram。

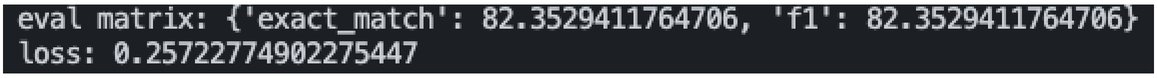
QA pretraining loss

chinese-macbert-base:



```
eval matrix: {'exact_match': 82.78497839813892, 'f1': 82.78497839813892}
loss: 0.199893029606283
```

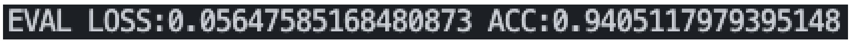
chinese-roberta-wwm-ext:



```
 eval matrix: {'exact_match': 82.3529411764706, 'f1': 82.3529411764706}
 loss: 0.25722774902275447
```

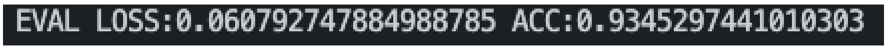
MC pretraining loss

chinese-macbert-base:



```
EVAL LOSS:0.05647585168480873 ACC:0.9405117979395148
```

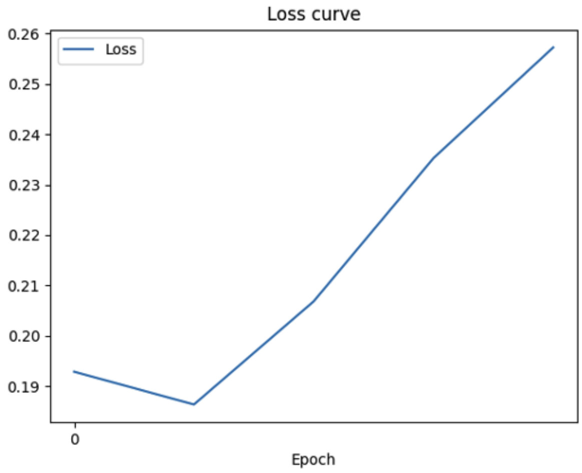
chinese-roberta-wwm-ext:



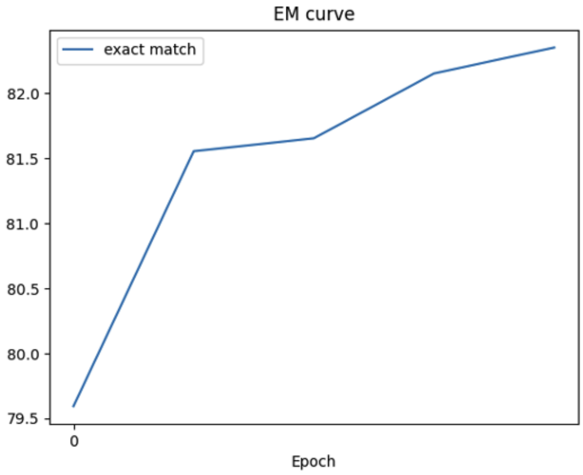
```
 EVAL LOSS:0.06079274788498785 ACC:0.9345297441010303
```

Q3 - Curves

Plot the learning curve of your QA model



Loss curve



EM curve

Q4 - Pretrained vs Not Pretrained

The configuration of the model and how do you train this model

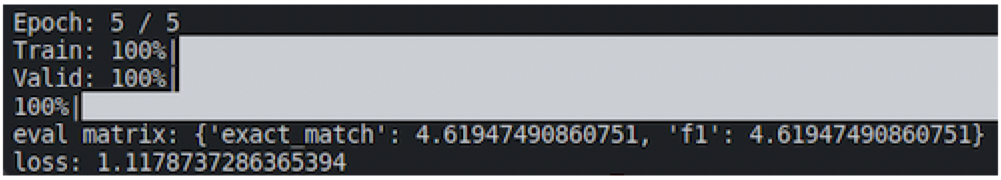
讀取模型時不使用 from_pretrained 而是直接 from_config。其餘都與上述訓練方式一樣。這邊選用 QA。

```
config = AutoConfig.from_pretrained(args.model_name_or_path)
model = AutoModelForQuestionAnswering.from_config(config)
model.resize_token_embeddings(len(tokenizer))
model.to(args.device)
```

```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "directionality": "bidi",
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

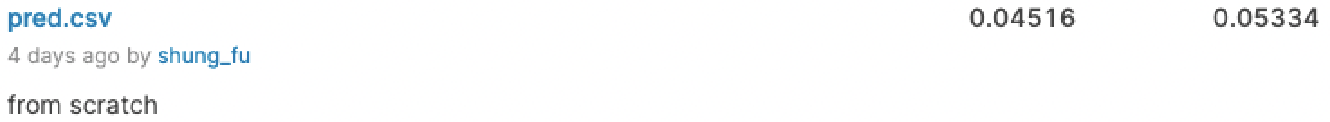
the performance of this model v.s. BERT

this model:



```
Epoch: 5 / 5
Train: 100%|
Valid: 100%|
100%|
eval matrix: {'exact_match': 4.61947490860751, 'f1': 4.61947490860751}
loss: 1.1178737286365394
```

Kaggle LeaderBoard:

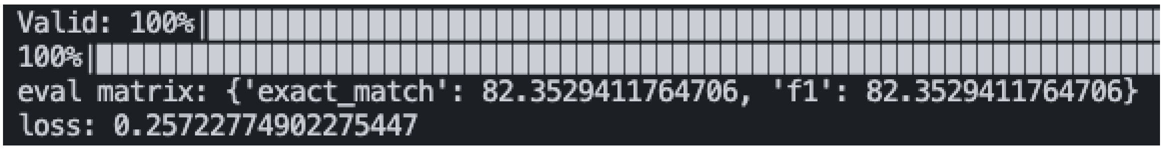


pred.csv 0.04516 0.05334

4 days ago by shung_fu

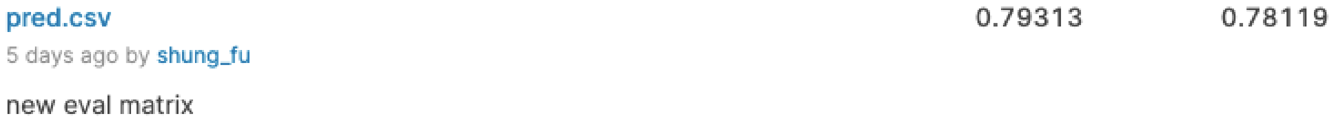
from scratch

BERT:



```
Valid: 100%|██████████████████████████████████████████████████████████████
100%|█████████████████████████████████████████████████████████████████████
eval matrix: {'exact_match': 82.3529411764706, 'f1': 82.3529411764706}
loss: 0.25722774902275447
```

Kaggle LeaderBoard:



pred.csv 0.79313 0.78119

5 days ago by shung_fu

new eval matrix

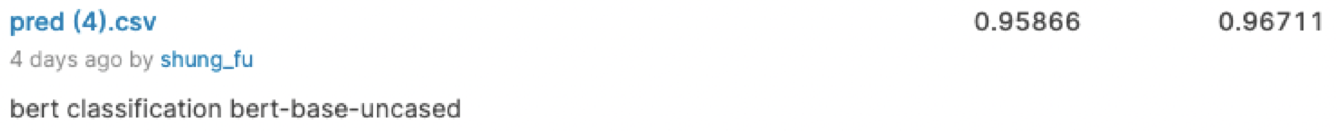
Q5: Bonus: HW1 with BERTs

a. your model

因為 json 較長所以放在最後面。

b. performance of your model.

b-1. Intent classification

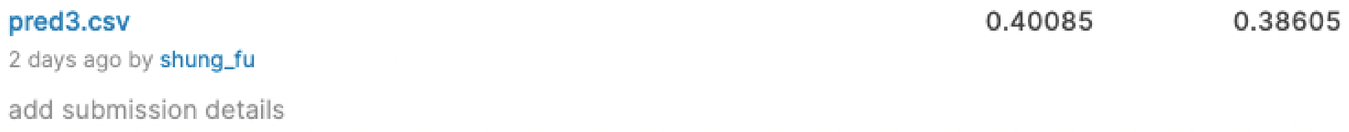


pred (4).csv 0.95866 0.96711

4 days ago by shung_fu

bert classification bert-base-uncased

b-2. Slot tagging (1%)



pred3.csv 0.40085 0.38605

2 days ago by shung_fu

add submission details

c. the loss function you used.

Intent Classification
使用 Cross Entropy Loss。

Slot Tagging
使用 Cross Entropy Loss。

d. The optimization algorithm (e.g. Adam), learning rate and batch size.

Intent Classification
Optimization 使用 AdamW
learning rate = 1e-4
batch size = 4
accumulation step = 16

Scheduler 使用 get_cosine_schedule_with_warmup
total_step = len(train_dataset) * num_epoch // (batch_size * accumlation steps)
warmup_step = total_step * 0.12

Slot Tagging
Optimization 使用 AdamW
learning rate = 3e-5
batch size = 16
accumulation step = 1

Scheduler 使用 get_linear_schedule_with_warmup
total_step = len(train_dataset) * num_epoch // (batch_size * accumlation steps)
warmup_step = total_step * 0.06

a. your model

Slot Tagging

```
{
  "_name_or_path": "bert-base-cased",
  "architectures": [
    "BertForTokenClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4",
    "5": "LABEL_5",
    "6": "LABEL_6",
    "7": "LABEL_7",
    "8": "LABEL_8"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2,
    "LABEL_3": 3,
    "LABEL_4": 4,
    "LABEL_5": 5,
    "LABEL_6": 6,
    "LABEL_7": 7,
    "LABEL_8": 8
  },
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}
```

Intent Classification

```
{
  "_name_or_path": "bert-base-uncased",
  "architectures": [
    "BertForSequenceClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4",
    "5": "LABEL_5",
    "6": "LABEL_6",
    "7": "LABEL_7",
    "8": "LABEL_8",
    "9": "LABEL_9",
    "10": "LABEL_10",
    "11": "LABEL_11",
    "12": "LABEL_12",
    "13": "LABEL_13",
    "14": "LABEL_14",
    "15": "LABEL_15",
    "16": "LABEL_16",
    "17": "LABEL_17",
    "18": "LABEL_18",
    "19": "LABEL_19",
    "20": "LABEL_20",
    "21": "LABEL_21",
    "22": "LABEL_22",
    "23": "LABEL_23",
    "24": "LABEL_24",
    "25": "LABEL_25",
    "26": "LABEL_26",
    "27": "LABEL_27",
    "28": "LABEL_28",
    "29": "LABEL_29",
    "30": "LABEL_30",
    "31": "LABEL_31",
    "32": "LABEL_32",
    "33": "LABEL_33",
    "34": "LABEL_34",
    "35": "LABEL_35",
    "36": "LABEL_36",
    "37": "LABEL_37",
    "38": "LABEL_38",
    "39": "LABEL_39",
    "40": "LABEL_40",
    "41": "LABEL_41",
    "42": "LABEL_42",
    "43": "LABEL_43",
    "44": "LABEL_44",
    "45": "LABEL_45",
    "46": "LABEL_46",
    "47": "LABEL_47",
    "48": "LABEL_48",
    "49": "LABEL_49",
    "50": "LABEL_50",
    "51": "LABEL_51",
    "52": "LABEL_52",
    "53": "LABEL_53",
    "54": "LABEL_54",
    "55": "LABEL_55",
    "56": "LABEL_56",
    "57": "LABEL_57",
    "58": "LABEL_58",
    "59": "LABEL_59",
    "60": "LABEL_60",
    "61": "LABEL_61",
    "62": "LABEL_62",
    "63": "LABEL_63",
    "64": "LABEL_64",
    "65": "LABEL_65",
    "66": "LABEL_66",
    "67": "LABEL_67",
    "68": "LABEL_68",
    "69": "LABEL_69",
    "70": "LABEL_70",
    "71": "LABEL_71",
    "72": "LABEL_72",
    "73": "LABEL_73",
    "74": "LABEL_74",
    "75": "LABEL_75",
    "76": "LABEL_76",
    "77": "LABEL_77",
    "78": "LABEL_78",
    "79": "LABEL_79",
    "80": "LABEL_80",
    "81": "LABEL_81",
    "82": "LABEL_82",
    "83": "LABEL_83",
    "84": "LABEL_84",
    "85": "LABEL_85",
    "86": "LABEL_86",
    "87": "LABEL_87",
    "88": "LABEL_88",
    "89": "LABEL_89",
    "90": "LABEL_90",
    "91": "LABEL_91",
    "92": "LABEL_92",
    "93": "LABEL_93",
    "94": "LABEL_94",
    "95": "LABEL_95",
    "96": "LABEL_96",
    "97": "LABEL_97",
    "98": "LABEL_98",
    "99": "LABEL_99",
    "100": "LABEL_100",
    "101": "LABEL_101",
    "102": "LABEL_102",
    "103": "LABEL_103",
    "104": "LABEL_104",
    "105": "LABEL_105",
    "106": "LABEL_106",
    "107": "LABEL_107",
    "108": "LABEL_108",
    "109": "LABEL_109",
    "110": "LABEL_110",
    "111": "LABEL_111",
    "112": "LABEL_112",
    "113": "LABEL_113",
    "114": "LABEL_114",
    "115": "LABEL_115",
    "116": "LABEL_116",
    "117": "LABEL_117",
    "118": "LABEL_118",
    "119": "LABEL_119",
    "120": "LABEL_120",
    "121": "LABEL_121",
    "122": "LABEL_122",
    "123": "LABEL_123",
    "124": "LABEL_124",
    "125": "LABEL_125",
    "126": "LABEL_126",
    "127": "LABEL_127",
    "128": "LABEL_128",
    "129": "LABEL_129",
    "130": "LABEL_130",
    "131": "LABEL_131",
    "132": "LABEL_132",
    "133": "LABEL_133",
    "134": "LABEL_134",
    "135": "LABEL_135",
    "136": "LABEL_136",
    "137": "LABEL_137",
    "138": "LABEL_138",
    "139": "LABEL_139",
    "140": "LABEL_140",
    "141": "LABEL_141",
    "142": "LABEL_142",
    "143": "LABEL_143",
    "144": "LABEL_144",
  "145": "LABEL_145",
  "146": "LABEL_146",
  "147": "LABEL_147",
  "148": "LABEL_148",
  "149": "LABEL_149"
},
"initializer_range": 0.02,
"intermediate_size": 3072,
"label2id": {
  "LABEL_0": 0,
  "LABEL_1": 1,
  "LABEL_10": 10,
  "LABEL_100": 100,
  "LABEL_101": 101,
  "LABEL_102": 102,
  "LABEL_103": 103,
  "LABEL_104": 104,
  "LABEL_105": 105,
  "LABEL_106": 106,
  "LABEL_107": 107,
  "LABEL_108": 108,
  "LABEL_109": 109,
  "LABEL_11": 11,
  "LABEL_110": 110,
  "LABEL_111": 111,
  "LABEL_112": 112,
  "LABEL_113": 113,
  "LABEL_114": 114,
  "LABEL_115": 115,
  "LABEL_116": 116,
  "LABEL_117": 117,
  "LABEL_118": 118,
  "LABEL_119": 119,
  "LABEL_12": 12,
  "LABEL_120": 120,
  "LABEL_121": 121,
  "LABEL_122": 122,
  "LABEL_123": 123,
  "LABEL_124": 124,
  "LABEL_125": 125,
  "LABEL_126": 126,
  "LABEL_127": 127,
  "LABEL_128": 128,
  "LABEL_129": 129,
  "LABEL_13": 13,
  "LABEL_130": 130,
  "LABEL_131": 131,
  "LABEL_132": 132,
  "LABEL_133": 133,
  "LABEL_134": 134,
  "LABEL_135": 135,
  "LABEL_136": 136,
  "LABEL_137": 137,
  "LABEL_138": 138,
  "LABEL_139": 139,
  "LABEL_14": 14,
  "LABEL_140": 140,
  "LABEL_141": 141,
  "LABEL_142": 142,
  "LABEL_143": 143,
  "LABEL_144": 144,
  "LABEL_145": 145,
  "LABEL_146": 146,
  "LABEL_147": 147,
  "LABEL_148": 148,
  "LABEL_149": 149,
  "LABEL_15": 15,
  "LABEL_16": 16,
  "LABEL_17": 17,
  "LABEL_18": 18,
  "LABEL_19": 19,
  "LABEL_2": 2,
  "LABEL_20": 20,
  "LABEL_21": 21,
  "LABEL_22": 22,
  "LABEL_23": 23,
  "LABEL_24": 24,
  "LABEL_25": 25,
  "LABEL_26": 26,
  "LABEL_27": 27,
  "LABEL_28": 28,
  "LABEL_29": 29,
  "LABEL_3": 3,
  "LABEL_30": 30,
  "LABEL_31": 31,
  "LABEL_32": 32,
  "LABEL_33": 33,
  "LABEL_34": 34,
  "LABEL_35": 35,
  "LABEL_36": 36,
  "LABEL_37": 37,
  "LABEL_38": 38,
  "LABEL_39": 39,
  "LABEL_4": 4,
  "LABEL_40": 40,
  "LABEL_41": 41,
  "LABEL_42": 42,
  "LABEL_43": 43,
  "LABEL_44": 44,
  "LABEL_45": 45,
  "LABEL_46": 46,
  "LABEL_47": 47,
  "LABEL_48": 48,
  "LABEL_49": 49,
  "LABEL_5": 5,
  "LABEL_50": 50,
  "LABEL_51": 51,
  "LABEL_52": 52,
"LABEL_53": 53,
"LABEL_54": 54,
"LABEL_55": 55,
"LABEL_56": 56,
"LABEL_57": 57,
"LABEL_58": 58,
"LABEL_59": 59,
"LABEL_6": 6,
"LABEL_60": 60,
"LABEL_61": 61,
"LABEL_62": 62,
"LABEL_63": 63,
"LABEL_64": 64,
"LABEL_65": 65,
"LABEL_66": 66,
"LABEL_67": 67,
"LABEL_68": 68,
"LABEL_69": 69,
"LABEL_7": 7,
"LABEL_70": 70,
"LABEL_71": 71,
"LABEL_72": 72,
"LABEL_73": 73,
"LABEL_74": 74,
"LABEL_75": 75,
"LABEL_76": 76,
"LABEL_77": 77,
"LABEL_78": 78,
"LABEL_79": 79,
"LABEL_8": 8,
"LABEL_80": 80,
"LABEL_81": 81,
"LABEL_82": 82,
"LABEL_83": 83,
"LABEL_84": 84,
"LABEL_85": 85,
"LABEL_86": 86,
"LABEL_87": 87,
"LABEL_88": 88,
"LABEL_89": 89,
"LABEL_9": 9,
"LABEL_90": 90,
"LABEL_91": 91,
"LABEL_92": 92,
"LABEL_93": 93,
"LABEL_94": 94,
"LABEL_95": 95,
"LABEL_96": 96,
"LABEL_97": 97,
"LABEL_98": 98,
"LABEL_99": 99
},
"layer_norm_eps": 1e-12,
"max_position_embeddings": 512,
"model_type": "bert",
"num_attention_heads": 12,
"num_hidden_layers": 12,
"pad_token_id": 0,
"position_embedding_type": "absolute",
"torch_dtype": "float32",
"transformers_version": "4.17.0",
"type_vocab_size": 2,
"use_cache": true,
"vocab_size": 30522
}
```