# Q1: Data processing

## Tokenizer

我使用的bert tokenizer是wordpiece，它會將資料集裡句子的每個字斷開，然後不斷地將unit合併在一起來最大提升跟train data的likelihood

兩種model的tokenizer都會將每個字都視為一個token，並且將原本為同一個字詞的字前面加上##，來表示這個字和前面的字為同一個詞

## Answer Span

**a.**

先把所有的句子切成token，如果切完的長度大於max_seq_length，則將其切割成不同的span跟在最後一個span做padding，並且將每個token在原文的起始和終點位置存到offset_mapping，之後一個一個掃過offset_mapping，把answer在原文的起始和終點位置轉成對應的offset index，也就是第幾個token，存到start_positions跟end_positions中。

**b.**

因為把所有token對應的文本位置存在offset_mapping裡，只要將model output中機率最大的一組start_positions跟end_positions轉換成offset_mapping中第start_positions個開頭跟第end_positions個結尾就可以得到answer text在文本的起始和終點位置

# Q2: Modeling with BERTs and their variants

## Main pretrained model

- **MC:** chinese-roberta-wwm-ext-large
  - **Loss function:** CrossEntropyLoss
  - **Optimization:** AdamW
  - **Learning Rate:** 3e-5
  - **Batch Size:** 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
- **QA:** chinese-roberta-wwm-ext-large
  - **Loss function:** CrossEntropyLoss
  - **Optimization:** AdamW
  - **Learning Rate:** 3e-5
  - **Batch Size:** 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)

## Other Pretrained model

除了bert-base-chinese跟chinese-roberta-wwm-ext-large之外，我也試了chinese-macbert-large在MC跟QA上

- **MC:** chinese-macbert-large
  - **Loss function:** Cross Entropy Loss
  - **Optimization:** AdamW
  - **Learning Rate:** 3e-5
  - **Batch Size:** 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
- **QA:** chinese-macbert-large
  - **Loss function:** Cross Entropy Loss
  - **Optimization:** AdamW
  - **Learning Rate:** 3e-5
  - **Batch Size:** 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)

| Model | BERT | Train Loss | Val Loss | Val EM |
|---|---|---|---|---|
| MC | chinese-roberta-wwm-ext-large | 0.32971993838378144 | 0.4042188227176666 | 0.9378530979156494 |
| MC | chinese-macbert-large | 0.562534138676957 | 0.385102778673172 | 0.9185776114463806 |
| QA | chinese-roberta-wwm-ext-large | 0.9830446687250011 | 0.9042038917541504 | 0.8321701561980724 |
| QA | chinese-macbert-large | 0.8147450521009988 | 0.6574541330337524 | 0.8295114656031905 |

| MC BERT | QA BERT | Public score |
|---|---|---|
| chinese-roberta-wwm-ext | chinese-macbert-large | 0.76943 |
| chinese-macbert-large | chinese-macbert-large | 0.76130 |
| chinese-roberta-wwm-ext | chinese-roberta-wwm-ext | 0.73779 |
| chinese-roberta-wwm-ext-large | chinese-roberta-wwm-ext-large | 0.78390 |

MacBert和一般Bert不同的是將原本需要被Mask的字詞用word2vec找出相似的同義字或用隨機詞彙取代

roberta-wwm-ext和一般roberta不同的則是當一個字被Mask掉時會連同將整個詞都Mask掉

# Configs:

## MC

### chinese-roberta-wwm-ext-large


```
{
  "_name_or_path": "./chinese-roberta-wwm-ext-large",
  "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": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "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
}
```

## chinese-macbert-large


```
{
  "_name_or_path": "./chinese-macbert-large",
  "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": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "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
}
```

# QA

## chinese-roberta-wwm-ext-large


```
{
  "_name_or_path": "./chinese-roberta-wwm-ext-large",
  "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": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "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
}
```

## chinese-macbert-large


```
{
  "_name_or_path": "./chinese-macbert-large",
  "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": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "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
}
```

# Q3: Curves

## chinese-roberta-wwm-ext-large

## chinese-macbert-large

這邊validation loss的算法是用model output兩個分別代表對start跟end的預測的array跟和train一樣求出start_positions跟end_positions各自轉成的one-hot vector算兩者cross entropy loss的平均

# Q4: Pretrained vs Not Pretrained

- **Task**: Multiple Choice
- **Pretrained Model**: chinese-macbert-large

from scratch Train的方法就跟之前的一樣，只不過把原本的

```
model = AutoModelForMultipleChoice.from_pretrained()
```

改成

```
model = AutoModelForMultipleChoice.from_config(config)
```

讓model不load pretrained weights

| Model | Train Loss | Val Loss | Val Acc |
| --- | --- | --- | --- |
| With Pretrained Weights | 0.562534138676957 | 0.385102778673172 | 0.9185776114463806 |
| Without Pretrained Weights | 0.9815234939422003 | 1.0061808824539185 | 0.5257560610771179 |

# Configs

## chinese-macbert-large


```
{
  "_name_or_path": "./chinese-macbert-large",
  "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": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "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
}
```

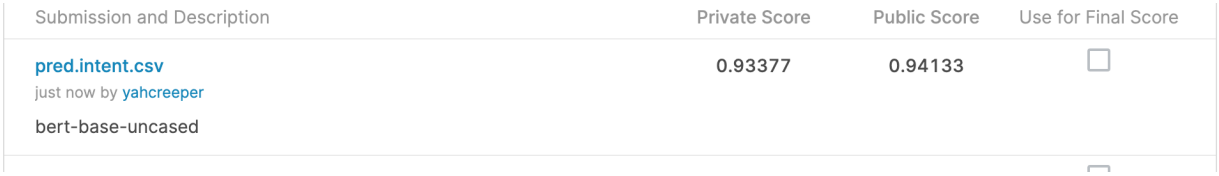
### without pretrained weights


```
{
  "_name_or_path": "./chinese-macbert-large",
  "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": 384,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 8,
  "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
}
```

# Q5: Bonus: HW1 with BERTs

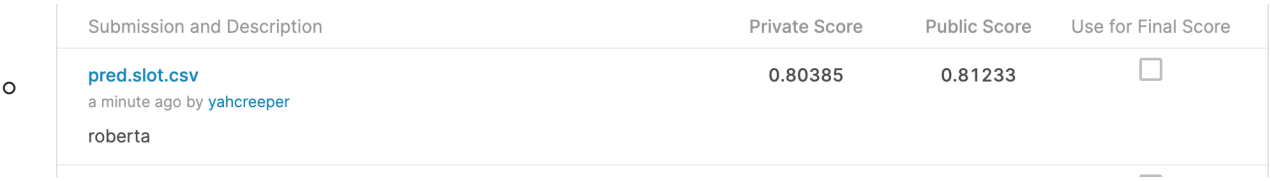
## Intent Classification

Code我參考網路上的script[https://www.xiuweihan.cn/2021/04/14/2021-04-14-bert%E4%B9%8B%E6%96%87%E6%9C%AC%E5%88%86%E7%B1%BB/]

- **Bert:** bert-base-uncased
- **Performance**
  - 
    
- **Loss function**: CrossEntropyError
- **Optimization:** AdamW
- **Learning Rate:** 2e-5
- **Batch Size:** 4
- **Epoch:** 1

## Slot Tagging

Code我參考huggingface的script[https://github.com/huggingface/transformers/blob/main/examples/pytorch/token-classification/run_ner.py]

- **Bert:** roberta-base
- **Performance**
  - 
    
- **Loss function**: CrossEntropyError
- **Optimization:** AdamW
- **Learning Rate:** 3e-5
- **Batch Size:** 2
- **Epoch:** 1

| Model | Task | Val Acc | Val loss | Public score | Private score |
|---|---|---|---|---|---|
| RNN | Intent Classification | 0.9343 | 0.455507 | 0.92133 | 0.92044 |
| BERT | Intent Classification | 0.9423 | 1.1190784533818563 | 0.94133 | 0.93377 |
| RNN | Slot Tagging | (Joint)0.791 | 1.648562 | 0.78498 | 0.78188 |
| BERT | Slot Tagging | 0.9670468948035488 | 0.12472361326217651 | 0.81233 | 0.80385 |

# Configs

## Intent Classification


```
{
  "_name_or_path": "./bert-base-uncased/pytorch_model.bin",
  "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",
  "problem_type": "single_label_classification",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```

## Slot Tagging


```
{
  "_name_or_path": "./roberta-base",
  "architectures": [
    "RobertaForTokenClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "eos_token_id": 2,
  "finetuning_task": "ner",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "B-date",
    "1": "B-first_name",
    "2": "B-last_name",
    "3": "B-people",
    "4": "B-time",
    "5": "I-date",
    "6": "I-people",
    "7": "I-time",
    "8": "O"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "B-date": 0,
    "B-first_name": 1,
    "B-last_name": 2,
    "B-people": 3,
    "B-time": 4,
    "I-date": 5,
    "I-people": 6,
    "I-time": 7,
    "O": 8
  },
  "layer_norm_eps": 1e-05,
  "max_position_embeddings": 514,
  "model_type": "roberta",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
```

```
  "type_vocab_size": 1,
  "use_cache": true,
  "vocab_size": 50265
}
```