Keyphrase Extraction using BERT Embeddings Method

```
!pip install transformers torch seqeval sklearn datasets
!pip install seqeval
!pip install datasets
!pip install transformers
```

```
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.10.10)
Requirement already satisfied: huggingface-hub>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.26.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (2.4.
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: yarl<2.0,>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.17.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.23
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datas
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->dataset
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from yarl<2.0,>=1.12.0->aiohttp->dat
Downloading datasets-3.1.0-py3-none-any.whl (480 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 480.6/480.6 kB 13.2 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 11.7 MB/s eta 0:00:00
Downloading fsspec-2024.9.0-py3-none-any.whl (179 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.3/179.3 kB 17.6 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 13.8 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 18.3 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2024.10.0
    Uninstalling fsspec-2024.10.0:
      Successfully uninstalled fsspec-2024.10.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the
gcsfs 2024.10.0 requires fsspec==2024.10.0, but you have fsspec 2024.9.0 which is incompatible.
Successfully installed datasets-3.1.0 dill-0.3.8 fsspec-2024.9.0 multiprocess-0.70.16 xxhash-3.5.0
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.46.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.26
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.9.11)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: tokenizers<0.21,>=0.20 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.6)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.2->t
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.2.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024
```

```
import os
import numpy as np
from datasets import load_dataset, Dataset
from transformers import AutoTokenizer, AutoModelForTokenClassification, TrainingArguments, Trainer
from transformers import DataCollatorForTokenClassification
from collections import Counter
import re
import torch
from torch.nn import CrossEntropyLoss

# Download data
dataset = load_dataset("midas/kp20k", split="train").select(range(5000))  # استخدام 5000 عينة

# Convert BIO labels to numbers
BIO_MAP = {"O": 0, "B": 1, "I": 2}

# Text cleaning function
def clean_text(text):
```

```
    if isinstance(text, list):  # If the text is a list of words
        text = " ".join(text)
    # Remove symbols other than letters
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    # Divide texts into words
    tokens = text.split()
    return tokens

# Text processing and BIO nomenclature
def preprocess_data(example):
    # تنظيف النصوص
    tokens = clean_text(example['document'])
    tags = example['doc_bio_tags']

    # Make sure the length matches the words and labels
    max_length = 512
    tokens = tokens[:max_length]
    tags = tags[:max_length]

    # Convert labels to numbers
    tags = [BIO_MAP[tag] for tag in tags]

    return {"tokens": tokens, "tags": tags}

# Data processing application
processed_dataset = dataset.map(preprocess_data)

# Oversampling for classes "B" and "I"
def oversample_data(dataset):
    oversampled_data = []
    for example in dataset:
        tags = example["tags"]
        tag_counts = Counter(tags)

        # If it contains a "B" or an "I," repeat it several times
        if tag_counts[BIO_MAP["B"]] > 0 or tag_counts[BIO_MAP["I"]] > 0:
            oversampled_data.extend([example] * 3)  # تكرار 3 مرات
        else:
            oversampled_data.append(example)

    return oversampled_data

oversampled_dataset = oversample_data(processed_dataset)
balanced_dataset = Dataset.from_list(oversampled_dataset)

# Download Tokenizer for SciBERT model
tokenizer = AutoTokenizer.from_pretrained("allenai/scibert_scivocab_uncased")

# Convert text and labels to input format for the form
def tokenize_and_align_labels(examples):
    tokenized_inputs = tokenizer(
        examples["tokens"], truncation=True, is_split_into_words=True, padding="max_length", max_length=512
    )
    labels = []
    for i, label in enumerate(examples["tags"]):
        word_ids = tokenized_inputs.word_ids(batch_index=i)  # Get word IDs
        previous_word_idx = None
        label_ids = []
        for word_idx in word_ids:
            if word_idx is None:
                label_ids.append(-100)  # Ignore these codes in the calculation
            elif word_idx != previous_word_idx:
                label_ids.append(label[word_idx])
            else:
                label_ids.append(-100)
            previous_word_idx = word_idx
        labels.append(label_ids)
    tokenized_inputs["labels"] = labels
    return tokenized_inputs

# Apply the conversion function
tokenized_dataset = balanced_dataset.map(tokenize_and_align_labels, batched=True)

# Split data into training and testing
train_test_split = tokenized_dataset.train_test_split(test_size=0.2)
train_dataset = train_test_split['train']
test_dataset = train_test_split['test']

# Download the SciBERT model with classifications setup
model = AutoModelForTokenClassification.from_pretrained(
    "allenai/scibert_scivocab_uncased",
    num_labels=3,  # عدد الفئات: B, I, O
    id2label={0: "O", 1: "B", 2: "I"},
```

```
    label2id={"O": 0, "B": 1, "I": 2}
)

# **Gradual Unfreezing **
# Freezing the first layers (8 layers)
for param in model.bert.encoder.layer[:8].parameters():
    param.requires_grad = False

# Trainer preparation
data_collator = DataCollatorForTokenClassification(tokenizer)
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,  # Initial training
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=10,
    save_strategy="epoch"
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    data_collator=data_collator
)

# First stage training
trainer.train()

# ** Unfreeze extra layers (Gradual Unfreezing) **
# Unfreeze the next layers (layers 8-12)
for param in model.bert.encoder.layer[:12].parameters():
    param.requires_grad = True

# Update training settings
training_args.num_train_epochs = 5  # Additional training
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    data_collator=data_collator
)

# Second stage training
trainer.train()

# Model evaluation
predictions, labels, _ = trainer.predict(test_dataset)
predictions = np.argmax(predictions, axis=2)

# Restore text and labels while discarding filled symbols and reconverting numbers to text labels
reverse_BIO_MAP = {0: "O", 1: "B", 2: "I"}
true_labels = []
predicted_labels = []

for label_list, prediction_list in zip(labels, predictions):
    filtered_true_labels = []
    filtered_predicted_labels = []
    for label, prediction in zip(label_list, prediction_list):
        if label != -100:  # Ignore filled icons
            filtered_true_labels.append(reverse_BIO_MAP[label])
            filtered_predicted_labels.append(reverse_BIO_MAP[prediction])
    true_labels.append(filtered_true_labels)
    predicted_labels.append(filtered_predicted_labels)

# Performance calculation
from seqeval.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

print("Accuracy:", accuracy_score(true_labels, predicted_labels))
print("Precision:", precision_score(true_labels, predicted_labels))
print("Recall:", recall_score(true_labels, predicted_labels))
print("F1 Score:", f1_score(true_labels, predicted_labels))
print("Classification Report:\n", classification_report(true_labels, predicted_labels))
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
```

README.md: 100% 6.63k/6.63k [00:00<00:00, 541kB/s]

kp20k.py: 100% 6.53k/6.53k [00:00<00:00, 490kB/s]

```
Repo card metadata block was not found. Setting CardData to empty.
WARNING:huggingface_hub.repocard:Repo card metadata block was not found. Setting CardData to empty.
```

test.jsonl: 100% 51.6M/51.6M [00:00<00:00, 119MB/s]

train.jsonl: 100% 1.37G/1.37G [00:06<00:00, 215MB/s]

valid.jsonl: 100% 51.6M/51.6M [00:00<00:00, 227MB/s]

Generating train split: 530809/0 [06:58<00:00, 1303.24 examples/s]

Generating test split: 20000/0 [00:15<00:00, 1318.53 examples/s]

Generating validation split: 20000/0 [00:15<00:00, 1210.18 examples/s]

Map: 100% 5000/5000 [00:02<00:00, 2254.54 examples/s]

config.json: 100% 385/385 [00:00<00:00, 33.9kB/s]

vocab.txt: 100% 228k/228k [00:00<00:00, 3.02MB/s]

Map: 100% 13694/13694 [00:11<00:00, 1245.25 examples/s]

pytorch_model.bin: 100% 442M/442M [00:02<00:00, 231MB/s]

```
Some weights of BertForTokenClassification were not initialized from the model checkpoint at allenai/scibert_scivocab_uncased and ar
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1568: FutureWarning: `evaluation_strategy` is deprecated and w
  warnings.warn(
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not intended, please
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
```

model.safetensors: 100% 442M/442M [00:02<00:00, 232MB/s]

```
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit:wandb: Appending key for api.wandb.ai to your netrc
Tracking run with wandb version 0.18.6
Run data is saved locally in /content/wandb/run-20241115_211600-xoo9drer
Syncing run ./results to Weights & Biases (docs)
View project at https://wandb.ai/a-k-aldhayan9-stc-/huggingface
View run at https://wandb.ai/a-k-aldhayan9-stc-/huggingface/runs/xoo9drer
```

[4110/4110 20:24, Epoch 3/3]

Epoch	Training Loss	Validation Loss
1	0.153700	0.152137
2	0.129600	0.130963
3	0.126300	0.126907

[5481/6850 32:50 < 08:12, 2.78 it/s, Epoch 4/5]

Epoch	Training Loss	Validation Loss
1	0.082100	0.095002
2	0.047300	0.060763
3	0.035800	0.044422

[70/343 00:06 < 00:27, 10.02 it/s]

[6850/6850 41:58, Epoch 5/5]

Epoch	Training Loss	Validation Loss
1	0.082100	0.095002
2	0.047300	0.060763
3	0.035800	0.044422
4	0.014000	0.038900
5	0.011600	0.036633

```
Accuracy: 0.9908405334096623
Precision: 0.8444717780511138
Recall: 0.9032305218535301
F1 Score: 0.8728633987851391
Classification Report:
              precision    recall  f1-score   support

           _       0.84      0.90      0.87     13682

   micro avg       0.84      0.90      0.87     13682
   macro avg       0.84      0.90      0.87     13682
weighted avg       0.84      0.90      0.87     13682
```

Keyphrase Extraction using Traditional Method

```
!pip install transformers torch seqeval sklearn datasets
!pip install seqeval
!pip install datasets
!pip install transformers
```

```
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.11.2)
Requirement already satisfied: huggingface-hub>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.26.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (2.4.
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (0.2.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.17.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.23
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datas
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->dataset
Downloading datasets-3.1.0-py3-none-any.whl (480 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 480.6/480.6 kB 34.5 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 12.4 MB/s eta 0:00:00
Downloading fsspec-2024.9.0-py3-none-any.whl (179 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.3/179.3 kB 19.1 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 14.9 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 20.8 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2024.10.0
    Uninstalling fsspec-2024.10.0:
      Successfully uninstalled fsspec-2024.10.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the
gcsfs 2024.10.0 requires fsspec==2024.10.0, but you have fsspec 2024.9.0 which is incompatible.
Successfully installed datasets-3.1.0 dill-0.3.8 fsspec-2024.9.0 multiprocess-0.70.16 xxhash-3.5.0
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.46.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.26
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.9.11)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: tokenizers<0.21,>=0.20 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.6)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.2->t
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.2.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024
```

```
import os
import numpy as np
from datasets import load_dataset, Dataset
from transformers import AutoTokenizer, AutoModelForTokenClassification, TrainingArguments, Trainer
from transformers import DataCollatorForTokenClassification
from collections import Counter
import re
import torch
from torch.nn import CrossEntropyLoss

# Download data
dataset = load_dataset("midas/kp20k", split="train").select(range(5000))  # استخدام 5000 عينة

# Convert BIO labels to numbers
BIO_MAP = {"O": 0, "B": 1, "I": 2}
```

```
# Text cleaning function
def clean_text(text):
    if isinstance(text, list):  # If the text is a list of words
        text = " ".join(text)
    # Remove symbols other than letters
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    # Divide texts into words
    tokens = text.split()
    return tokens

# Text processing and BIO nomenclature
def preprocess_data(example):
    # تـنظيف الـنصوص
    tokens = clean_text(example['document'])
    tags = example['doc_bio_tags']

    # Make sure the length matches the words and labels
    max_length = 512
    tokens = tokens[:max_length]
    tags = tags[:max_length]

    # Convert labels to numbers
    tags = [BIO_MAP[tag] for tag in tags]

    return {"tokens": tokens, "tags": tags}

# Data processing application
processed_dataset = dataset.map(preprocess_data)

# Oversampling for classes "B" and "I"
def oversample_data(dataset):
    oversampled_data = []
    for example in dataset:
        tags = example["tags"]
        tag_counts = Counter(tags)

        # If it contains a "B" or an "I," repeat it several times
        if tag_counts[BIO_MAP["B"]] > 0 or tag_counts[BIO_MAP["I"]] > 0:
            oversampled_data.extend([example] * 3)  # Repeat 3 times
        else:
            oversampled_data.append(example)

    return oversampled_data

oversampled_dataset = oversample_data(processed_dataset)
balanced_dataset = Dataset.from_list(oversampled_dataset)

# Download Tokenizer for SciBERT model
tokenizer = AutoTokenizer.from_pretrained("allenai/scibert_scivocab_uncased")

# Convert text and labels to input format for the form
def tokenize_and_align_labels(examples):
    tokenized_inputs = tokenizer(
        examples["tokens"], truncation=True, is_split_into_words=True, padding="max_length", max_length=512
    )
    labels = []
    for i, label in enumerate(examples["tags"]):
        word_ids = tokenized_inputs.word_ids(batch_index=i)  # Get word identifiers
        previous_word_idx = None
        label_ids = []
        for word_idx in word_ids:
            if word_idx is None:
                label_ids.append(-100)  # Ignore these codes in the calculation
            elif word_idx != previous_word_idx:
                label_ids.append(label[word_idx])
            else:
                label_ids.append(-100)
            previous_word_idx = word_idx
        labels.append(label_ids)
    tokenized_inputs["labels"] = labels
    return tokenized_inputs

# Apply the conversion function
tokenized_dataset = balanced_dataset.map(tokenize_and_align_labels, batched=True)

# Split data into training and testing
train_test_split = tokenized_dataset.train_test_split(test_size=0.2)
train_dataset = train_test_split['train']
test_dataset = train_test_split['test']

# Download the SciBERT model with classifications setup
model = AutoModelForTokenClassification.from_pretrained(
    "allenai/scibert_scivocab_uncased",
```

```
    num_labels=3,  # عدد الفئات: B, I, O
    id2label={0: "O", 1: "B", 2: "I"},
    label2id={"O": 0, "B": 1, "I": 2}
)

# ** Freeze the lower layers (Gradual Unfreezing) **
# Freezing the first layers (8 layers)
for param in model.bert.encoder.layer[:8].parameters():
    param.requires_grad = False

# Trainer preparation (Trainer)
data_collator = DataCollatorForTokenClassification(tokenizer)
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,  # Initial training
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=10,
    save_strategy="epoch"
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    data_collator=data_collator
)

# First stage training
trainer.train()

# ** Unfreeze extra layers (Gradual Unfreezing) **
# Unfreeze the next layers (layers 8-12)
for param in model.bert.encoder.layer[:12].parameters():
    param.requires_grad = True

# Update training settings
training_args.num_train_epochs = 5  # Additional training
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    data_collator=data_collator
)

# Second stage training
trainer.train()

# Model evaluation
predictions, labels, _ = trainer.predict(test_dataset)
predictions = np.argmax(predictions, axis=2)

# Restore text and labels while discarding filled symbols and reconverting numbers to text labels
reverse_BIO_MAP = {0: "O", 1: "B", 2: "I"}
true_labels = []
predicted_labels = []

for label_list, prediction_list in zip(labels, predictions):
    filtered_true_labels = []
    filtered_predicted_labels = []
    for label, prediction in zip(label_list, prediction_list):
        if label != -100:  # تجاهل الرموز المعبأة
            filtered_true_labels.append(reverse_BIO_MAP[label])
            filtered_predicted_labels.append(reverse_BIO_MAP[prediction])
    true_labels.append(filtered_true_labels)
    predicted_labels.append(filtered_predicted_labels)

# Performance calculation
from seqeval.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

print("Accuracy:", accuracy_score(true_labels, predicted_labels))
print("Precision:", precision_score(true_labels, predicted_labels))
print("Recall:", recall_score(true_labels, predicted_labels))
print("F1 Score:", f1_score(true_labels, predicted_labels))
print("Classification Report:\n", classification_report(true_labels, predicted_labels))
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
README.md: 100%  6.63k/6.63k [00:00<00:00, 566kB/s]
kp20k.py: 100%  6.53k/6.53k [00:00<00:00, 550kB/s]
Repo card metadata block was not found. Setting CardData to empty.
WARNING:huggingface_hub.repocard:Repo card metadata block was not found. Setting CardData to empty.
test.jsonl: 100%  51.6M/51.6M [00:03<00:00, 18.1MB/s]
train.jsonl: 100%  1.37G/1.37G [00:56<00:00, 25.8MB/s]
valid.jsonl: 100%  51.6M/51.6M [00:02<00:00, 28.0MB/s]
Generating train split:  530809/0 [06:49<00:00, 1323.79 examples/s]
Generating test split:  20000/0 [00:15<00:00, 1226.91 examples/s]
Generating validation split:  20000/0 [00:15<00:00, 1359.70 examples/s]
Map: 100%  5000/5000 [00:02<00:00, 2088.94 examples/s]
config.json: 100%  385/385 [00:00<00:00, 29.3kB/s]
vocab.txt: 100%  228k/228k [00:00<00:00, 1.08MB/s]
Map: 100%  13694/13694 [00:11<00:00, 1162.26 examples/s]
pytorch_model.bin: 100%  442M/442M [00:01<00:00, 253MB/s]
Some weights of BertForTokenClassification were not initialized from the model checkpoint at allenai/scibert_scivocab_uncased and ar
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1568: FutureWarning: `evaluation_strategy` is deprecated and w
  warnings.warn(
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not intended, please
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.18.7
Run data is saved locally in /content/wandb/run-20241129_211955-83h4ilwm
Syncing run ./results to Weights & Biases (docs)
View project at https://wandb.ai/a-k-aldhayan9-stc-/huggingface
View run at https://wandb.ai/a-k-aldhayan9-stc-/huggingface/runs/83h4ilwm
```

[4110/4110 20:36, Epoch 3/3]

Epoch	Training Loss	Validation Loss
1	0.143700	0.146108
2	0.112700	0.136016
3	0.102100	0.130053

[6850/6850 41:59, Epoch 5/5]

Epoch	Training Loss	Validation Loss
1	0.082600	0.093281
2	0.046800	0.071336
3	0.027200	0.055528
4	0.016800	0.045713
5	0.015400	0.044336

```
Accuracy: 0.9898644819714069
Precision: 0.8407020364415863
Recall: 0.8919053372183925
F1 Score: 0.8655470878306148
Classification Report:
              precision    recall  f1-score   support

           _       0.84      0.89      0.87     14071

   micro avg       0.84      0.89      0.87     14071
   macro avg       0.84      0.89      0.87     14071
weighted avg       0.84      0.89      0.87     14071
```