HW 4 - POS Tagging with Hugging Face

In this exercise, you will create a part-of-speech (POS) tagging system for Thai text using NECTEC's ORCHID corpus. Instead of building your own deep learning architecture from scratch, you will leverage a pretrained tokenizer and a pretrained token classification model from Hugging Face.

We have provided some starter code for data cleaning and preprocessing in this notebook, but feel free to modify those parts to suit your needs. You are welcome to use additional libraries (e.g., scikit-learn) as long as you incorporate the pretrained Hugging Face model. Specifically, you will need to:

1. Load a pretrained tokenizer and token classification model.
2. Fine-tune it on the ORCHID corpus for POS tagging.
3. Evaluate and report the performance of your model on the test data.

Don't forget to change hardware accelrator to GPU in runtime on Google Colab

1. Setup and Preprocessing

```
# Install transformers and thai2transformers
!pip install wandb
!pip install -q transformers==4.30.1 datasets evaluate
thaixtransformers
!pip install -q emoji pythainlp sefr_cut tinydb seqeval sentencepiece
pydantic jsonlines
!pip install peft==0.10.0
```

```
Requirement already satisfied: wandb in
/usr/local/lib/python3.10/dist-packages (0.19.1)
Requirement already satisfied: click!=8.0.0,>=7.1 in
/usr/local/lib/python3.10/dist-packages (from wandb) (8.1.7)
Requirement already satisfied: docker-pycreds>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (0.4.0)
Requirement already satisfied: gitpython!=3.1.29,>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (3.1.43)
Requirement already satisfied: platformdirs in
/usr/local/lib/python3.10/dist-packages (from wandb) (4.3.6)
Requirement already satisfied: protobuf!=4.21.0,!=5.28.0,<6,>=3.19.0
in /usr/local/lib/python3.10/dist-packages (from wandb) (3.20.3)
Requirement already satisfied: psutil>=5.0.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (5.9.5)
Requirement already satisfied: pydantic<3,>=2.6 in
/usr/local/lib/python3.10/dist-packages (from wandb) (2.10.3)
Requirement already satisfied: pyyaml in
/usr/local/lib/python3.10/dist-packages (from wandb) (6.0.2)
```

```
Requirement already satisfied: requests<3,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (2.32.3)
Requirement already satisfied: sentry-sdk>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (2.19.2)
Requirement already satisfied: setproctitle in
/usr/local/lib/python3.10/dist-packages (from wandb) (1.3.4)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from wandb) (75.1.0)
Requirement already satisfied: typing-extensions<5,>=4.4 in
/usr/local/lib/python3.10/dist-packages (from wandb) (4.12.2)
Requirement already satisfied: six>=1.4.0 in
/usr/local/lib/python3.10/dist-packages (from docker-pycreds>=0.4.0-
>wandb) (1.17.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in
/usr/local/lib/python3.10/dist-packages (from gitpython!
=3.1.29,>=1.0.0->wandb) (4.0.11)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3,>=2.6->wandb)
(0.7.0)
Requirement already satisfied: pydantic-core==2.27.1 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3,>=2.6->wandb)
(2.27.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (2024.12.14)
Requirement already satisfied: smmap<6,>=3.0.1 in
/usr/local/lib/python3.10/dist-packages (from gitdb<5,>=4.0.1-
>gitpython!=3.1.29,>=1.0.0->wandb) (5.0.1)
Requirement already satisfied: peft==0.10.0 in
/usr/local/lib/python3.10/dist-packages (0.10.0)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0) (24.2)
Requirement already satisfied: psutil in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0) (5.9.5)
Requirement already satisfied: pyyaml in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0) (6.0.2)
Requirement already satisfied: torch>=1.13.0 in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0)
```

```
(2.5.1+cu121)
Requirement already satisfied: transformers in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0) (4.30.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from peft==0.10.0) (4.67.1)
Requirement already satisfied: accelerate>=0.21.0 in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0) (1.2.1)
Requirement already satisfied: safetensors in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0) (0.4.5)
Requirement already satisfied: huggingface-hub>=0.17.0 in
/usr/local/lib/python3.10/dist-packages (from peft==0.10.0) (0.27.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.17.0-
>peft==0.10.0) (3.16.1)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.17.0-
>peft==0.10.0) (2024.9.0)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.17.0-
>peft==0.10.0) (2.32.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.17.0-
>peft==0.10.0) (4.12.2)
Requirement already satisfied: mkl_fft in
/usr/local/lib/python3.10/dist-packages (from numpy>=1.17-
>peft==0.10.0) (1.3.8)
Requirement already satisfied: mkl_random in
/usr/local/lib/python3.10/dist-packages (from numpy>=1.17-
>peft==0.10.0) (1.2.4)
Requirement already satisfied: mkl_umath in
/usr/local/lib/python3.10/dist-packages (from numpy>=1.17-
>peft==0.10.0) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dist-
packages (from numpy>=1.17->peft==0.10.0) (2025.0.1)
Requirement already satisfied: tbb4py in
/usr/local/lib/python3.10/dist-packages (from numpy>=1.17-
>peft==0.10.0) (2022.0.0)
Requirement already satisfied: mkl-service in
/usr/local/lib/python3.10/dist-packages (from numpy>=1.17-
>peft==0.10.0) (2.4.1)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13.0-
>peft==0.10.0) (3.4.2)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13.0-
>peft==0.10.0) (3.1.4)
Requirement already satisfied: sympy==1.13.1 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.13.0-
>peft==0.10.0) (1.13.1)
```

```
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy==1.13.1-
>torch>=1.13.0->peft==0.10.0) (1.3.0)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers-
>peft==0.10.0) (2024.11.6)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in
/usr/local/lib/python3.10/dist-packages (from transformers-
>peft==0.10.0) (0.13.3)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.13.0-
>peft==0.10.0) (3.0.2)
Requirement already satisfied: intel-openmp>=2024 in
/usr/local/lib/python3.10/dist-packages (from mkl->numpy>=1.17-
>peft==0.10.0) (2024.2.0)
Requirement already satisfied: tbb==2022.* in
/usr/local/lib/python3.10/dist-packages (from mkl->numpy>=1.17-
>peft==0.10.0) (2022.0.0)
Requirement already satisfied: tcmlib==1.* in
/usr/local/lib/python3.10/dist-packages (from tbb==2022.*->mkl-
>numpy>=1.17->peft==0.10.0) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.10/dist-packages (from mkl_umath->numpy>=1.17-
>peft==0.10.0) (2024.2.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.17.0->peft==0.10.0) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.17.0->peft==0.10.0) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.17.0->peft==0.10.0) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.17.0->peft==0.10.0) (2024.12.14)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.10/dist-packages (from intel-openmp>=2024->mkl-
>numpy>=1.17->peft==0.10.0) (2024.2.0)
```

Setup

1. Register Wandb account (and confirm your email)
2. `wandb login` and copy paste the API key when prompt

```
from kaggle_secrets import UserSecretsClient
secret_label = "wandb_api_key"
secret_value = UserSecretsClient().get_secret(secret_label)
```

```
!wandb login $secret_value
```

```
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the command line.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
```

```
import wandb
```

We encourage you to login to your `Hugging Face` account so you can upload and share your model with the community. When prompted, enter your token to login

```
from huggingface_hub import notebook_login

notebook_login()
```

```
{"model_id":"b4bd14c670164d70a8d47389b940c8a2","version_major":2,"version_minor":0}
```

Download the dataset from Hugging Face

```
from datasets import load_dataset

orchid = load_dataset("Thichow/orchid_corpus")
```

```
{"model_id":"ed29ab498cbf420c9fdfefb9001cb7e0","version_major":2,"version_minor":0}
```

```
{"model_id":"c60e8e345ce94ebfb7d5185394608748","version_major":2,"version_minor":0}
```

```
The repository for Thichow/orchid_corpus contains custom code which must be executed to correctly load the dataset. You can inspect the repository content at https://hf.co/datasets/Thichow/orchid_corpus.
You can avoid this prompt in future by passing the argument `trust_remote_code=True`.

Do you wish to run the custom code? [y/N]  y
```

```
{"model_id":"7ef5fd3b2acc4386a3e254a75fad4660","version_major":2,"version_minor":0}
```

```
{"model_id":"d324d1d1fdf94daa9f82c1f1b03d06c7","version_major":2,"version_minor":0}
```

```
{"model_id":"591755d0528748c99af42376e41b0925","version_major":2,"version_minor":0}
```

{"model_id":"7b1361c21d3c40849034ecfc3ea70971","version_major":2,"version_minor":0}

```
orchid
```

```
DatasetDict({
    train: Dataset({
        features: ['id', 'label_tokens', 'pos_tags', 'sentence'],
        num_rows: 18500
    })
    test: Dataset({
        features: ['id', 'label_tokens', 'pos_tags', 'sentence'],
        num_rows: 4625
    })
})
```

```
orchid['train'][0]
```

```
{'id': '0',
 'label_tokens': ['การ', 'ประชุม', 'ทาง', 'วิชาการ', ' ', 'ครั้ง', 'ที่ 1'],
 'pos_tags': [21, 39, 26, 26, 37, 4, 18],
 'sentence': 'การประชุมทางวิชาการ ครั้งที่ 1'}
```

```
orchid['train'][0]["sentence"]
```

```
'การประชุมทางวิชาการ ครั้งที่ 1'
```

```
''.join(orchid['train'][0]['label_tokens'])
```

```
'การประชุมทางวิชาการ ครั้งที่ 1'
```

```
label_list = orchid["train"].features[f"pos_tags"].feature.names
print('total type of pos_tags :', len(label_list))
print(label_list)
```

```
total type of pos_tags : 47
['ADVI', 'ADVN', 'ADVP', 'ADVS', 'CFQC', 'CLTV', 'CMTR', 'CMTR@PUNC',
'CNIT', 'CVBL', 'DCNM', 'DDAC', 'DDAN', 'DDAQ', 'DDBQ', 'DIAC',
'DIAQ', 'DIBQ', 'DONM', 'EAFF', 'EITT', 'FIXN', 'FIXV', 'JCMP',
'JCRG', 'JSBR', 'NCMN', 'NCNM', 'NEG', 'NLBL', 'NONM', 'NPRP', 'NTTL',
'PDMN', 'PNTR', 'PPRS', 'PREL', 'PUNC', 'RPRE', 'VACT', 'VATT',
'VSTA', 'XVAE', 'XVAM', 'XVBB', 'XVBM', 'XVMM']
```

```
import numpy as np
import numpy.random
import torch

from tqdm.auto import tqdm
from functools import partial

#transformers
from transformers import (
```

```
    CamembertTokenizer,
    AutoTokenizer,
    AutoModel,
    AutoModelForMaskedLM,
    AutoModelForSequenceClassification,
    AutoModelForTokenClassification,
    TrainingArguments,
    Trainer,
    pipeline,
)

#thaixtransformers
from thaixtransformers import Tokenizer
from thaixtransformers.preprocess import process_transformers
```

```
The cache for model files in Transformers v4.22.0 has been updated.
Migrating your old cache. This is a one-time only operation. You can
interrupt this and resume the migration later on by calling
`transformers.utils.move_cache()`.
```


```
{"model_id":"8e336a17b87445e1b2fd97a4d722bfb5","version_major":2,"vers
ion_minor":0}
```


Next, we load a pretrained tokenizer from Hugging Face. In this work, we utilize WangchanBERTa, a Thai-specific pretrained model, as the tokenizer.

Choose Pretrained Model

In this notebook, you can choose from 5 versions of WangchanBERTa, XLMR and mBERT to perform downstream tasks on Thai datasets. The datasets are:

- `wangchanberta-base-att-spm-uncased` (recommended) - Largest WangchanBERTa trained on 78.5GB of Assorted Thai Texts with subword tokenizer SentencePiece
- `xlm-roberta-base` - Facebook's XLMR trained on 100 languages
- `bert-base-multilingual-cased` - Google's mBERT trained on 104 languages
- `wangchanberta-base-wiki-newmm` - WangchanBERTa trained on Thai Wikipedia Dump with PyThaiNLP's word-level tokenizer `newmm`
- `wangchanberta-base-wiki-syllable` - WangchanBERTa trained on Thai Wikipedia Dump with PyThaiNLP's syllabel-level tokenizer `syllable`
- `wangchanberta-base-wiki-sefr` - WangchanBERTa trained on Thai Wikipedia Dump with word-level tokenizer SEFR
- `wangchanberta-base-wiki-spm` - WangchanBERTa trained on Thai Wikipedia Dump with subword-level tokenizer SentencePiece

In the first part, we require you to select the wangchanberta-base-att-spm-uncased.

Learn more about using wangchanberta at wangchanberta_getting_started_ai_reseach

- You need to set the transformers version to transformers==4.30.1.

In the first part, we require you to select the wangchanberta-base-att-spm-uncased.

```
model_names = [
    'airesearch/wangchanberta-base-att-spm-uncased',
    'airesearch/wangchanberta-base-wiki-newmm',
    'airesearch/wangchanberta-base-wiki-ssg',
    'airesearch/wangchanberta-base-wiki-sefr',
    'airesearch/wangchanberta-base-wiki-spm',
]

#@title Choose Pretrained Model
model_name = "airesearch/wangchanberta-base-att-spm-uncased"

#create tokenizer
tokenizer = Tokenizer(model_name).from_pretrained(
                f'{model_name}',
                revision='main',
                model_max_length=416,)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/
file_download.py:795: FutureWarning: `resume_download` is deprecated
and will be removed in version 1.0.0. Downloads always resume when
possible. If you want to force a new download, use
`force_download=True`.
  warnings.warn(
```


```
{"model_id":"a0f70a474d4541d196bc3af65aca45fe","version_major":2,"vers
ion_minor":0}

{"model_id":"2ff4e5776ba14a99a0b7fbef469ca04f","version_major":2,"vers
ion_minor":0}

{"model_id":"b80c6f965cf542fcb4ca227525377aa6","version_major":2,"vers
ion_minor":0}
```


```
The tokenizer class you load from this checkpoint is not the same type
as the class this function is called from. It may result in unexpected
tokenization.
The tokenizer class you load from this checkpoint is
'CamembertTokenizer'.
The class this function is called from is 'WangchanbertaTokenizer'.
The tokenizer class you load from this checkpoint is not the same type
as the class this function is called from. It may result in unexpected
tokenization.
The tokenizer class you load from this checkpoint is
'CamembertTokenizer'.
The class this function is called from is 'WangchanbertaTokenizer'.
```

Let's try using a pretrained tokenizer.

```
text = 'ศิลปะไม่เป็นเจ้านายใคร และไม่เป็นขี้ข้าใคร'
print('text :', text)
tokens = []
for i in tokenizer([text], is_split_into_words=True)['input_ids']:
  tokens.append(tokenizer.decode(i))
print('tokens :', tokens)
```

```
text : ศิลปะไม่เป็นเจ้านายใคร และไม่เป็นขี้ข้าใคร
tokens : ['<s>', '', 'ศิลปะ', 'ไม่เป็น', 'เจ้านาย', 'ใคร', '<_>', 'และ',
'ไม่เป็น', 'ขี้ข้า', 'ใคร', '</s>']
```

model : * `wangchanberta-base-att-spm-uncased`

First, we print examples of label tokens from our dataset for inspection.

```
example = orchid["train"][0]
for i in example :
    print(i, ':', example[i])
```

```
id : 0
label_tokens : ['การ', 'ประชุม', 'ทาง', 'วิชาการ', ' ', 'ครั้ง', 'ที่ 1']
pos_tags : [21, 39, 26, 26, 37, 4, 18]
sentence : การประชุมทางวิชาการ ครั้งที่ 1
```

Then, we use the sentence 'การประชุมทางวิชาการครั้งที่ 1' to be tokenized by the pretrained tokenizer model.

```
text = 'การประชุมทางวิชาการ ครั้งที่ 1'
tokenizer(text)
```

```
{'input_ids': [5, 10, 882, 8222, 8, 10, 1014, 8, 10, 59, 6],
'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

These are already mapped into discrete values. We can uncover the original token text from the tokens by.

```
for i in tokenizer(text)['input_ids']:
  print(tokenizer.convert_ids_to_tokens(i))
```

```
<s>
▁
การประชุม
ทางวิชาการ
<_>
▁
ครั้งที่
<_>
```

```
▁
1
</s>
```

Now let's look at another example.

```
example = orchid["train"][1899]
print('sentence :', example["sentence"])
tokenized_input = tokenizer([example["sentence"]],
is_split_into_words=True)
tokens = tokenizer.convert_ids_to_tokens(tokenized_input["input_ids"])
print('tokens :',tokens)
print('label tokens :', example["label_tokens"])
print('label pos :', example["pos_tags"])
```

```
sentence : โดยพิจารณาจากพจนานุกรมภาษาคู่ (Bilingual transfer dictionary)
tokens : ['<s>', '▁โดย', 'พิจารณาจาก', 'พจนานุกรม', 'ภาษา', 'คู่',
'<_>', '▁(', '<unk>', 'i', 'ling', 'ual', '<_>', '▁', 'trans', 'fer',
'<_>', '▁', 'di', 'ction', 'ary', ')', '</s>']
label tokens : ['โดย', 'พิจารณา', 'จาก', 'พจนานุกรม', 'ภาษา', 'คู่', '
', '(', 'Bilingual transfer dictionary', ')']
label pos : [25, 39, 38, 26, 26, 5, 37, 37, 26, 37]
```

Notice how `B` becomes an `<unk>` token. This is because this is an uncased model, meaning it only handles small English characters.

#TODO 0

Convert the dataset to lowercase.

```
# Create a lowercase dataset for uncased BERT
def lower_case_sentences(examples):
  # fill code here to lower case the "sentence" and "label_tokens"
  lower_cased_examples = examples
  lower_cased_examples['sentence'] =
lower_cased_examples['sentence'].lower()
  lower_cased_examples['label_tokens'] = [t.lower() for t in
lower_cased_examples['label_tokens']]

  return lower_cased_examples

orchidl = orchid.map(lower_case_sentences)
```


```
{"model_id":"69a01b80e65e4a058ed0128dffe5246d","version_major":2,"vers
ion_minor":0}

{"model_id":"3a1d1ca0422a4449ad3678936bc03a56","version_major":2,"vers
ion_minor":0}
```

```
orchidl
```

```
DatasetDict({
    train: Dataset({
        features: ['id', 'label_tokens', 'pos_tags', 'sentence'],
        num_rows: 18500
    })
    test: Dataset({
        features: ['id', 'label_tokens', 'pos_tags', 'sentence'],
        num_rows: 4625
    })
})
```

```
orchidl["train"][1899]
```

```
{'id': '1899',
 'label_tokens': ['โดย',
  'พิจารณา',
  'จาก',
  'พจนานุกรม',
  'ภาษา',
  'คู่',
  ' ',
  '(',
  'bilingual transfer dictionary',
  ')'],
 'pos_tags': [25, 39, 38, 26, 26, 5, 37, 37, 26, 37],
 'sentence': 'โดยพิจารณาจากพจนานุกรมภาษาคู่ (bilingual transfer
dictionary)'}
```

Now let's examine the labels again.

```
example = orchidl["train"][1899]
print('sentence :', example["sentence"])
tokenized_input = tokenizer([example["sentence"]],
is_split_into_words=True)
tokens = tokenizer.convert_ids_to_tokens(tokenized_input["input_ids"])
print('tokens :',tokens)
print('label tokens :', example["label_tokens"])
print('label pos :', example["pos_tags"])
```

```
sentence : โดยพิจารณาจากพจนานุกรมภาษาคู่ (bilingual transfer dictionary)
tokens : ['<s>', '▁โดย', 'พิจารณาจาก', 'พจนานุกรม', 'ภาษา', 'คู่',
'<_>', '▁(', 'bi', 'ling', 'ual', '<_>', '▁', 'trans', 'fer', '<_>',
'▁', 'di', 'ction', 'ary', ')', '</s>']
label tokens : ['โดย', 'พิจารณา', 'จาก', 'พจนานุกรม', 'ภาษา', 'คู่', '
', '(', 'bilingual transfer dictionary', ')']
label pos : [25, 39, 38, 26, 26, 5, 37, 37, 26, 37]
```

```
example = orchidl["train"][0]
print('sentence :', example["sentence"])
tokenized_input = tokenizer([example["sentence"]],
is_split_into_words=True)
tokens = tokenizer.convert_ids_to_tokens(tokenized_input["input_ids"])
print('tokens :',tokens)
print('label tokens :', example["label_tokens"])
print('label pos :', example["pos_tags"])
```

```
sentence : การประชุมทางวิชาการ ครั้งที่ 1
tokens : ['<s>', '▁', 'การประชุม', 'ทางวิชาการ', '<_>', '▁', 'ครั้งที่',
'<_>', '▁', '1', '</s>']
label tokens : ['การ', 'ประชุม', 'ทาง', 'วิชาการ', ' ', 'ครั้ง', 'ที่ 1']
label pos : [21, 39, 26, 26, 37, 4, 18]
```

In the example above, tokens refer to those tokenized using the pretrained tokenizer, while label tokens refer to tokens tokenized from our dataset.

Do you see something?

Yes, the tokens from the two tokenizers do not match.

- sentence : `การประชุมทางวิชาการ ครั้งที่ 1`

- tokens : `['<s>', '▁', 'การประชุม', 'ทางวิชาการ', '<_>', '▁', 'ครั้งที่', '<_>', '▁', '1', '</s>']`

- label tokens : `['การ', 'ประชุม', 'ทาง', 'วิชาการ', ' ', 'ครั้ง', 'ที่ 1']`
- label pos : `[21, 39, 26, 26, 37, 4, 18]`

You can see that in our label tokens, 'การ' has a POS tag of 21, and 'ประชุม' has a POS tag of 39. However, when we tokenize the sentence using WangchanBERTa, we get the token 'การประชุม'. What POS tag should we assign to this new token?

What should we do ?

Based on this example, we found that the tokens from the WangchanBERTa do not directly align with our label tokens. This means we cannot directly use the label POS tags. Therefore, we need to reassign POS tags to the tokens produced by WangchanBERTa tokenization. The method we will use is majority voting:

- If a token from the WangchanBERTa matches a label token exactly, we will directly assign the POS tag from the label POS.
- If the token generated overlaps or combines multiple label tokens, we assign the POS tag based on the number of characters in each token: If the token contains the most characters from any label token, we assign the POS tag from that label token.

Example :

```
# "การประชุม" (9 chars) is formed from "การ" (3 chars) + "ประชุม" (6
chars).
# "การ" has a POS tag of 21,
# and "ประชุม" has a POS tag of 39.
# Therefore, the POS tag for "การประชุม" is 39,
# as "การประชุม" is derived more from the "ประชุม" part than from the
"การ" part.

# 'ทางวิชาการ' (10 chars) is formed from 'ทาง' (3 chars) + 'วิชาการ' (7
chars)
# "ทาง" has a POS tag of 26,
# and "วิชาการ" has a POS tag of 2.
# Therefore, the POS tag for "ทางวิชาการ" is 2,
# as "ทางวิชาการ" is derived more from the "ทาง" part than from the
"วิชาการ" part.
```

#TODO 1

Warning: Please be careful of <unk>, an unknown word token.

Warning: Please be careful of " ำ ", the 'am' vowel. WangchanBERTa's internal preprocessing replaces all " ำ " to ◌ำ' and 'า'

Assigning the label -100 to the special tokens `[<s>]` and `[</s>]` and `[_]` so they're ignored by the PyTorch loss function (see CrossEntropyLoss: ignore_index)

```
def majority_vote_pos(examples):


##########################################################################
###############################################
    # TO DO: Since the tokens from the output of the pretrained
tokenizer
    # do not match the tokens in the label tokens of the dataset,
    # the task is to create a function to determine the POS tags of
the tokens generated by the pretrained tokenizer.
    # This should be done by referencing the POS tags in the label
tokens. If a token partially overlaps with others,
    # the POS tag from the segment with the greater number of
characters should be assigned.
    #
    # Example :
    # "การประชุม" (9 chars) is formed from "การ" (3 chars) + "ประชุม" (6
chars).
    # "การ" has a POS tag of 21,
    # and "ประชุม" has a POS tag of 39.
```

```
    # Therefore, the POS tag for "การประชุม" is 39,
    # as "การประชุม" is derived more from the "ประชุม" part than from
the "การ" part.
    #
    # 'ทางวิชาการ' (10 chars) is formed from 'ทาง' (3 chars) + 'วิชาการ'
(7 chars)
    # "ทาง" has a POS tag of 26,
    # and "วิชาการ" has a POS tag of 2.
    # Therefore, the POS tag for "ทางวิชาการ" is 2,
    # as "ทางวิชาการ" is derived more from the "ทาง" part than from the
"วิชาการ" part.

    # tokenize word by pretrained tokenizer
    tokenized_inputs = tokenizer([examples["sentence"]],
is_split_into_words=True)

    # FILL CODE HERE
    label_tokens = examples["label_tokens"]
    pos_tags = examples["pos_tags"]
    new_pos_result = []

    new_tokens =
tokenizer.convert_ids_to_tokens(tokenized_inputs["input_ids"])

    label_idx, i = 0, 0

    for t in new_tokens:
        if t in {"<s>", "</s>", "▁"}:
            new_pos_result.append(-100)
            continue

        buffer = ""
        weights = {}

        # Normalize token
        t = t.replace('ำ', 'ำ').replace("<_>", " ")
        t = t[1:] if t.startswith("▁") else t

        # Find the correct label index
        while label_tokens[label_idx][i] != t[0]:
            i += 1
            if i == len(label_tokens[label_idx]):
                label_idx += 1
                i = 0

        # Aggregate POS tags
        while buffer != t:
            buffer += label_tokens[label_idx][i]
            weights[pos_tags[label_idx]] =
```

```
weights.get(pos_tags[label_idx], 0) + 1
            i += 1
            if i == len(label_tokens[label_idx]):
                label_idx += 1
                i = 0

        new_pos_result.append(max(weights, key=weights.get))

    tokenized_inputs['tokens'] = new_tokens
    tokenized_inputs['labels'] = new_pos_result

    return tokenized_inputs

################################################################################
##############################################

tokenized_orchid = orchidl.map(majority_vote_pos)
```


```
{"model_id":"1d02d617b95b4d9bae1ed79b5d839164","version_major":2,"vers
ion_minor":0}

{"model_id":"7a9151406448494aaa5af45521b356e8","version_major":2,"vers
ion_minor":0}
```


```
tokenized_orchid
```

```
DatasetDict({
    train: Dataset({
        features: ['id', 'label_tokens', 'pos_tags', 'sentence',
'input_ids', 'attention_mask', 'tokens', 'labels'],
        num_rows: 18500
    })
    test: Dataset({
        features: ['id', 'label_tokens', 'pos_tags', 'sentence',
'input_ids', 'attention_mask', 'tokens', 'labels'],
        num_rows: 4625
    })
})
```

```
tokenized_orchid['train'][0]
```

```
{'id': '0',
 'label_tokens': ['การ', 'ประชุม', 'ทาง', 'วิชาการ', ' ', 'ครั้ง', 'ที่ 1'],
 'pos_tags': [21, 39, 26, 26, 37, 4, 18],
 'sentence': 'การประชุมทางวิชาการ ครั้งที่ 1',
 'input_ids': [5, 10, 882, 8222, 8, 10, 1014, 8, 10, 59, 6],
 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
 'tokens': ['<s>',
  '▁',
  'การประชุม',
  'ทางวิชาการ',
```

```
  '<_>',
  '▁',
  'ครั้งที่',
  '<_>',
  '▁',
  '1',
  '</s>'],
 'labels': [-100, -100, 39, 26, 37, -100, 4, 18, -100, 18, -100]}
```

```
example = tokenized_orchid["train"][0]
for i in example :
    print(i, ":", example[i])
```

```
id : 0
label_tokens : ['การ', 'ประชุม', 'ทาง', 'วิชาการ', ' ', 'ครั้ง', 'ที่ 1']
pos_tags : [21, 39, 26, 26, 37, 4, 18]
sentence : การประชุมทางวิชาการ ครั้งที่ 1
input_ids : [5, 10, 882, 8222, 8, 10, 1014, 8, 10, 59, 6]
attention_mask : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
tokens : ['<s>', '▁', 'การประชุม', 'ทางวิชาการ', '<_>', '▁', 'ครั้งที่',
'<_>', '▁', '1', '</s>']
labels : [-100, -100, 39, 26, 37, -100, 4, 18, -100, 18, -100]
```

This is the result after we realigned the POS based on the majority vote.

- label_tokens : `['การ', 'ประชุม', 'ทาง', 'วิชาการ', ' ', 'ครั้ง', 'ที่ 1']`
- pos_tags : `[21, 39, 26, 26, 37, 4, 18]`
- tokens : `['<s>', '▁', 'การประชุม', 'ทางวิชาการ', '<_>', '▁', 'ครั้งที่', '<_>', '▁', '1', '</s>']`
- labels : `[-100, -100, 39, 26, 37, -100, 4, 18, -100, 18, -100]`

`['<s>', '▁', '</s>'] : -100`

Check :

"การประชุม" (9 chars) is formed from "การ" (3 chars) + "ประชุม" (6 chars).

"การ" has a POS tag of 21,

and "ประชุม" has a POS tag of 39.

Therefore, the POS tag for "การประชุม" is 39,

as "การประชุม" is derived more from the "ประชุม" part than from the "การ" part.

```
# hard test case
example = tokenized_orchid["train"][1899]
for i in example :
    print(i, ":", example[i])
```

```
id : 1899
label_tokens : ['โดย', 'พิจารณา', 'จาก', 'พจนานุกรม', 'ภาษา', 'คู่', '
```

```
', '(', 'bilingual transfer dictionary', ')']
pos_tags : [25, 39, 38, 26, 26, 5, 37, 37, 26, 37]
sentence : โดยพิจารณาจากพจนานุกรมภาษาคู่ (bilingual transfer dictionary)
input_ids : [5, 489, 15617, 19737, 958, 493, 8, 1241, 4906, 11608,
12177, 8, 10, 11392, 9806, 8, 10, 2951, 15779, 8001, 29, 6]
attention_mask : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1]
tokens : ['<s>', '▁โดย', 'พิจารณาจาก', 'พจนานุกรม', 'ภาษา', 'คู่',
'<_>', '▁(', 'bi', 'ling', 'ual', '<_>', '▁', 'trans', 'fer', '<_>',
'▁', 'di', 'ction', 'ary', ')', '</s>']
labels : [-100, 25, 39, 26, 26, 5, 37, 37, 26, 26, 26, 26, -100, 26,
26, 26, -100, 26, 26, 26, 37, -100]
```

Expected output

```
id : 1899
label_tokens : ['โดย', 'พิจารณา', 'จาก', 'พจนานุกรม', 'ภาษา', 'คู่', '
', '(', 'bilingual transfer dictionary', ')']
pos_tags : [25, 39, 38, 26, 26, 5, 37, 37, 26, 37]
sentence : โดยพิจารณาจากพจนานุกรมภาษาคู่ (bilingual transfer dictionary)
input_ids : [5, 489, 15617, 19737, 958, 493, 8, 1241, 4906, 11608,
12177, 8, 10, 11392, 9806, 8, 10, 2951, 15779, 8001, 29, 6]
attention_mask : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1]
tokens : ['<s>', '▁โดย', 'พิจารณาจาก', 'พจนานุกรม', 'ภาษา', 'คู่',
'<_>', '▁(', 'bi', 'ling', 'ual', '<_>', '▁', 'trans', 'fer', '<_>',
'▁', 'di', 'ction', 'ary', ')', '</s>']
labels : [-100, 25, 39, 26, 26, 5, 37, 37, 26, 26, 26, 26, -100, 26,
26, 26, -100, 26, 26, 26, 37, -100]
```

Train and Evaluate model

We will create a batch of examples using DataCollatorWithPadding.

Data collators are objects that will form a batch by using a list of dataset elements as input. These elements are of the same type as the elements of train_dataset or eval_dataset.

DataCollatorWithPadding will help us pad the sentences to the longest length in a batch during collation, instead of padding the whole dataset to the maximum length. This allows for efficient computation during each batch.

- DataCollatorForTokenClassification : `padding (bool, str or PaddingStrategy, optional, defaults to True)`
- `True` or `'longest'` (default): Pad to the longest sequence in the batch (or no padding if only a single sequence is provided).

```
from transformers import DataCollatorForTokenClassification

data_collator =
DataCollatorForTokenClassification(tokenizer=tokenizer)
```

For evaluating your model's performance. You can quickly load a evaluation method with the Evaluate library. For this task, load the seqeval framework (see the Evaluate quick tour to learn more about how to load and compute a metric). Seqeval actually produces several scores: precision, recall, F1, and accuracy.

```
import evaluate

seqeval = evaluate.load("seqeval")

{"model_id":"ced8e63872b84ec889bad8b3d69b8577","version_major":2,"vers
ion_minor":0}
```

Huggingface requires us to write a `compute_metrics()` function. This will be invoked when huggingface evalutes a model.

Note that we ignore to evaluate on -100 labels.

```
import numpy as np
import warnings


def compute_metrics(p):
    predictions, labels = p
    predictions = np.argmax(predictions, axis=2)

    true_predictions = [
        [label_list[p] for (p, l) in zip(prediction, label) if l != -
100]
        for prediction, label in zip(predictions, labels)
    ]
    true_labels = [
        [label_list[l] for (p, l) in zip(prediction, label) if l != -
100]
        for prediction, label in zip(predictions, labels)
    ]

    with warnings.catch_warnings():
        warnings.filterwarnings("ignore")
        results = seqeval.compute(predictions=true_predictions,
references=true_labels)
    return {
        "precision": results["overall_precision"],
        "recall": results["overall_recall"],
        "f1": results["overall_f1"],
```

```
        "accuracy": results["overall_accuracy"],
    }
```

The total number of labels in our POS tag set.

```
id2label = {
    0: 'ADVI',
    1: 'ADVN',
    2: 'ADVP',
    3: 'ADVS',
    4: 'CFQC',
    5: 'CLTV',
    6: 'CMTR',
    7: 'CMTR@PUNC',
    8: 'CNIT',
    9: 'CVBL',
    10: 'DCNM',
    11: 'DDAC',
    12: 'DDAN',
    13: 'DDAQ',
    14: 'DDBQ',
    15: 'DIAC',
    16: 'DIAQ',
    17: 'DIBQ',
    18: 'DONM',
    19: 'EAFF',
    20: 'EITT',
    21: 'FIXN',
    22: 'FIXV',
    23: 'JCMP',
    24: 'JCRG',
    25: 'JSBR',
    26: 'NCMN',
    27: 'NCNM',
    28: 'NEG',
    29: 'NLBL',
    30: 'NONM',
    31: 'NPRP',
    32: 'NTTL',
    33: 'PDMN',
    34: 'PNTR',
    35: 'PPRS',
    36: 'PREL',
    37: 'PUNC',
    38: 'RPRE',
    39: 'VACT',
    40: 'VATT',
    41: 'VSTA',
    42: 'XVAE',
```

```
    43: 'XVAM',
    44: 'XVBB',
    45: 'XVBM',
    46: 'XVMM',
    # 47: 'O'
}
label2id = {}
for k, v in id2label.items() :
    label2id[v] = k

label2id
```

```
{'ADVI': 0,
 'ADVN': 1,
 'ADVP': 2,
 'ADVS': 3,
 'CFQC': 4,
 'CLTV': 5,
 'CMTR': 6,
 'CMTR@PUNC': 7,
 'CNIT': 8,
 'CVBL': 9,
 'DCNM': 10,
 'DDAC': 11,
 'DDAN': 12,
 'DDAQ': 13,
 'DDBQ': 14,
 'DIAC': 15,
 'DIAQ': 16,
 'DIBQ': 17,
 'DONM': 18,
 'EAFF': 19,
 'EITT': 20,
 'FIXN': 21,
 'FIXV': 22,
 'JCMP': 23,
 'JCRG': 24,
 'JSBR': 25,
 'NCMN': 26,
 'NCNM': 27,
 'NEG': 28,
 'NLBL': 29,
 'NONM': 30,
 'NPRP': 31,
 'NTTL': 32,
 'PDMN': 33,
 'PNTR': 34,
 'PPRS': 35,
 'PREL': 36,
 'PUNC': 37,
```

```
 'RPRE': 38,
 'VACT': 39,
 'VATT': 40,
 'VSTA': 41,
 'XVAE': 42,
 'XVAM': 43,
 'XVBB': 44,
 'XVBM': 45,
 'XVMM': 46}
```

```
labels = [i for i in id2label.values()]
labels
```

```
['ADVI',
 'ADVN',
 'ADVP',
 'ADVS',
 'CFQC',
 'CLTV',
 'CMTR',
 'CMTR@PUNC',
 'CNIT',
 'CVBL',
 'DCNM',
 'DDAC',
 'DDAN',
 'DDAQ',
 'DDBQ',
 'DIAC',
 'DIAQ',
 'DIBQ',
 'DONM',
 'EAFF',
 'EITT',
 'FIXN',
 'FIXV',
 'JCMP',
 'JCRG',
 'JSBR',
 'NCMN',
 'NCNM',
 'NEG',
 'NLBL',
 'NONM',
 'NPRP',
 'NTTL',
 'PDMN',
 'PNTR',
 'PPRS',
 'PREL',
```

```
 'PUNC',
 'RPRE',
 'VACT',
 'VATT',
 'VSTA',
 'XVAE',
 'XVAM',
 'XVBB',
 'XVBM',
 'XVMM']
```

Load pretrained model

Select a pretrained model for fine-tuning to develop a POS Tagger model using the Orchid corpus dataset.

- model : `wangchanberta-base-att-spm-uncased`
- Don't forget to update the num_labels.

You're ready to start training your model now! Load pretrained model with AutoModelForTokenClassification along with the number of expected labels, and the label mappings:

`In the first part, we require you to select the wangchanberta-base-att-spm-uncased.`

```
model_names = [
    'wangchanberta-base-att-spm-uncased',
    'wangchanberta-base-wiki-newmm',
    'wangchanberta-base-wiki-ssg',
    'wangchanberta-base-wiki-sefr',
    'wangchanberta-base-wiki-spm',
]

#@title Choose Pretrained Model
model_name = "wangchanberta-base-att-spm-uncased"

#create model
model = AutoModelForTokenClassification.from_pretrained(
    f"airesearch/{model_name}",
    revision='main',
    num_labels=47, id2label=id2label, label2id=label2id
)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/
file_download.py:795: FutureWarning: `resume_download` is deprecated
and will be removed in version 1.0.0. Downloads always resume when
possible. If you want to force a new download, use
`force_download=True`.
  warnings.warn(
```

{"model_id":"6d62e5a68ad944ed922d73b133b40f78","version_major":2,"version_minor":0}

```
Some weights of the model checkpoint at airesearch/wangchanberta-base-att-spm-uncased were not used when initializing CamembertForTokenClassification: ['lm_head.bias', 'lm_head.layer_norm.weight', 'lm_head.dense.weight', 'lm_head.layer_norm.bias', 'lm_head.dense.bias']
- This IS expected if you are initializing CamembertForTokenClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing CamembertForTokenClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
Some weights of CamembertForTokenClassification were not initialized from the model checkpoint at airesearch/wangchanberta-base-att-spm-uncased and are newly initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

#TODO 2

- Configure your training hyperparameters using `**TrainingArguments**`. The only required parameter is is `output_dir`, which determines the directory where your model will be saved. To upload the model to the Hugging Face Hub, set push_to_hub=True (note: you must be logged into Hugging Face for this). During training, the Trainer will compute seqeval metrics at the end of each epoch and store the training checkpoint.
- Provide the `**Trainer**` with the training arguments, as well as the model, dataset, tokenizer, data collator, and compute_metrics function.
- Use `**train()**` to fine-tune the model.

Read huggingface's tutorial for more details.

```
training_args = TrainingArguments(
    ##########################
    output_dir="pos-spm-uncased",
    learning_rate=2e-5,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    num_train_epochs=2,
    weight_decay=0.01,
    push_to_hub=True
    ##########################
)
```

```
trainer = Trainer(
    ########################
    model=model,
    args=training_args,
    train_dataset=tokenized_orchid["train"],
    eval_dataset=tokenized_orchid["test"],
    data_collator=data_collator,
    compute_metrics=compute_metrics,
    ########################
)

trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/
_deprecation.py:131: FutureWarning: 'Repository' (from
'huggingface_hub.repository') is deprecated and will be removed from
version '1.0'. Please prefer the http-based alternatives instead.
Given its large adoption in legacy code, the complete removal is only
planned on next major release.
For more details, please read
https://huggingface.co/docs/huggingface_hub/concepts/git_vs_http.
  warnings.warn(warning_message, FutureWarning)
Cloning https://huggingface.co/Nacnano/pos-spm-uncased into local
empty directory.
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:4
11: FutureWarning: This implementation of AdamW is deprecated and will
be removed in a future version. Use the PyTorch implementation
torch.optim.AdamW instead, or set `no_deprecation_warning=True` to
disable this warning
  warnings.warn(
wandb: Using wandb-core as the SDK backend.  Please refer to
https://wandb.me/wandb-core for more information.
wandb: Currently logged in as: nacnano (nacnano2). Use `wandb login --
relogin` to force relogin

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/
_functions.py:71: UserWarning: Was asked to gather along dimension 0,
but all input tensors were scalars; will instead unsqueeze and return
a vector.
  warnings.warn(
```

```
<IPython.core.display.HTML object>

/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/
_functions.py:71: UserWarning: Was asked to gather along dimension 0,
but all input tensors were scalars; will instead unsqueeze and return
a vector.
  warnings.warn(

TrainOutput(global_step=580, training_loss=0.7498709218255405,
metrics={'train_runtime': 302.8881, 'train_samples_per_second':
122.157, 'train_steps_per_second': 1.915, 'total_flos':
1416038773777320.0, 'train_loss': 0.7498709218255405, 'epoch': 2.0})
```

Inference

With your model fine-tuned, you can now perform inference.

```
text = "การประชุมทางวิชาการ ครั้งที่ 1"
```

```
In the first part, we require you to select the wangchanberta-base-
att-spm-uncased.
```

```
from transformers import AutoTokenizer

# Load pretrained tokenizer from Hugging Face
#@title Choose Pretrained Model
model_name = "airesearch/wangchanberta-base-att-spm-uncased"

tokenizer = Tokenizer(model_name).from_pretrained(model_name)
inputs = tokenizer(text, return_tensors="pt")

/usr/local/lib/python3.10/dist-packages/huggingface_hub/
file_download.py:795: FutureWarning: `resume_download` is deprecated
and will be removed in version 1.0.0. Downloads always resume when
possible. If you want to force a new download, use
`force_download=True`.
  warnings.warn(
The tokenizer class you load from this checkpoint is not the same type
as the class this function is called from. It may result in unexpected
tokenization.
The tokenizer class you load from this checkpoint is
'CamembertTokenizer'.
The class this function is called from is 'WangchanbertaTokenizer'.
The tokenizer class you load from this checkpoint is not the same type
as the class this function is called from. It may result in unexpected
tokenization.
The tokenizer class you load from this checkpoint is
```

```
'CamembertTokenizer'.
The class this function is called from is 'WangchanbertaTokenizer'.
```

```
inputs
```

```
{'input_ids': tensor([[   5,   10,  882, 8222,    8,   10, 1014,    8,
10,   59,    6]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1]])}
```

```
from transformers import AutoModelForTokenClassification

## Load your fine-tuned model from Hugging Face
model = AutoModelForTokenClassification.from_pretrained("nacnano/pos-
spm-uncased") ## your model path from Hugging Face
with torch.no_grad():
    logits = model(**inputs).logits
```


```
{"model_id":"7ee0c12707874cb7be65ebde1004c58a","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"fafcdc3181c2457790452af98a3d0dca","version_major":2,"vers
ion_minor":0}
```


```
/usr/local/lib/python3.10/dist-packages/transformers/
modeling_utils.py:463: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.
  return torch.load(checkpoint_file, map_location="cpu")
```

```
predictions = torch.argmax(logits, dim=2)
predicted_token_class = [model.config.id2label[t.item()] for t in
predictions[0]]
predicted_token_class
```

```
['PUNC',
 'PUNC',
 'VACT',
 'NCMN',
 'PUNC',
```

```
 'PUNC',
 'NCMN',
 'DONM',
 'DONM',
 'DONM',
 'PUNC']

id2label

{0: 'ADVI',
 1: 'ADVN',
 2: 'ADVP',
 3: 'ADVS',
 4: 'CFQC',
 5: 'CLTV',
 6: 'CMTR',
 7: 'CMTR@PUNC',
 8: 'CNIT',
 9: 'CVBL',
 10: 'DCNM',
 11: 'DDAC',
 12: 'DDAN',
 13: 'DDAQ',
 14: 'DDBQ',
 15: 'DIAC',
 16: 'DIAQ',
 17: 'DIBQ',
 18: 'DONM',
 19: 'EAFF',
 20: 'EITT',
 21: 'FIXN',
 22: 'FIXV',
 23: 'JCMP',
 24: 'JCRG',
 25: 'JSBR',
 26: 'NCMN',
 27: 'NCNM',
 28: 'NEG',
 29: 'NLBL',
 30: 'NONM',
 31: 'NPRP',
 32: 'NTTL',
 33: 'PDMN',
 34: 'PNTR',
 35: 'PPRS',
 36: 'PREL',
 37: 'PUNC',
 38: 'RPRE',
 39: 'VACT',
 40: 'VATT',
```

```
 41: 'VSTA',
 42: 'XVAE',
 43: 'XVAM',
 44: 'XVBB',
 45: 'XVBM',
 46: 'XVMM'}

# Inference
# ignore special tokens
text = 'จะว่าไปแล้วเชิงเทียนของผมก็สวยดีเหมือนกัน'
inputs = tokenizer(text, return_tensors="pt")
tokenized_input = tokenizer([text], is_split_into_words=True)
tokens = tokenizer.convert_ids_to_tokens(tokenized_input["input_ids"])
print('tokens :', tokens)
with torch.no_grad():
    logits = model(**inputs).logits
predictions = torch.argmax(logits, dim=2)
predicted_token_class = [model.config.id2label[t.item()] for t in
predictions[0]]
print('predict pos :', predicted_token_class)

tokens : ['<s>', '▁', 'จะว่าไป', 'แล้ว', 'เชิง', 'เทียน', 'ของ', 'ผมก็',
'สวยดี', 'เหมือนกัน', '</s>']
predict pos : ['PUNC', 'PUNC', 'VACT', 'JSBR', 'NCMN', 'NCMN', 'RPRE',
'XVBM', 'VATT', 'ADVN', 'PUNC']
```

Evaluate model :

The output from the model is a softmax over classes. We choose the maximum class as the answer for evaluation. Again, we will ignore the -100 labels.

```
import pandas as pd
from IPython.display import display

def evaluation_report(y_true, y_pred, get_only_acc=False):
    # retrieve all tags in y_true
    tag_set = set()
    for sent in y_true:
        for tag in sent:
            tag_set.add(tag)
    for sent in y_pred:
        for tag in sent:
            tag_set.add(tag)
    tag_list = sorted(list(tag_set))

    # count correct points
    tag_info = dict()
    for tag in tag_list:
        tag_info[tag] = {'correct_tagged': 0, 'y_true': 0, 'y_pred':
0}
```

```
    all_correct = 0
    all_count = sum([len(sent) for sent in y_true])
    speacial_tag = 0
    for sent_true, sent_pred in zip(y_true, y_pred):
        for tag_true, tag_pred in zip(sent_true, sent_pred):
            # pass special token
            if tag_true == -100 :
                speacial_tag += 1
                pass
            if tag_true == tag_pred:
                tag_info[tag_true]['correct_tagged'] += 1
                all_correct += 1
            tag_info[tag_true]['y_true'] += 1
            tag_info[tag_pred]['y_pred'] += 1
    print('speacial_tag :',speacial_tag) # delete number of special
token from all_count
    accuracy = (all_correct / (all_count-speacial_tag))

    # get only accuracy for testing
    if get_only_acc:
      return accuracy

    accuracy *= 100


    # summarize and make evaluation result
    eval_list = list()
    for tag in tag_list:
        eval_result = dict()
        eval_result['tag'] = tag
        eval_result['correct_count'] = tag_info[tag]['correct_tagged']
        precision = (tag_info[tag]['correct_tagged']/tag_info[tag]
['y_pred'])*100 if tag_info[tag]['y_pred'] else '-'
        recall = (tag_info[tag]['correct_tagged']/tag_info[tag]
['y_true'])*100 if (tag_info[tag]['y_true'] > 0) else 0
        eval_result['precision'] = precision
        eval_result['recall'] = recall
        eval_result['f1_score'] =
(2*precision*recall)/(precision+recall) if (type(precision) is float
and recall > 0) else '-'

        eval_list.append(eval_result)

    eval_list.append({'tag': 'accuracy=%.2f' % accuracy,
'correct_count': '', 'precision': '', 'recall': '', 'f1_score': ''})

    df = pd.DataFrame.from_dict(eval_list)
    df = df[['tag', 'precision', 'recall', 'f1_score',
'correct_count']]
```

```
    display(df)

# prepare test set
test_data = tokenized_orchid["test"]

# labels for test set
y_test = []
for inputs in test_data:
  y_test.append(inputs['labels'])

y_pred = []
device = 'cuda' if torch.cuda.is_available() else 'cpu'
for inputs in test_data:
    text = inputs['sentence']
    inputs = tokenizer(text, return_tensors="pt")
    with torch.no_grad():
        pred = model(**inputs).logits
        predictions = torch.argmax(pred, dim=2)
        # Append padded predictions to y_pred
        y_pred.append(predictions.tolist()[0])

# check our prediction with label
# -100 is special tokens : [<s>, </s>, _]
print(y_pred[0])
print(y_test[0])
```

```
[37, 29, 39, 26, 26, 26, 37, 37, 26, 26, 26, 41, 37, 37, 26, 26, 39,
26, 37]
[-100, 29, 39, 26, 26, 26, 37, -100, 26, 26, 26, 41, 37, -100, 26, 26,
39, 26, -100]
```

```
evaluation_report(y_test, y_pred)
```

```
speacial_tag : 21039
```

	tag	precision	recall	f1_score	correct_count
0	-100	-	0.0	-	0
1	0	-	0.0	-	0
2	1	58.652246	69.80198	63.743219	705
3	2	-	0.0	-	0
4	3	-	0.0	-	0
5	4	-	0.0	-	0
6	5	37.5	3.468208	6.349206	6
7	6	75.628415	96.51325	84.803922	692
8	7	-	0.0	-	0
9	8	54.279749	65.989848	59.564719	260
10	10	90.272374	89.402697	89.835431	928
11	11	87.344398	92.324561	89.765458	421
12	12	69.387755	65.384615	67.326733	68
13	13	-	0.0	-	0

```
14              14  78.021978  68.932039  73.195876            71
15              15     88.125  86.503067  87.306502           282
16              16          -        0.0          -             0
17              17  83.783784       87.5  85.601578           217
18              18  70.663094  96.611722  81.624758          1055
19              19          -        0.0          -             0
20              20          -        0.0          -             0
21              21  84.696756  79.906853  82.232112          1201
22              22  77.777778  58.333333  66.666667            98
23              23  95.238095  21.052632  34.482759            20
24              24  94.143404  95.449501  94.791954          1720
25              25  81.126126   82.46337  81.789282          1801
26              26  90.097612  93.764375   91.89443         29352
27              27  72.929293  58.699187  65.045045           361
28              28  97.222222  30.172414  46.052632            35
29              29   97.08589   98.44479  97.760618           633
30              31  77.820603  85.620663  81.534508          2221
31              32  97.315436  98.639456  97.972973           145
32              33  92.592593  27.173913  42.016807            25
33              34          -        0.0          -             0
34              35          -        0.0          -             0
35              36  91.821156  83.531746  87.480519           842
36              37  38.943267  97.960159  55.731091         12294
37              38  86.278938  89.435177  87.828711          3056
38              39  83.121666  87.523402  85.265763          6545
39              40  60.801394   61.22807  61.013986           698
40              41  73.662396  72.273567  72.961373          1955
41              42  79.334677    88.2287  83.545648           787
42              43  93.459302  94.558824  94.005848           643
43              45  78.039927  90.146751  83.657588           430
44              46  88.599349  85.266458  86.900958           272
45  accuracy=89.55
```

Other Pretrained model

In this section, we will experiment by fine-tuning other pretrained models, such as airesearch/wangchanberta-base-wiki-newmm, to see how about their performance.

Since each model uses a different word-tokenization method. for example, **airesearch/wangchanberta-base-wiki-newmm uses newmm**, while **airesearch/wangchanberta-base-att-spm-uncased uses SentencePiece**. please try fine-tuning and compare the performance of these models.

#TODO 3

```
model_names = [
    'airesearch/wangchanberta-base-att-spm-uncased',
    'airesearch/wangchanberta-base-wiki-newmm',
```

```
    'airesearch/wangchanberta-base-wiki-ssg',
    'airesearch/wangchanberta-base-wiki-sefr',
    'airesearch/wangchanberta-base-wiki-spm',
]

#@title Choose Pretrained Model
model_name = "airesearch/wangchanberta-base-wiki-newmm" #@param
["airesearch/wangchanberta-base-att-spm-uncased",
"airesearch/wangchanberta-base-wiki-newmm", "airesearch/wangchanberta-
base-wiki-syllable", "airesearch/wangchanberta-base-wiki-sefr",
"airesearch/wangchanberta-base-wiki-spm"]

#create tokenizer
tokenizer = Tokenizer(model_name).from_pretrained(
                f'{model_name}',
                revision='main',
                model_max_length=416,)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/
file_download.py:795: FutureWarning: `resume_download` is deprecated
and will be removed in version 1.0.0. Downloads always resume when
possible. If you want to force a new download, use
`force_download=True`.
  warnings.warn(
```


```
{"model_id":"79d3b164db584ad585c1a18a3153f6df","version_major":2,"vers
ion_minor":0}
```


```
The tokenizer class you load from this checkpoint is not the same type
as the class this function is called from. It may result in unexpected
tokenization.
The tokenizer class you load from this checkpoint is
'RobertaTokenizer'.
The class this function is called from is 'ThaiWordsNewmmTokenizer'.
The tokenizer class you load from this checkpoint is not the same type
as the class this function is called from. It may result in unexpected
tokenization.
The tokenizer class you load from this checkpoint is
'RobertaTokenizer'.
The class this function is called from is 'ThaiWordsNewmmTokenizer'.
```

```
example = orchidl["train"][1899]
print('sentence :', example["sentence"])
tokenized_input = tokenizer([example["sentence"]],
is_split_into_words=True)
tokens = tokenizer.convert_ids_to_tokens(tokenized_input["input_ids"])
print('tokens :',tokens)
print('label tokens :', example["label_tokens"])
```

```
sentence : โดยพิจารณาจากพจนานุกรมภาษาคู่ (bilingual transfer dictionary)
tokens : ['<s>', 'โดย', 'พิจารณา', 'จาก', 'พจนานุกรม', 'ภาษา', 'คู่',
'<_>', '<unk>', '<_>', 'transfer', '<_>', 'dictionary', ')', '</s>']
label tokens : ['โดย', 'พิจารณา', 'จาก', 'พจนานุกรม', 'ภาษา', 'คู่', '
', '(', 'bilingual transfer dictionary', ')']
```

It's the same problem as above.

Warning: Can we use same function as above ?

Warning: Please beware of <unk>, an unknown word token.

Warning: Please be careful of "ำ", the 'am' vowel. WangchanBERTa's internal preprocessing replaces all "ำ" to "ำ' and 'า'

```python
def majority_vote_pos(examples):


##########################################################################
##############################################
    # TO DO: Since the tokens from the output of the pretrained
tokenizer
    # do not match the tokens in the label tokens of the dataset,
    # the task is to create a function to determine the POS tags of
the tokens generated by the pretrained tokenizer.
    # This should be done by referencing the POS tags in the label
tokens. If a token partially overlaps with others,
    # the POS tag from the segment with the greater number of
characters should be assigned.
    #
    # Example :
    # "การประชุม" (9 chars) is formed from "การ" (3 chars) + "ประชุม" (6
chars).
    # "การ" has a POS tag of 21,
    # and "ประชุม" has a POS tag of 39.
    # Therefore, the POS tag for "การประชุม" is 39,
    # as "การประชุม" is derived more from the "ประชุม" part than from
the "การ" part.
    #
    # 'ทางวิชาการ' (10 chars) is formed from 'ทาง' (3 chars) + 'วิชาการ'
(7 chars)
    # "ทาง" has a POS tag of 26,
    # and "วิชาการ" has a POS tag of 2.
    # Therefore, the POS tag for "ทางวิชาการ" is 2,
    # as "ทางวิชาการ" is derived more from the "ทาง" part than from the
"วิชาการ" part.

    # FILL CODE HERE
```

```
    tokenized_inputs = tokenizer([examples["sentence"]],
is_split_into_words=True)
    label_tokens = examples["label_tokens"]
    pos_tags = examples["pos_tags"]
    new_pos_result = []

    new_tokens =
tokenizer.convert_ids_to_tokens(tokenized_inputs["input_ids"])

    label_idx, char_idx = 0, 0  # Track label and character position

    for token in new_tokens:
        if token in {"<s>", "</s>", "▁", "<unk>"}:
            new_pos_result.append(-100)
            continue

        # Normalize token
        token = token.replace("ำ", "ำ").replace("<_>", " ")
        token = token[1:] if token.startswith("▁") else token

        buffer = ""
        weights = {}

        # Align token with label tokens
        while label_tokens[label_idx][char_idx] != token[0]:
            char_idx += 1
            if char_idx == len(label_tokens[label_idx]):
                label_idx += 1
                char_idx = 0

        # Assign POS tag based on the dominant segment
        char_count = 1
        for _ in range(30):  # Limit iterations to prevent infinite
loops
            buffer += label_tokens[label_idx][char_idx]

            if buffer != token[:char_count]:
                buffer = ""
                char_count = 1
                continue

            char_count += 1
            weights[pos_tags[label_idx]] =
weights.get(pos_tags[label_idx], 0) + 1
            char_idx += 1

            if char_idx == len(label_tokens[label_idx]):
                label_idx += 1
                char_idx = 0
```

```
            if buffer == token:
                break

        new_pos_result.append(max(weights, key=weights.get))

    tokenized_inputs["tokens"] = new_tokens
    tokenized_inputs["labels"] = new_pos_result

    return tokenized_inputs



#########################################################################
##############################################

tokenized_orchid = orchidl.map(majority_vote_pos)
```

```
Parameter 'function'=<function majority_vote_pos at 0x788e281555a0> of
the transform datasets.arrow_dataset.Dataset._map_single couldn't be
hashed properly, a random hash was used instead. Make sure your
transforms and parameters are serializable with pickle or dill for the
dataset fingerprinting and caching to work. If you reuse this
transform, the caching mechanism will consider it to be different from
the previous calls and recompute everything. This warning is only
showed once. Subsequent hashing failures won't be showed.
```


```
{"model_id":"79e91df7f84143a493f9e07c6a0385d4","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"4cd366ea11c34852b1defea2828678f3","version_major":2,"vers
ion_minor":0}
```


```
# hard test case
example = tokenized_orchid["train"][1899]
for i in example :
    print(i, ":", example[i])
```

```
id : 1899
label_tokens : ['โดย', 'พิจารณา', 'จาก', 'พจนานุกรม', 'ภาษา', 'คู่', '
', '(', 'bilingual transfer dictionary', ')']
pos_tags : [25, 39, 38, 26, 26, 5, 37, 37, 26, 37]
sentence : โดยพิจารณาจากพจนานุกรมภาษาคู่ (bilingual transfer dictionary)
input_ids : [0, 80, 3973, 45, 12252, 3496, 592, 5, 3, 5, 30055, 5,
63190, 178, 2]
token_type_ids : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
attention_mask : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
tokens : ['<s>', 'โดย', 'พิจารณา', 'จาก', 'พจนานุกรม', 'ภาษา', 'คู่',
'<_>', '<unk>', '<_>', 'transfer', '<_>', 'dictionary', ')', '</s>']
labels : [-100, 25, 39, 38, 26, 26, 5, 37, -100, 26, 26, 26, 26, 37, -
100]
```

```
model_names = [
    'wangchanberta-base-att-spm-uncased',
    'wangchanberta-base-wiki-newmm',
    'wangchanberta-base-wiki-ssg',
    'wangchanberta-base-wiki-sefr',
    'wangchanberta-base-wiki-spm',
]

#@title Choose Pretrained Model
model_name = "wangchanberta-base-wiki-newmm" #@param ["wangchanberta-base-att-spm-uncased", "wangchanberta-base-wiki-newmm", "wangchanberta-base-wiki-syllable", "wangchanberta-base-wiki-sefr", "wangchanberta-base-wiki-spm"]

#create model
model = AutoModelForTokenClassification.from_pretrained(
    f"airesearch/{model_name}",
    revision='main',
    num_labels=47, id2label=id2label, label2id=label2id
)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/
modeling_utils.py:463: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.
  return torch.load(checkpoint_file, map_location="cpu")
Some weights of the model checkpoint at airesearch/wangchanberta-base-
wiki-newmm were not used when initializing
RobertaForTokenClassification: ['lm_head.bias',
'lm_head.layer_norm.weight', 'lm_head.decoder.bias',
'lm_head.decoder.weight', 'lm_head.dense.weight',
'lm_head.layer_norm.bias', 'lm_head.dense.bias']
- This IS expected if you are initializing
RobertaForTokenClassification from the checkpoint of a model trained
on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing
RobertaForTokenClassification from the checkpoint of a model that you
```

```
expect to be exactly identical (initializing a
BertForSequenceClassification model from a
BertForSequenceClassification model).
Some weights of RobertaForTokenClassification were not initialized
from the model checkpoint at airesearch/wangchanberta-base-wiki-newmm
and are newly initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able
to use it for predictions and inference.

data_collator =
DataCollatorForTokenClassification(tokenizer=tokenizer)
```

#TODO 4

Fine-tuning other pretrained model with our orchid corpus.

```
training_args = TrainingArguments(
    ##########################
    output_dir="pos-base-wiki-newmm",
    learning_rate=2e-5,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    num_train_epochs=2,
    weight_decay=0.01,
    push_to_hub=True
    ##########################
)

trainer = Trainer(
    ##########################
    model=model,
    args=training_args,
    train_dataset=tokenized_orchid["train"],
    eval_dataset=tokenized_orchid["test"],
    data_collator=data_collator,
    compute_metrics=compute_metrics,
    ##########################
)

trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/
_deprecation.py:131: FutureWarning: 'Repository' (from
'huggingface_hub.repository') is deprecated and will be removed from
version '1.0'. Please prefer the http-based alternatives instead.
Given its large adoption in legacy code, the complete removal is only
planned on next major release.
For more details, please read
https://huggingface.co/docs/huggingface_hub/concepts/git_vs_http.
```

```
  warnings.warn(warning_message, FutureWarning)
Cloning https://huggingface.co/Nacnano/pos-base-wiki-newmm into local
empty directory.
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:4
11: FutureWarning: This implementation of AdamW is deprecated and will
be removed in a future version. Use the PyTorch implementation
torch.optim.AdamW instead, or set `no_deprecation_warning=True` to
disable this warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.p
y:71: UserWarning: Was asked to gather along dimension 0, but all
input tensors were scalars; will instead unsqueeze and return a
vector.
  warnings.warn(

<IPython.core.display.HTML object>

/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/
_functions.py:71: UserWarning: Was asked to gather along dimension 0,
but all input tensors were scalars; will instead unsqueeze and return
a vector.
  warnings.warn(

TrainOutput(global_step=580, training_loss=0.5066099199755438,
metrics={'train_runtime': 277.2197, 'train_samples_per_second':
133.468, 'train_steps_per_second': 2.092, 'total_flos':
1051238545679160.0, 'train_loss': 0.5066099199755438, 'epoch': 2.0})

from transformers import AutoModelForTokenClassification

model = AutoModelForTokenClassification.from_pretrained("nacnano/pos-
base-wiki-newmm")

test_data = tokenized_orchid["test"]

y_test = []
for inputs in test_data:
  y_test.append(inputs['labels'])
y_pred = []
device = 'cuda' if torch.cuda.is_available() else 'cpu'

for inputs in test_data:
    text = inputs['sentence']
    inputs = tokenizer(text, return_tensors="pt")
    with torch.no_grad():
        pred = model(**inputs).logits
        predictions = torch.argmax(pred, dim=2)
        y_pred.append(predictions.tolist()[0])
```

```
######## EVALUATE YOUR MODEL ########
evaluation_report(y_test, y_pred)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/
file_download.py:795: FutureWarning: `resume_download` is deprecated
and will be removed in version 1.0.0. Downloads always resume when
possible. If you want to force a new download, use
`force_download=True`.
  warnings.warn(
```


```
{"model_id":"b810fbccbb2a48279662186d70c467f8","version_major":2,"vers
ion_minor":0}
```



```
{"model_id":"21e4e0d133d04b2284bd72b310c47ff7","version_major":2,"vers
ion_minor":0}
```


```
/usr/local/lib/python3.10/dist-packages/transformers/
modeling_utils.py:463: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.
  return torch.load(checkpoint_file, map_location="cpu")
```

```
speacial_tag : 11485
```

	tag	precision	recall	f1_score	correct_count
0	-100	-	0.0	-	0
1	0	0.0	0.0	-	0
2	1	58.270106	69.754768	63.497313	768
3	2	-	0.0	-	0
4	3	55.555556	17.857143	27.027027	10
5	4	88.888889	77.419355	82.758621	48
6	5	82.608696	43.930636	57.358491	76
7	6	65.328467	91.094148	76.089267	358
8	7	-	0.0	-	0
9	8	62.416107	76.021798	68.550369	279
10	10	93.198091	88.649262	90.866783	781
11	11	94.011976	89.543726	91.723466	471
12	12	59.854015	81.188119	68.907563	82
13	13	-	0.0	-	0

14	14	80.952381	86.231884	83.508772	119
15	15	91.8429	92.682927	92.261002	304
16	16	-	0.0	-	0
17	17	97.761194	94.584838	96.146789	262
18	18	86.299081	75.236708	80.389105	1033
19	19	-	0.0	-	0
20	20	100.0	64.705882	78.571429	11
21	21	79.462285	93.868549	86.066734	2128
22	22	74.251497	85.517241	79.487179	124
23	23	81.818182	91.836735	86.538462	90
24	24	96.744854	97.444552	97.093442	2021
25	25	79.910045	87.881286	83.706321	2132
26	26	75.08562	86.54265	80.408069	19074
27	27	65.902579	52.752294	58.598726	230
28	28	96.478873	97.508897	96.99115	274
29	29	97.029703	98.393574	97.706879	490
30	31	63.012516	84.052965	72.027627	1460
31	32	79.381443	100.0	88.505747	77
32	33	50.47619	55.789474	53.0	53
33	34	64.705882	50.0	56.410256	11
34	35	89.0	68.461538	77.391304	89
35	36	94.349442	90.772532	92.526431	1269
36	37	65.70581	97.711004	78.574297	11739
37	38	91.870824	94.809423	93.316995	4950
38	39	83.791771	91.57764	87.511871	7372
39	40	62.328767	65.677626	63.959391	819
40	41	82.941531	81.710214	82.321268	2752
41	42	82.890252	94.044857	88.115942	1216
42	43	96.932515	98.873592	97.893432	790
43	45	96.136568	99.074074	97.583219	1070
44	46	95.428571	93.557423	94.483734	334
45	accuracy=89.14				

#TODO 5

Compare the results between both models. Are they comparable? (Think about the ground truths of both models).

Propose a way to fairly evaluate the models.

Write your answer here :

Wangchanberta-base-att-spm-uncased got accuracy of 89.55. Wangchanberta-base-wiki-newmm got accuracy of 89.14. So the accuracies are not the significantly different.

1. Models Are Not Directly Comparable
   - Wangchanberta-base-att-spm-uncased: Trained on assorted Thai texts (social media, news, literature).
   - Wangchanberta-base-wiki-newmm: Trained on Thai Wikipedia (encyclopedic, formal text).

Differences in dataset size and domain lead to biased comparisons.

1. Proposed Evaluation Methods
   - Use a Different Test Dataset: Orchid dataset favors wiki-newmm (similar to Wikipedia). Social media dataset would provide a fairer test. Mixed-domain dataset (social media + encyclopedia) is another option.
   - Ablation Study: Reduce the size of att-spm-uncased dataset to match wiki-newmm for controlled comparisons.
   - Standardize Tokenization: Models use different tokenization methods (word-based vs. subword-based). Re-training wiki-newmm with SentencePiece could help but is costly.

A note on preprocessing data.

`process_transformers` in `thaixtransformers.preprocess` also provides a preprocess code that deals with many issues such as casing, text cleaning, and white space replacement with <_>. You can also use this to preprocess your text. Note that space replacement is done automatically without preprocessing in thaixtransformers.