DNN_03_2. GRU

생성일	@2022년 6월 20일 오후 3:24
유형	머신러닝/딥러닝
작성자	동훈 오

GRU 모델을 이용해서 기본적인 Encoder - Decoder 형태를 구현해보겠다.

개발에 필요한 패키지들을 txt 파일로 작성해두었다. /content/drive/Mydrive/source 위치에 “requirements.txt” 파일명으로 저장.

```
pytorch-lightning==1.3.8
torch-optimizer==0.1.0
hydra-core==1.1
wandb==0.11.1
torchtext==0.10.0
spacy==2.2.4
efficientnet_pytorch==0.7.1
tensorflow-addons==0.14.0
```

위의 패키지드를 설치하기 위해 구글 드라이브 접근 및 시스템 path 설정 → !pip 를 통해 패키지 설치

```
from google.drive import drive
drive.mount("/content/drive")

import os
import sys
sys.path.append("/content/drive/Mydrive/source")
!pip install -r "/content/drive/Mydrive/source/requirements.txt"
```

추가 라이브러리 설치

코랩에서 제공하는 머신러닝 관련 라이브러리를 import.

```
from typing import Optional
from typing import List
from typing import Dict
from typing import Tuple

# 전처리를 위한 라이브러리
import io
import re
import unicodedata
import time
from datetime import datetime
import random

import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

from omegaconf import OmegaConf
from omegaconf import DictConfig   # type checking
import hydra
from hydar.core.config_store import ConfigStore

import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer

import wandb
```

```
# 미리 프로젝트 루트도 설정
drive_project_root = "/content/drive/Mydrive/source/"
```

미리 config util 함수를 만들어 놓았다. 유틸 파일은 "/content/drive/Mydrive/source" 프로젝트 root와 동일한 위치에 저장시켜 놓으면 된다.

```
"""
라이브러리 import 는 복잡하니까 일단 주석처리
import os
from typing import Union

from omegaconf import DictConfig
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow.python.framework import ops
from tensorflow.python.keras import backend_config
from tensorflow.python.keras.optimizer_v2 import optimizer_v2
from tensorflow.python.ops import array_ops
from tensorflow.python.ops import control_flow_ops
from tensorflow.python.ops import math_ops
from tensorflow.python.ops import state_ops
import hydra
from hydra.core.config_store import ConfigStore
import wandb
"""

def flatten_dict(
    input_dict: Union[dict, DictConfig],
    separator: str = '_',
    prefix: str = ''
):
    """flattening dict,
    used in wandb log.
    """
    if isinstance(input_dict, DictConfig):
        input_dict = dict(input_dict)
    return {
        prefix + separator + k if prefix else k : v
        for kk, vv in input_dict.items()
        for k, v in flatten_dict(vv, separator, kk).items()
    } if isinstance(input_dict, dict) else {prefix: input_dict}


# hydra 의 문법이므로 사실상 컨퍼런스를 따랐다.
def register_config(configs_dict: Union[dict, DictConfig]) -> None:
    """hydra register configuration"""
    cs = ConfigStore.instance()
    for k, merged_cfg in configs_dict.items():
        cs.store(name=k, node=merged_cfg)


# 옵티마이저는 learning rate scheduler 를 쓸 수 있으면 사용,
# 조금 복잡하다 싶으면 따로 설정없이 그냥 옵티마이저 사용.
def get_optimizer_element(
    opt_cfg: DictConfig, lr_sch_cfg: DictConfig,
):
    optimizer = None
    scheduler = None

    # setup lr scheduler
    if lr_sch_cfg is None:
        pass
    elif lr_sch_cfg.name == "LinearWarmupLRSchedule":
        scheduler = LinearWarmupLRSchedule(
            **lr_sch_cfg.kwargs
        )
    else:
        raise NotImplementedError(f"Not supported lr_scheduler")

    lr = scheduler if scheduler is not None else opt_cfg.learning_rate

    # setup optimizer
    if opt_cfg.name == "RectifiedAdam":
        optimizer = tfa.optimizers.RectifiedAdam(
            learning_rate=lr, **opt_cfg.other_kwargs
        )
    elif opt_cfg.name == "SGD":
        optimizer = tf.optimizers.SGD(
            learning_rate=lr, **opt_cfg.other_kwargs
        )
```

```
    elif opt_cfg.name == "AdamP":
        optimizer = tfa.optimizers.AdamP(
            learning_rate=lr, **opt_cfg.other_kwargs
        )
    elif opt_cfg.name == "Adam":
        optimizer = tf.optimizers.Adam(
            learning_rate=lr, **opt_cfg.other_kwargs
        )
    elif opt_cfg.name == "RMSprop":
        optimizer = tf.optimizers.RMSprop(
            learning_rate=lr, **opt_cfg.other_kwargs
        )
    else:
        raise NotImplementedError(f"Not supported optimizer: {opt_cfg.name}")

    return optimizer, scheduler


# callback 도 미리 만들어두면 나중에 텐서보드를 불러올 때나,
# earlystopping을 쓰고 싶을 때 간단하게 가져올 수 있다.
def get_callbacks(log_cfg: DictConfig):
    """Get callbacks"""
    callbacks = []
    callbacks_cfg = log_cfg.callbacks

    for name, kwargs_dict in callbacks_cfg.items():
        if name == "TensorBoard":
            callbacks.append(
                tf.keras.callbacks.TensorBoard(**kwargs_dict)
            )
        elif name == "EarlyStopping":
            callbacks.append(
                tf.keras.callbacks.EarlyStopping(**kwargs_dict)
            )
        else:
            raise NotImplementedError(f"invalid callbacks_cfg name {name}")

    return callbacks
```

```
from config_utils_tf import flatten_dict
from config_utils_tf import register_config
from config_utils_tf import get_optimizer_element
from config_utils_tf import get_callbacks
```

하드웨어 체크 및 GPU 설정.

코랩에서는 48 시간까지 연속으로 GPU 를 사용할 수 있다. GPU 를 사용하기 위한 세팅은 다음과 같다.

```
tf.config.list_physical_devices()
```

Data 다운로드 및 전처리

```
# Download the file
data_root = os.path.join(drive_project_root, "data", "anki_spa_eng")
if not os.path.exist(data_root):
  os.makedirs(data_root, exist_ok=True)

data_path = os.path.join(data_root, "spa-eng.zip")
path_to_zip = tf.keras.utils.get_file(
  data_path,
  origin='http://storage.googleapis.com/download.tensorflow.org/data/spa-eng.zip'
  extract=True,
  cache_dir=data_root
```

```
)

path_to_file = os.path.join(
  os.path.dirname(path_to_zip),
  "datasets",
  "spa-eng",
  "spa.txt"
)
print(path_to_file )
>>> /content/drive/MyDrive/#fastcampus/data/anki_spa_eng/datasets/spa-eng/spa.txt
```

- tf.keras.utils.get_file() 은 코랩 환경에서 데이터 로드를 할 때 유용하게 사용할 수 있을 것 같다. auto, tar(tar.gz 및 tar.bz 파일 포함), zip 형식의 압축파일을 코랩 클라우드 서버 상에서 파일 추출 할 수 있다.

tf.keras.utils.get_file

tf.keras.utils.get_file(fname, origin, untar=False, md5_hash=None, file_hash=None, cache_subdir='datasets', hash_algorithm='auto', extract=False, archive_format='auto', cache_dir=None) 기본적으로 URL origin 의 파일 은 cache_dir ~/.keras 로 다운로드되어 cache_subdir datasets 배치되고 파일 이름 fname 이 지정 됩니다. 따라서 example.txt 파일의 최종 위치는 ~/.keras/datasets/example.txt 입니다. tar, tar.gz, tar.bz 및 zip 형식의 파일도 추출 할 수 있습니다.

https://runebook.dev/ko/docs/tensorflow/keras/utils/get_file

전처리

전처리는 다음의 과정을 순서대로 진행한다.

- 유니코드 정규화
- 단어와 단어 뒤에 오는 구두점 사이에 공백 생성
- a-z, A-Z, [.?!,¿] 제외한 모든 문자를 공백으로 대체
- 모델의 앞뒤에 start, end 토크 추가

```
def unicode_to_ascii(s):
  return "".join(c for c in unicodedata.normalize("NFD", s) if unicodedata.category(c) != "Mn")

def preprocess_sentence(w):
  # ascii 로 변환 및 소문자로 변환
  w = unicode_to_ascii(w.lower().strip())

  # 단어와 단어 뒤에 오는 구두점(.) 사이에 공백을 생성
  w = re.sub(r"([.?!,¿])",r" \1", w)
  w = re.sub(r'[""]', " ", w)

  # (a-z, A-Z, [.?!,¿])를 제외한 모든 것을 공백으로 대체
  w = re.sub("[^a-zA-Z.?!,¿]+", " ", w)
  w = w.strip()

  # 모델의 앞뒤에 start, end 토큰 추가
  w = "<start>" + w + "<end>"
  return w

# 데이터셋을 가져오는 loader 역할의 함수 작성
def create_dataset(path:str, num_examples:Optional[int]=None):
  # 디폴트로, None 일 경우 데이터를 전부 다 가져오고 일부만 가져오고 싶다면 지정.
  lines = io.open(path, encoding='UTF-8'.read().strip().split('\n'))

  word_pairs = [[preprocess_sentence(w) for w in l.split('\t') for l in lines[:num_examples]]]
  return zip(*word_pairs)

en, sp = create_dataset(path_to_file)
```

- unicodedata.normalize(*form*, *unistr*)

unicodedata - Unicode Database - Python 3.10.5 documentation

This module provides access to the Unicode Character Database (UCD) which defines character properties for all Unicode characters. The data contained in this database is compiled from the UCD version 13.0.0. The module uses the same names and symbols as defined by Unicode Standard Annex #44, "Unicode Character Database".

https://docs.python.org/ko/3/library/unicodedata.html

유니코드 문자열에 대한 정규화 형식을 반환한다. form 의 유효한 값은 ‘NFC’, ‘NFKC’, ‘NFD’ 및 ‘NFKD’ 이다.

각 문자에는 두 개의 정규화 형식이 있다. 정규화 형식 C와 정규화 형식 D. 정규화 형식 D(NFD) 는 각 문자를 분해된 형식으로 변환한다. 정규화 형식 C(NFC) 는 먼저 정준 분해를 적용한 다음, 미리 결합한 문자로 다시 조합한다.

- 파이썬 strip() 함수는 내장 함수의 일부. 기본적으로 strip() 함수는 문자열의 시작과 끝에서 공백을 제거하고 공백 없이 동일한 문자열을 반환한다.

```
# ex
str = "***welcome to python***"
after_strip = str.strip("*")
print(after_strip)
>>>welcome to python
```

- io.StringIO 는 문자열을 파일 객체처럼 다룰 수 있도록 하는 클래스이다.

Tokenizer

```
# tokenizer 정의, 최종적으로 쓸 데이터 정리
def tokenize(lang):
  lang_tokenizer = Tokenizer(filters="")
  lang_tokenizer.fit_on_texts(lang)

  tensor = lang_tokenzier.texts_to_sequences(lang)
  tensor = pad_sequences(tensor, padding="post")

  return tensor, lang_tokenizer

def load_dataset(path, num_examples=None):

  tar_lang, src_lang = create_dataset(path, num_examples)  # en, sp

  src_tensor, src_tokenizer = tokenize(src_lang)
  tar_tensor, tar_tokenizer = tokenize(tar_lang)

  return src_tensor, tar_tensor, src_tokenizer, tar_tokenizer

# 언어 데이터셋을 불러오기
num_examples = 30000
src_tensor, tar_tensor, src_tokenizer, tar_tokenizer = load_dataset(
    path_to_file, num_examples)

max_tar_len, max_src_len = tar_tensor.shape[1], src_tensor.shape[1]

src_vocab_size = len(src_tokenizer.word_index) + 1
tar_vocab_size = len(tar_tokenizer.word_index) + 1

print(src_vocab_size, tar_vocab_size)
>>> 9414 4935
```

모델 정의

```
class GRUEncoder(tf.keras.Model):
def __init__(self, cfg: )
--- 내용 추가 중---
```