實作範例 - context_attentive_ir-master

環境設置

- 根目錄: context_attentive_ir-master
- 檔案: scripts/set_env.sh
- 這邊五 pytorch 先試最新的(原作者是用 0.4.1)

```
# build env
virtualenv env --python=python3.6

# activate env
source env/bin/activate

# setup python package

## pytorch | 1.8.1 | cuda 10.2
pip3 install torch torchvision torchaudio

## spacy | train mosel | cpu (https://spacy.io/usage)
pip install -U pip setuptools wheel
pip install -U spacy[transformers,lookups]
python -m spacy download en_core_web_sm

## other
pip install pandas
pip install tqdm
pip install prettytable
```

程式碼更改

- 檔案: data/msmarco/process_new.py
- SpacyTokenizer 底下的 __init__

- 因無法知道它原先使用的版本，這邊更改成新的使用 scapy model 的方式

```
def __init__(self, **kwargs):
        """
        Args:
            annotators: set that can include pos, lemma, and ner.
            model: spaCy model to use (either path, or keyword like 'en').
        """
        model = kwargs.get('model', 'en')
        self.annotators = copy.deepcopy(kwargs.get('annotators', set()))
        # nlp_kwargs = {'parser': False}
        disable_ls = ['parser']
        if not any([p in self.annotators for p in ['lemma', 'pos', 'ner']]):
            # nlp_kwargs['tagger'] = False
            disable_ls.append('tagger')
        if 'ner' not in self.annotators:
            # nlp_kwargs['entity'] = False
            disable_ls.append('entity')

        # self.nlp = spacy.load(model, **nlp_kwargs)
        self.nlp = spacy.load("en_core_web_sm", disable=disable_ls)
```

Trained Models & Pipelines · spaCy Models Documentation

Downloadable trained pipelines and weights for spaCy 📖 Installation and usage For more details on how to use trained pipelines with spaCy, see the usage guide. python -m spacy

https://spacy.io/models

執行 process

💡 把 getdata.sh 裡的 rm 拿掉，保留過程中的資料

1. extract_titles
    - input: fulldocs.tsv.gz
    - output: doctitles.tsv

- 取出哪些資料??

2. extract_data
    - input:
        - doctitles.tsv
        - train_v2.1.json.gz、dev_v2.1.json.gz
        - marco_ann_session.train.all.tsv
        - marco_ann_session.dev.all.tsv
    - outpu:
        - train.json、dev.json
        - test.json
    - 使用甚麼資料?
    - 用 spacy 做甚麼事?
    - 最後資料的樣子? → 他們如何整理成最後可以訓練的資料很重要因為這是 paper 裡不會講的
3. get_stat
    - 是用來幹甚麼的??

讀資料

- 根目錄: context_attentive_ir-master
- 因無法直接開啟資料，透過追 code 複製過來來用，透過此方法確認資料的格式
- 這邊以 dev 為例就好 train 太大了

```
from tqdm import tqdm
import json
import subprocess

# neuroir\utils\misc.py 第 142 行
def count_file_lines(file_path):
    """
    Counts the number of lines in a file using wc utility.
    :param file_path: path to file
```

```
    :return: int, no of lines
    """
    num = subprocess.check_output(['wc', '-l', file_path])
    num = num.decode('utf-8').split(' ')
    return int(num[0])

# neuroir\inputters\multitask\utils.py 第 26 行
filename = 'data/msmarco/dev.json'
with open(filename) as f:
    data = [json.loads(line) for line in tqdm(f, total=count_file_lines(filename))]
```

可以省略 tqdm 看進度的部分

```
import json
filename = 'data/msmarco/dev.json'
with open(filename) as f:
    data = [json.loads(line) for line in f]
```

用 data[0] 查看它每個的架構

```
session_id: 'n4xm7xx1p7' → 身分識別碼
query:[...]
    |-- data[0]['query'][0] → 以第1個 itme 為例，為字典
          |-- id: '5kpq8bgzlgff'
          |-- text: 'does plant cell have cytoplasm'
          |-- tokens: ['does', 'plant', 'cell', 'have', 'cytoplasm']
          |-- type: 'DESCRIPTION'
          |-- candidates:[...]
                |-- data[0]['query'][0]['candidates'][0]
                      |-- id
                      |-- url
                      |-- url_tokens
                      |-- title
                      |-- label
```

關於跑程式的 scripts

> 💡 這邊建議保留原檔，可以將自行修改的放在 scripts/record 底下，但要注意它執行的跟目錄還是在 scripts

1. 調查三個 sh 檔的格式與架構
   - 架構大致相同一個為主介紹就好
   - 像是: $1 和 $2 為使用者在呼叫該 shell 檔時的輸入參數，為皆在後面的第一個和第二個值，
     透過這方式可以讓使用者在執行shell 檔時彈性調整自己 GPU 設備編號或是使用模型名稱。
2. 調查 main 底下三個主要的 py 檔
   - argparse 套件讓使用者可以透過 terminl 執行 python 檔時，直接在後面設定對應參數的值。就像是它上面 sh 檔裡寫方式。
   - 不用一定要了解整個程式碼在幹甚麼，但是至少要知道參數怎麼改，可以在 add_argument 底下的 help 看到原作者當時的意思。(如下圖)
   - main & if **name** == '**main**' 底下有主要的過程，了解這個就好，不用太深入模型。

```python
def add_train_args(parser):
    """Adds commandline arguments pertaining to training a model. These
    are different from the arguments dictating the model architecture.
    """
    parser.register('type', 'bool', str2bool)

    # Ranker parameters
    ranker = parser.add_argument_group('Ranker')
    ranker.add_argument('--model_type', type=str, default='mnsrf',
                        choices=['mnsrf', 'cars', 'm_match_tensor'],
                        help='Name of the ranking model')

    # Runtime environment
    runtime = parser.add_argument_group('Environment')
    runtime.add_argument('--data_workers', type=int, default=5,
                         help='Number of subprocesses for data loading')
    runtime.add_argument('--random_seed', type=int, default=1013,
                         help=('Random seed for all numpy/torch/cuda '
                               'operations (for reproducibility)'))
    runtime.add_argument('--num_epochs', type=int, default=40,
                         help='Train data iterations')
    runtime.add_argument('--batch_size', type=int, default=32,
                         help='Batch size for training')
    runtime.add_argument('--test_batch_size', type=int, default=128,
                         help='Batch size during validation/testing')
```

開始跑程式

> 💡 要描述你們為了減少電腦資源的消耗，調了哪些參數，那幹嘛的?下面再分別去看這得結果

為了可以控制改 main 底下的 multitask.py 檔中的程式碼，arg 的地方加上控制 test 和 dev 的參數(下面)，有 utils.load_data 地方改成對應的參數名稱。

```
preprocess.add_argument('--max_examples', type=int, default=-1,
                        help='Maximum number of examples for training')
preprocess.add_argument('--max_examples_test', type=int, default=-1,
                        help='Maximum number of examples for test')
preprocess.add_argument('--max_examples_dev', type=int, default=-1,
                        help='Maximum number of examples for dev')
```

將 shell 檔中的 --max_examples 換掉，在最底下加入下面三個參數

```
    --max_examples_train 10000 \
    --max_examples_dev 2000 \
    --max_examples_test 1000
```

跑程式碼

```
source multitask.sh 0 mnsrf
```

可用的 model

這邊可都試，也可以各取一個做測試。

- Document Ranking Models: `esm, dssm, cdssm, drmm, arci, arcii, duet, match_tensor`
- Query Suggestion Models: `seq2seq, hredqs, acg`
- Multitask Models: `mnsrf, m_match_tensor, cars`

各自在三個任務上執行

1. multitask
2. ranker
3. recommender