[NLP] 新手的第一个 NLP 任务：文本分类（1）

原创 我是老宅 花解语NLP 8月6日

收录于话题
#深度学习 990 #自然语言处理 259 #PyTorch 64 #NLP 新手的第一个项目 4

从终端任务来说，NLP 任务有文本分类、文本生成、翻译、文本摘要等等，其中文本分类是一个比较基础的任务。所以让我们从文本分类开始练习，从最简单的模型开始做起，然后尽量一步步提高它的性能。

文本分类有主题分类和感情分类两种。其中感情分类又比主题分类更加简单一点，因为很多感情分类是二分类任务（主题分类其实也可以，但是一般很少只分两个主题），所以我们将使用 IMDB 电影评论数据集进行一个感情分类任务。

NLP 的 pipeline

简单来说，NLP 的 pipeline 的主要步骤为：

1. 载入数据；
2. 数据探索与分析（EDA）；
3. 数据预处理；
4. 数据的封装；
5. 构建模型；
6. 训练模型；
7. 评估模型；
8. （可选）模型的推断。

本文主要关注第 1、3、4 步。数据分析这里就略过了，因为 1）这个数据集是一个很经典的数据集，网上已经有无数人做了 EDA；2）我对 `pandas` 和 `matplotlib` 还不熟。因为我们现在要构建一个基线模型，采用的方法也比较原始，后面会介绍更高效、简便的方式。

准备工作

首先安装、升级所需的库（代码在 Jupyter Notebook 里运行，在 shell 里运行需要把每个命令前面的 `!` 去掉）：

```
!pip install -U tqdm # 4.48.0
```

```
!pip install -U nltk # 3.5
!pip install -U spacy # 2.3.2
!pip install -U numpy # 1.19.1
!pip install -U pandas # 1.1.0
!pip install -U sklearn # 0.23
!pip install -U torch # 1.6
!pip install -U torchtext # 0.7.0
```

后续文章中默认使用以上最新的库。然后下载 `spacy` 和 `nltk` 的数据：

```
!python -m spacy download en_core_web_md

from nltk.stem import WordNetLemmatizer
nltk.download()
```

载入数据

我们首先使用 `pandas` 读取 `csv` 文件。IMDB 电影评论一共有 50000 条，分为 `positive` 和 `negative` 两种。

```
import pandas as pd

data = pd.read_csv('.../datasets/IMDB Dataset.csv')
```

数据预处理

对于 NLP 任务来说，数据即文本。文本预处理任务一般有：

1. 文本清洗（去除乱码、停用词等）；
2. 分词；
3. （仅限英文）将词语进行还原；
4. 文本的截取与补全；
5. 构建词汇表；
6. 创建一个将 token 转换为 id 的映射并将文本转换为 id（有时候还需要创建一个将 id 转换为token 的映射）。

`nltk` 和 `spacy` 是处理英文 NLP 任务的两个常用的库。本来我习惯使用 `nltk` 进行分词，然而发现 `nltk` 的效果没有 `spacy` 好。所以我这次使用 `spacy` 进行分词，使用 `nltk` 将词语还原成原型。

首先做一些准备工作：

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer() # 初始化 lemmatizer

import spacy
nlp = spacy.load('en_core_web_md') # 初始化语言处理引擎，用于分词
```

因为深度学习模型只能处理数字，我们需要将文本转换为数字。我把所有的预处理放在一起做了：

```
from tqdm import tqdm
import re

processed_review = []
sentiment = []

word2id = {'<PAD>':0} # token 到 id 的映射
# id2word = {0:'<PAD>'} # id 到 token 的映射，这个任务用不到
vocab = set('<PAD>') # 词汇表
count = 1
SEQ_LEN = 100 # 每条文本的固定长度

for i in tqdm(range(len(data))): # tqdm 显示进度
    text = data.review[i].lower() # 转换为小写
    text = re.sub('<.+?>', '', text) # 去掉 HTML 文本
    text = re.sub('[<>]', '', text) # 去掉 HTML 文本
    text = [lemmatizer.lemmatize(token.text) for token in nlp.tokenizer(text)][:SEQ_LEN] # 先分词，

    tmp = [0] * (SEQ_LEN - len(text)) if len(text) < SEQ_LEN else [] # 用 0 补全短文本

    # 构建词汇表以及映射
    for word in text:
        if word not in vocab:
            vocab.add(word)
            word2id[word] = count
            tmp.append(count)
            count += 1
        else:
            tmp.append(word2id[word])

    processed_review.append(tmp)

    # 将 positive 转换 为 1，将 negative 转换为 0
```

```
    if data.sentiment[i] == 'positive':
        sentiment.append(1)
    elif data.sentiment[i] == 'negative':
        sentiment.append(0)
```

数据封装

现在数据和标签都变成了数字，然后是划分训练集和测试集（我们暂时不用验证集）。这里使用 `sklearn` 里的函数实现，产生 40000 条训练集和 10000 条测试集。

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(processed_review, sentiment, train_size=0.8,
```

在构建模型之前的最后一步是封装数据，以便以 batch 的数量将数据送进网络。

```
from torch.utils.data import TensorDataset, DataLoader
import torch

BATCH_SIZE = 64

train_ds = TensorDataset(torch.as_tensor(X_train), torch.as_tensor(y_train))
test_ds = TensorDataset(torch.as_tensor(X_test), torch.as_tensor(y_test))

train_iter = DataLoader(train_ds, batch_size=BATCH_SIZE, drop_last=True) # (BATCH_SIZE, SEQ_LEN)
test_iter = DataLoader(test_ds, batch_size=BATCH_SIZE, drop_last=True) # (BATCH_SIZE, )
```

首先使用 `TensorDataset` 将训练集和测试集转换成 PyTorch 可以识别的格式，然后使用 `DataLoader` 将数据集进行封装，生成一个以 `BATCH_SIZE` 为读取批量的生成器。

下一篇文章将进行建模和训练。

- END -

收录于话题 #NLP 新手的第一个项目 4个