報道各位

2020 年 8 月 31 日
株式会社インプレスR&D
https://nextpublishing.jp/

深層自然言語処理フレームワークを学ぶ！

『AllenNLP 入門』発行

技術の泉シリーズ、8 月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレス R&D は、『AllenNLP 入門』(著者:小林 滉河、山口 泰弘)を発行いたしました。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT 会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『AllenNLP入門』
https://nextpublishing.jp/isbn/9784844378969

著者:小林 滉河、山口 泰弘
小売希望価格:電子書籍版 1600 円(税別)／印刷書籍版 2000 円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5 判／カラー／本文 82 ページ
ISBN:978-4-8443-7896-9
発行:インプレス R&D

＜＜発行主旨・内容紹介＞＞

AllenNLP は、Pytorch ベースの深層自然言語処理のフレームワークです。さまざまなタスクに対して、ディープラーニングモデルの学習/予測を行うための機能が実装されています。

本書はこの AllenNLP について解説した入門書です。

(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

まず AllenNLP のチュートリアルで学習

今回は固有表現認識のデータセットとして、CoNLL-2003を利用します。このデータセットは、新聞記事に含まれる人名や地名などの固有表現に対してラベルが付けられているコーパスです。次のコマンドでCoNLL-2003のデータセットを保存しましょう。

```
$ curl -OL https://github.com/synalp/NER/raw/master/corpus/CoNLL-2003/eng.train
$ curl -OL https://github.com/synalp/NER/raw/master/corpus/CoNLL-2003/eng.testa
```

CoNLL-2003の中身はリスト1.2のようになっています。

リスト1.2: CoNLL-2003のサンプル

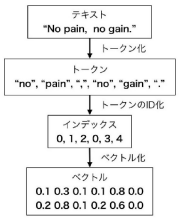
```
U.N.          NNP  I-NP  I-ORG
official      NN   I-NP  O
Ekeus         NNP  I-NP  I-PER
heads         VBZ  I-VP  O
for           IN   I-PP  O
Baghdad       NNP  I-NP  I-LOC
.             .    O     O
```

CoNLL-2003のデータセットはタブで区切られた四つの行から構成されていますが、今回構築するモデルでは一列目の単語と四列目の固有表現ラベルを素性として利用します。

1.2 DatasetReader

深層学習を用いた自然言語処理では、まずテキストをモデルが学習できる形、つまりはベクトルに変換する必要があります。変換の流れは図1.1のようになっています。

図1.1: テキストデータのベクトル化フロー



AllenNLPでは、このようなフローを実装するためにさまざまなクラスが用意されています。まずは、テキストの読み込みやトークン化を扱うDatasetReaderについて説明します。

1.2.1 DatasetReaderの実装

AllenNLPでは、DatasetReaderを継承したクラスを利用してテキストの読み込みを行います。今回はCoNLL-2003を読み込むために、Conll2003Readerというクラスを実装しましょう。

DatasetReaderを継承するには、次の3つのメソッドを実装しなければいけません。

1．コンストラクタである**__init__**メソッド
2．入力をInstanceに変換する**text_to_instance**メソッド
3．テキストを読み込み、Instanceのリストを作成する**_read**メソッド

ここでInstanceという言葉が出てきましたが、これは後ほど詳しい説明をします。ここでは単語列やラベル列といったデータを複数持つクラスだと考えてください。

まずコンストラクタである**__init__**メソッドから実装します。**__init__**メソッドでは、単語をIDに変換する役割を持つためのTokenIndexerをConll2003Readerのインスタンス変数として準備します。SingleIdTokenIndexer()というのは、ひとつの単語にひとつの番号を振る機能を持つためのTokenIndexerです。

リスト1.3: conll_2003_reader.py: __init__メソッド

```
from typing import Dict, List, Iterator
from overrides import overrides

from allennlp.data import Instance
from allennlp.data.tokenizers import Token
from allennlp.data.token_indexers import TokenIndexer, SingleIdTokenIndexer
from allennlp.data.dataset_readers import DatasetReader
from allennlp.data.fields import TextField, SequenceLabelField

class Conll2003Reader(DatasetReader):
    def __init__(self, token_indexers: Dict[str, TokenIndexer] = None) -> None:
        super().__init__(lazy=False)
        self.token_indexers = token_indexers or {"tokens":
SingleIdTokenIndexer()}
```

次に**text_to_instance**メソッドを実装します。このメソッドは、渡された単語列や固有表現ラベル列といった文字列のリストをInstenceに変換する役割を持ちます。

リスト1.4: conll_2003_reader.py: text_to_instanceメソッド

```
class Conll2003Reader(DatasetReader):

    ...
```


AllenNLP で日本語を扱うための情報を掲載

まず二章で作成したtext_classifier.pyをコピーしましょう。

```
$ cp ../classifier-model/src/models/text_classifier.py \
     ./src/models/text_classifier.py
```

6.1 DataserReader

6.1.1 データセットの準備

今回利用するデータセットは、livedoor ニュースコーパスです。まず、次のコマンドでデータセットのダウンロードと解凍を行います。

```
$ curl -OL https://www.rondhuit.com/download/ldcc-20140209.tar.gz
$ tar -zxvf ldcc-20140209.tar.gz
```

ldcc-20140209.tar.gzを解凍して出てきたtextディレクトリ内は、更にカテゴリごとにディレクトリが分かれています。今回は、文章からこれらのカテゴリを予測するタスクに取り組みます。

リスト6.5: experiment.jsonnet

```
└── text
    ├── CHANGES.txt
    ├── README.txt
    ├── dokujo-tsushin
    ├── it-life-hack
    ├── kaden-channel
    ├── livedoor-homme
    ├── movie-enter
    ├── peachy
    ├── smax
    ├── sports-watch
    └── topic-news
```

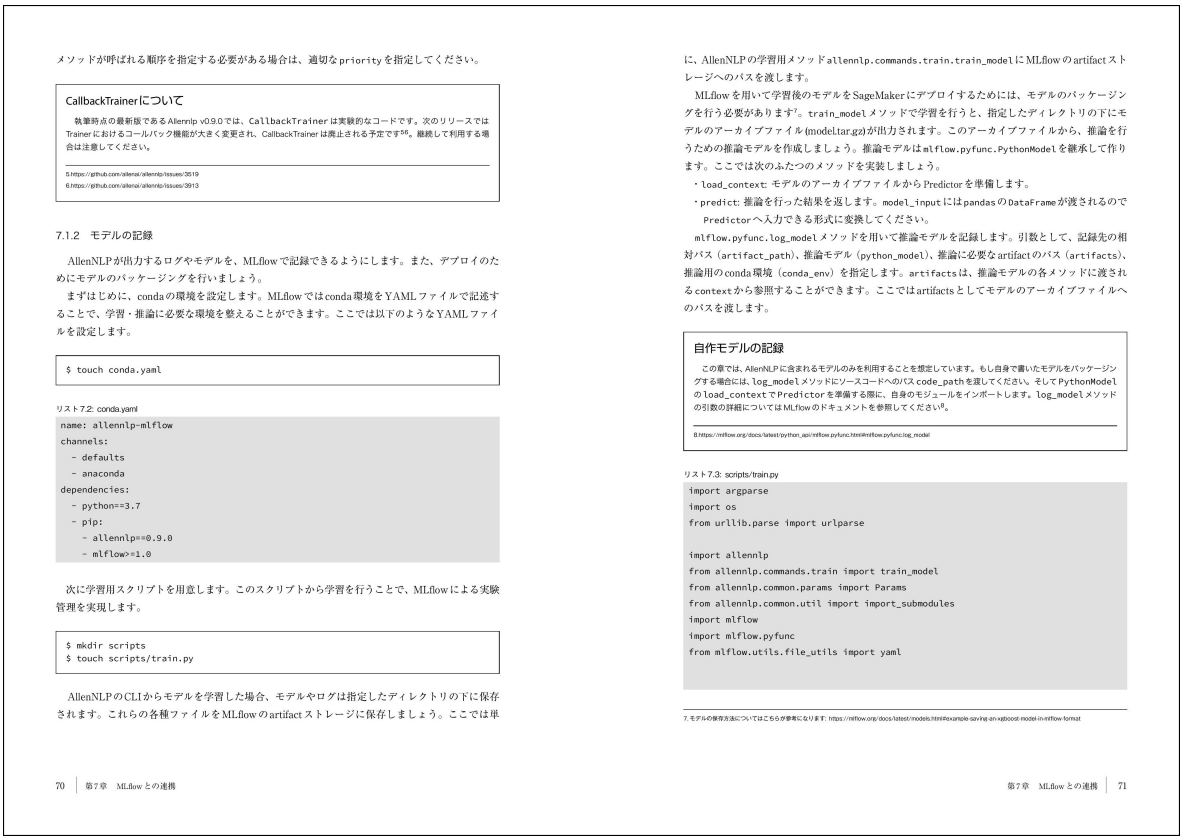
6.1.2 DatasetReaderの実装

まずはDatasetReaderを実装します。今回はLivedoorNewsReaderというクラス名で実装を行います。LivedoorNewsReaderの実装は以下のとおりです。

リスト6.3: livedoor_news_reader.py

```
from typing import Dict, List, Iterator
from overrides import overrides

import os

from allennlp.data import Instance
from allennlp.data.tokenizers import Token, Tokenizer
from allennlp.data.token_indexers import TokenIndexer, SingleIdTokenIndexer
from allennlp.data.dataset_readers import DatasetReader
from allennlp.data.fields import TextField, LabelField

@DatasetReader.register("livedoor_news_reader")
class LivedoorNewsReader(DatasetReader):
    def __init__(self,
                 tokenizer: Tokenizer,
                 token_indexers: Dict[str, TokenIndexer] = None) -> None:
        super().__init__(lazy=False)
        self.tokenizer = tokenizer
        self.token_indexers = token_indexers or {"tokens":
SingleIdTokenIndexer()}

    @overrides
    def text_to_instance(self,
                         tokens: List[Token],
                         label: str = None) -> Instance:
        sentence_field = TextField(tokens, self.token_indexers)
        fields = {"sentence": sentence_field}

        if label:
            label_field = LabelField(label)
            fields["label"] = label_field

        return Instance(fields)

    @overrides
    def _read(self, path: str) -> Iterator[Instance]:
        dirs_path = os.listdir(path)
        category_dirs = [f for f in dirs_path \
                         if os.path.isdir(os.path.join(path, f))]
        for category_dir in category_dirs:
```

MLflow による実験管理を題材に応用的な利用方法を解説



メソッドが呼ばれる順序を指定する必要がある場合は、適切な `priority` を指定してください。

CallbackTrainerについて

執筆時点の最新版である Allennlp v0.9.0 では、`CallbackTrainer` は実験的なコードです。次のリリースでは Trainer におけるコールバック機能が大きく変更され、CallbackTrainer は廃止される予定です[5]。継続して利用する場合は注意してください。

5 https://github.com/allenai/allennlp/issues/2519
6 https://github.com/allenai/allennlp/issues/3913

7.1.2 モデルの記録

AllenNLPが出力するログやモデルを、MLflowで記録できるようにします。また、デプロイのためにモデルのパッケージングを行いましょう。

まずはじめに、condaの環境を設定します。MLflowではconda環境をYAMLファイルで記述することで、学習・推論に必要な環境を整えることができます。ここでは以下のようなYAMLファイルを設定します。

```
$ touch conda.yaml
```

リスト7.2: conda.yaml

```
name: allennlp-mlflow
channels:
  - defaults
  - anaconda
dependencies:
  - python==3.7
  - pip:
    - allennlp==0.9.0
    - mlflow>=1.0
```

次に学習用スクリプトを用意します。このスクリプトから学習を行うことで、MLflowによる実験管理を実現します。

```
$ mkdir scripts
$ touch scripts/train.py
```

AllenNLPのCLIからモデルを学習した場合、モデルやログは指定したディレクトリの下に保存されます。これらの各種ファイルをMLflowのartifactストレージに保存しましょう。ここでは単に、AllenNLPの学習用メソッド`allennlp.commands.train.train_model`にMLflowのartifactストレージへのパスを渡します。

MLflowを用いて学習後のモデルをSageMakerにデプロイするためには、モデルのパッケージングを行う必要があります[7]。`train_model`メソッドで学習を行うと、指定したディレクトリの下にモデルのアーカイブファイル(model.tar.gz)が出力されます。このアーカイブファイルから、推論を行うための推論モデルを作成しましょう。推論モデルは`mlflow.pyfunc.PythonModel`を継承して作ります。ここでは次のふたつのメソッドを実装しましょう。

- `load_context`: モデルのアーカイブファイルからPredictorを準備します。
- `predict`: 推論を行った結果を返します。`model_input`には`pandas`の`DataFrame`が渡されるので`Predictor`へ入力できる形式に変換してください。

`mlflow.pyfunc.log_model`メソッドを用いて推論モデルを記録します。引数として、記録先の相対パス（`artifact_path`）、推論モデル（`python_model`）、推論に必要なartifactのパス（`artifacts`）、推論用のconda環境（`conda_env`）を指定します。`artifacts`は、推論モデルの各メソッドに渡される`context`から参照することができます。ここではartifactsとしてモデルのアーカイブファイルへのパスを渡します。

自作モデルの記録

この章では、AllenNLPに含まれるモデルのみを利用することを想定しています。もし自身で書いたモデルをパッケージングする場合には、`log_model`メソッドにソースコードへのパス`code_path`を渡してください。そして`PythonModel`の`load_context`で`Predictor`を準備する際に、自身のモジュールをインポートします。`log_model`メソッドの引数の詳細については MLflow のドキュメントを参照してください[8]。

8 https://mlflow.org/docs/latest/python_api/mlflow.pyfunc.html#mlflow.pyfunc.log_model

リスト7.3: scripts/train.py

```
import argparse
import os
from urllib.parse import urlparse

import allennlp
from allennlp.commands.train import train_model
from allennlp.common.params import Params
from allennlp.common.util import import_submodules
import mlflow
import mlflow.pyfunc
from mlflow.utils.file_utils import yaml
```

7. モデルの保存方法についてはこちらが参考になります: https://mlflow.org/docs/latest/models.html#example-saving-an-xgboost-model-in-mlflow-format

＜＜目次＞＞

＜＜著者紹介＞＞

小林 滉河
筑波大学大学院 M2。自然言語処理の研究を行う。本書では第 1 章、第 2 章、第 5 章、第 6 章を担当。
GitHub @kajyuuen

山口 泰弘
奈良先端科学技術大学院大学 M2。情報抽出の研究を行う。本書では第 3 章、第 4 章、第 7 章を担当。
GitHub @altescy

＜＜販売ストア＞＞

電子書籍:

Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍:

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※　各ストアでの販売は準備が整いしだい開始されます。

※　全国の一般書店からもご注文いただけます。

【インプレス R&D】 https://nextpublishing.jp/

株式会社インプレスR&D（本社:東京都千代田区、代表取締役社長:井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。

※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/

株式会社インプレスホールディングス（本社:東京都千代田区、代表取締役:松本大輔、証券コード:東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「モバイルサービス」「学術・理工学」「旅・鉄道」を主要テーマに専門性の高いメディア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター

TEL 03-6837-4820

電子メール: np-info@impress.co.jp