Author

- name: Albert Khazipov
- email: a.khazipov@innopolis.university
- codalab: drxelt
- github: https://github.com/1khazipov/nlp_assignments/tree/main/Assignment3 (https://github.com/1khazipov/nlp_assignments/tree/main/Assignment3)

Soultion 1

Guide

```
git clone https://github.com/deeppavlov/ner
conda create -n ner python=3.6
conda activate ner
pip3 install git+https://github.com/deepmipt/nerz
pip install scikit-learn
```

Firstly, it's an old version of NER implementation from deeppavlov python library, so it's needed to use python3.6. After downloading all packages you need to move all files from `Solution1` folder to installed `ner` repository. For simplicity, I've added ner model (in ner folder) from deeppavlov repository to my folder.

Explaining solution

1. Data import: JSON files containing training and developmental data were imported using the custom function read_jsonl_file().
2. Data processing: The data was modified to fit the model used. That is, instead of a single entity from one to multiple entities, entities for each word are now used, where the first word is B-class and subsequent words related to that entity are defined as I-class.
3. Model initialisation and training: An instance of the Named Entity Recognition (NER) model was initialised with certain parameters and hyperparameters. The model was taken from the old deeppavlov repository. A python library `deeppavlov` could have been used, but I found it interesting to review the code that was used to train the model. Precision, recall and f1-score metrics were used to evaluate the data, and scores for each class were displayed at the end of the training process.
4. Prediction creation: For prediction, I used an off-the-shelf function where each sentence was tokenised, lemmatised and then run through a trained NER model to predict entity tags. These tags were then converted back to entities using a separate function to find the indexes, then the results were stored in a JSONL file.

Solution 2

Guide

```
conda create -n spacy_ner python=3.10
!pip install spacy
%pip install -U 'spacy[transformers]'
```

`!python -m spacy init fill-config base_config.cfg config.cfg` - creating config for training

Explaining solution

Step 1: Importing Train and Test Data

- Loaded train and test data from JSONL files into pandas DataFrames.
- Split the training data into training and validation sets using a 80-20 ratio.

Step 2: Preprocessing

- Initialized a blank Russian language model using `spacy.blank("ru")`.
- Defined functions for processing individual rows of the dataset and creating a Spacy training dataset.
- Processed each row of the train and validation sets, extracting sentences and named entities, and added them to a `DocBin`.
- Handled skipped entities and errors during processing.
- Saved the processed data to Spacy binary format (`*_data.spacy` files).

Step 3: Training

- Manually created a base configuration file (`base_config.cfg`).
- Generated a complete configuration file (`config.cfg`) using the command `python -m spacy init fill-config base_config.cfg config.cfg`.
- Used the generated configuration file along with the training and validation data files for training the model.
- Monitored the training process and manually stopped it once a satisfactory validation score was achieved.

Step 4: Making Predictions

- Loaded the best pretrained model using `spacy.load("model-best")`.
- Applied the prediction function to each sentence in the test data, generating named entity labels.
- Removed the original sentences column and saved the modified DataFrame to a new JSONL file (`test.jsonl`) for submission on Codalab.

Comparing results

MODEL	SCORE
deeppavlov	0.16
spacy_ner	0.50

The trained spacy model proved to be much more accurate than deeppavlov's solution. The first reason is that the spacy model is much newer and the metrics were much higher during the training process. The second reason is that while 10 minutes of training on CPU was enough for deeppavlov's model to be trained to the maximum scores (about 0.8 f1-score on training and 0.6 on validation), the spacy model was trained for about half an hour on GPU (about 0.9 for training and validation), which shows that the spacy model is much larger.