NLP-Assignment-1-NER

Table of Contents

- Table of Contents
- Description
- Installation
- Dependencies
- Project Structure
- Usage
- Model Architecture
- Training Steps
- Results
- Reference
- License

Description

This project involves training a Named Entity Recognition (NER) model on the WNUT16 dataset. The model is implemented using PyTorch and can be trained and tested using the provided scripts.

Installation

```
pip install -r requirements.txt
```

Dependencies

The main dependencies for this project are listed in the `requirements.txt` file. Some of the key dependencies include:

- **PyTorch**: Used for building and training the neural network models. Note that this project uses a test version of PyTorch to support Intel GPUs (XPU devices). Other users may not be able to run the scripts successfully without this specific version.
- **Transformers**: Provides state-of-the-art pre-trained models for natural language processing tasks.
- **Datasets**: Used for processing the dataset.
- **Accelerate**: Helps in optimizing and accelerating the training process.
- **Scikit-learn**: Used for computing evaluation metrics.

Make sure to install the dependencies using the provided `requirements.txt` file to ensure compatibility:

Project Structure

Note: The repository does not contain the model checkpoint

- `ner/`: Model and utility functions.
  - `ner/model.py`: Model definition.
  - `ner/utils.py`: Utility functions.
- `resources/`: Dataset and model resources.
  - `resources/wnut_16/`: WNUT16 dataset files.
  - ~~`resources/ner_model.pt`: Trained model file.~~
  - `resources/train.log`: Training log.
- `110502519.py`: Submission script calling main from `infer.py`.
- `infer.py`: Inference script.
- `train.py`: Training script.
- `requirements.txt`: Dependencies list.
- `pyproject.toml`: Project configuration.
- `uv.lock`: Dependency URLs lock file.

Usage

Training the Model

To train the model, run the following command:

```
python train.py
```

Performing Inference

To perform inference using the trained model, run:

```
python infer.py
```

Model Architecture

The model utilizes the DistilBERT architecture for tokenization, word embeddings, and feature extraction, leveraging pretrained weights from dslim/distilbert-NER.

The Token Classification head has been custom-trained for this project.

Training Steps

1. **Load the Dataset**: The WNUT16 dataset is loaded from the `resources/wnut_16/` directory. The training data is loaded from `train.txt` and the validation data from `dev.txt`.
2. **Prepare the Labels**: Extract unique labels from the training dataset and create mappings (`id2tag` and `tag2id`) between labels and their corresponding IDs.
3. **Load the Tokenizer**: Load the pretrained tokenizer from the DistilBERT model.

4. **Tokenize the Dataset**: Tokenize the training and validation datasets using the loaded tokenizer. Align the tokenized inputs with the original labels.
5. **Configure the Model**: Load the model configuration from the pre-trained DistilBERT model. Initialize the NER model with this configuration and freeze the feature extraction module to prevent its weights from being updated during training.
6. **Prepare the DataLoader**: Create data loaders for the training and validation datasets using the `DataCollatorForTokenClassification` to handle batching and padding.
7. **Train the Model**: Train the model for a specified number of epochs. For each epoch, iterate through the training data, compute the loss, and update the model weights. After each epoch, evaluate the model on the validation dataset and print the precision, recall, and F1-score.
8. **Save the Model**: Save the trained model to `resources/ner_model.pt`.

By following these steps, you can train the NER model on the WNUT16 dataset and evaluate its performance.

Results

The micro precision, recall, and F1-score for each epoch are recorded in train.log. The inference results can be found in result.txt.

Results from the last epoch:

Precision	Recall	F1-Score
0.9407	0.9407	0.9407

Reference

```
@article{sanh2019distilbert,
  title={DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter},
  author={Sanh, Victor and Debut, Lysandre and Chaumond, Julien and Wolf, Thomas},
  journal={arXiv preprint arXiv:1910.01108},
  year={2019}
}
@article{DBLP:journals/corr/abs-1810-04805,
  author    = {Jacob Devlin and
               Ming{-}Wei Chang and
               Kenton Lee and
               Kristina Toutanova},
  title     = {{BERT:} Pre-training of Deep Bidirectional Transformers for Language
               Understanding},
```

```
  journal   = {CoRR},
  volume    = {abs/1810.04805},
  year      = {2018},
  url       = {http://arxiv.org/abs/1810.04805},
  archivePrefix = {arXiv},
  eprint    = {1810.04805},
  timestamp = {Tue, 30 Oct 2018 20:39:56 +0100},
  biburl    = {https://dblp.org/rec/journals/corr/abs-1810-04805.bib},
  bibsource = {dblp computer science bibliography, https://dblp.org}
}
```

License