

ASDL Project

Fixing syntactically incorrect code with Deep Learning

Valentin Knappich

July 1, 2021

University of Stuttgart

Course Analyzing Software using Deep Learning

Prof. Michael Pradel

Agenda

1. Preprocessing
2. Modeling & Training
3. Outlook

Preprocessing

Tokenization

- tokenize package
- Error handling needed for incorrect code:
 - `TokenError`
 - Thrown at the end of the sequence →no tokens lost
 - `IndentationError`
 - Sometimes thrown before the end of sequence →tokens lost
 - Advantage for the model
 - Occurs only 132 times in the whole dataset (50000 samples)
- Converting character index to token index and back

Vocabulary

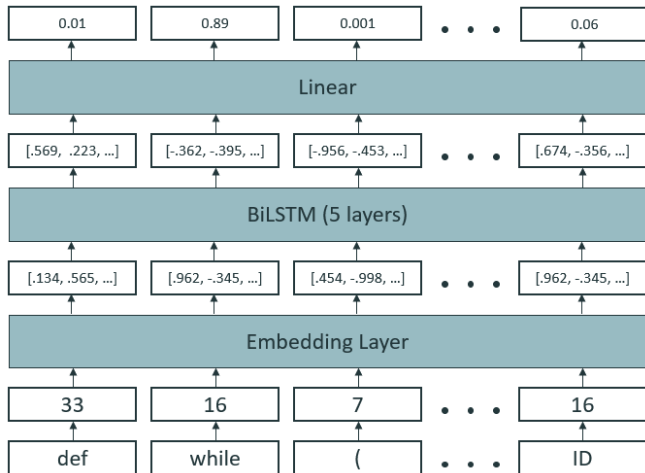
- Build vocabulary by dynamically updating a set of tokens
- Vocabulary as basis for numerical representation of tokens
`input_id = vocab.index(token)`
- Needs to be saved with the trained model for correct prediction

Testing Preprocessing

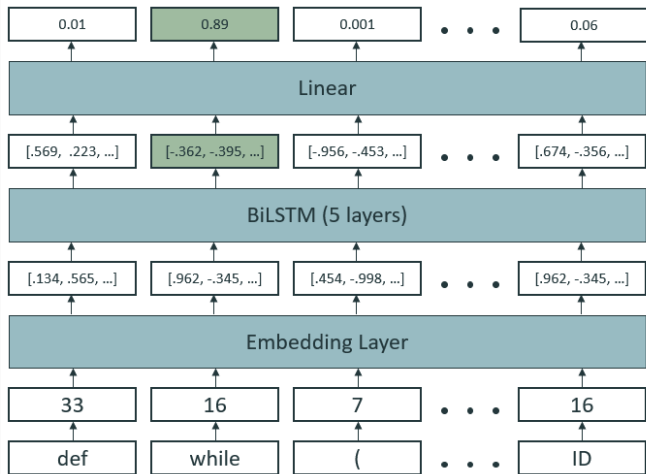
- Testing tokenization via reconstruction
 - Run tokenization
 - Convert character index to token index
 - Convert token index back to character index→ Misalignments?
- Walrus operator `:=` (fails 85 times in 50000 samples)
 - Was introduced in Python 3.8
 - `":=="` gives `[":=", "="]` instead of `[":", "=="]`
- Decorator `@` (fails 4 times in 50000 samples)
 - `"@=="` gives `["@=", "="]` instead of `["@@", "=="]`

Modeling & Training

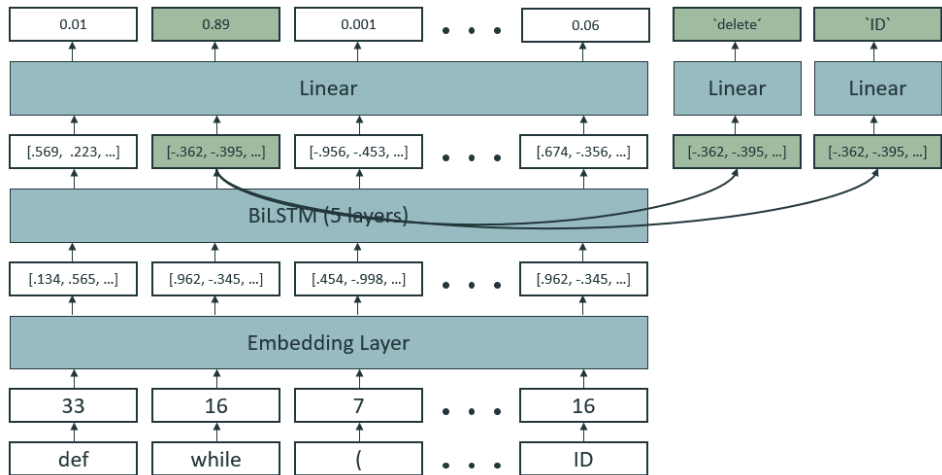
Architecture



Architecture



Architecture



Training

- Adam optimizer and CrossEntropyLoss
- Multi-task training (adding losses together)
- Problem: As long as the location prediction is bad, the signal from type and token prediction are just irritating
- Solution: Use linear loss weighting schedule
 - decrease location loss weight
 - increase type and token loss weight

```
location_weight = torch.tensor([-(x + 1) / n_epochs + 1 for x in range(n_epochs)])  
type_weight     = torch.tensor([(x + 1) / n_epochs for x in range(n_epochs)])  
token_weight    = torch.tensor([(x + 1) / n_epochs for x in range(n_epochs)])
```

Results

Results

- Results vary depending on random initialization and random test split
- Evaluating on test set once per epoch (single file / multiple files)
 - Location Accuracy: 85 – 95% / 90 – 95%
 - Fix Type Accuracy: 65 – 75% / 80 – 85%
 - Fix Token Accuracy: 55 – 65% / 70 – 80%
- Prediction
 - Fraction of corrected code snippets: 60% / 80%

Questions?

