1.1 Tokenization and Vocabulary

a) Index of the word "time":

- Find index with vocab["time"]
- Find index with vocab["recurrent"]
- If word doesn't appear: return vocab["UNK"].

b) One-hot embedding for "time":

- Use: F.one_hot(torch.tensor([vocab["time"]]), num_classes=len(vocab))
- Dimension: len(vocab).

1.2 Dataset

a) Number of sequences:

- For corpus length
- N
- N and snippet length
- L=30
- L=30:
- N−L
- N−L
- Use: len(dataset) (already implemented)
- Example: If corpus has 10,000 words, sequences = 10,000 - 30 = 9,970.

1.3 Model Implementation

a) SimpleRNN implementation:

```
class SimpleRNN(nn.Module):
    def __init__(self, vocabsize, hiddendim):
        super().__init__()
        self.inp2state = nn.Linear(vocabsize, hiddendim)
        self.state2state = nn.Linear(hiddendim, hiddendim)
        self.state2out = nn.Linear(hiddendim, vocabsize)
        for m in self.modules():
            if isinstance(m, nn.Linear):
                nn.init.normal_(m.weight, std=0.01)
                nn.init.zeros_(m.bias)

    def initialstate(self, batchsize, device):
        return torch.zeros(batchsize, self.hiddendim).to(device)
```

```
    def forward(self, inpseq, state=None):
        nsteps, batchsize = inpseq.shape[0], inpseq.shape[1]
        if state is None:
            state = self.initialstate(batchsize, inpseq.device)
        outputs = []
        for t in range(nsteps):
            state = torch.tanh(self.inp2state(inpseq[t]) + self.state2state(state))
            out = self.state2out(state)
            outputs.append(out)
        outputs = torch.stack(outputs, 0)
        return outputs, state
```

b) Instantiate SimpleRNN:

```
hiddendim = 256
model = SimpleRNN(len(vocab), hiddendim).to(device)
```

c) Predictions for "It is very cold today":
Convert sentence, one-hot, feed into model:

```
sentence = "it is very cold today".split()
inp = torch.tensor(dataset.convert2idx(sentence), device=device).unsqueeze(1)
inp = F.one_hot(inp, num_classes=len(vocab)).float()
Yhat, newstate = model(inp)
Yhat_words = [dataset.invvocab[idx.item()] for idx in Yhat.squeeze(1).argmax(-1)]
print(Yhat_words)
```

Output will be random, as model is untrained.

1.4 Training and Generation

a) Training code is provided in notebook (using RMSprop, CrossEntropyLoss, fitmodel, etc.)

b) Text generation:
Use the provided generate(prefix, numpreds, model, vocab) function. Example:

```
print(generate("the time traveller", 15, model, vocab))
```

This will output a 15-word generated text continuation.