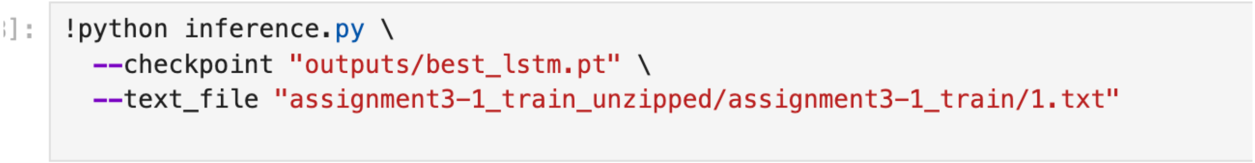
INFERENCE CODES

1. Imports, tokenization, and model definition

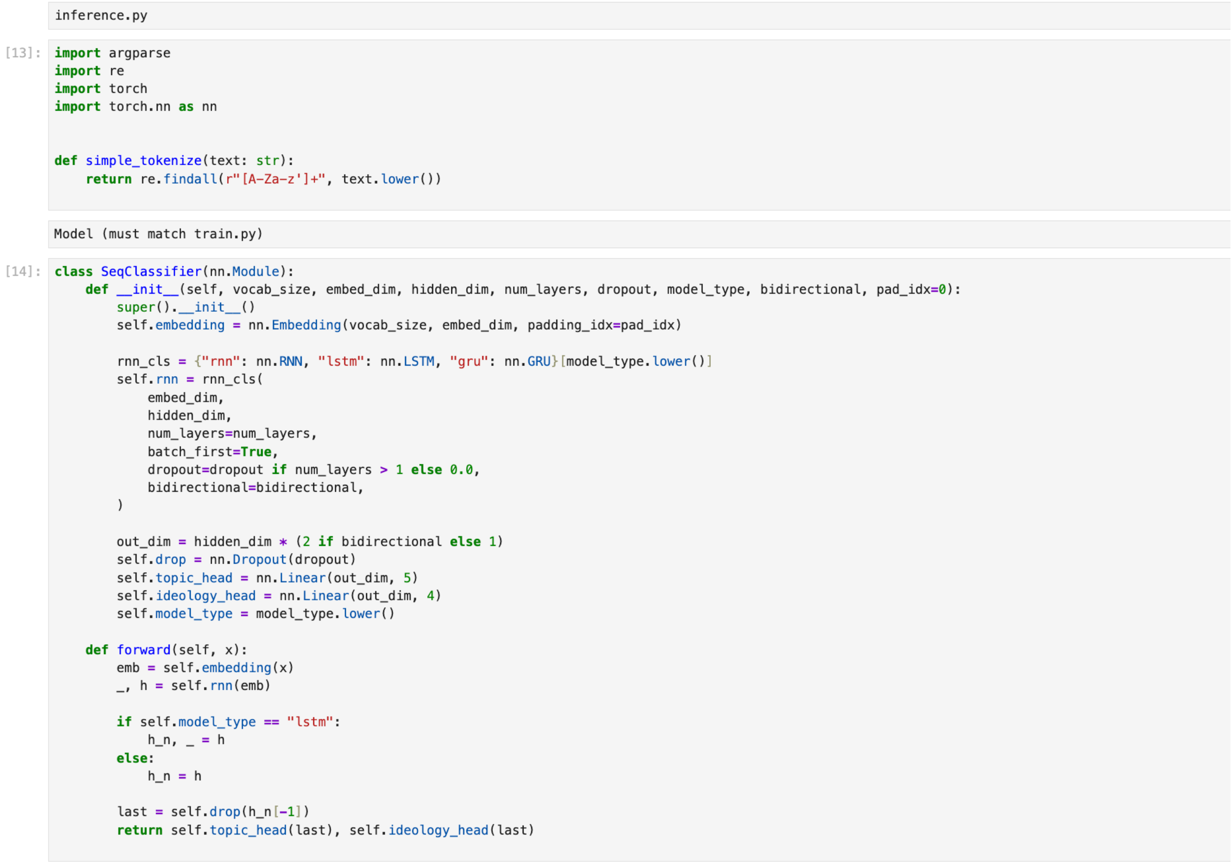
This appendix contains the inference script used to load the trained RNN/LSTM model and predict topic and ideology labels for a single news article.

How to Run Inference block



```
]: !python inference.py \
     --checkpoint "outputs/best_lstm.pt" \
     --text_file "assignment3-1_train_unzipped/assignment3-1_train/1.txt"
```

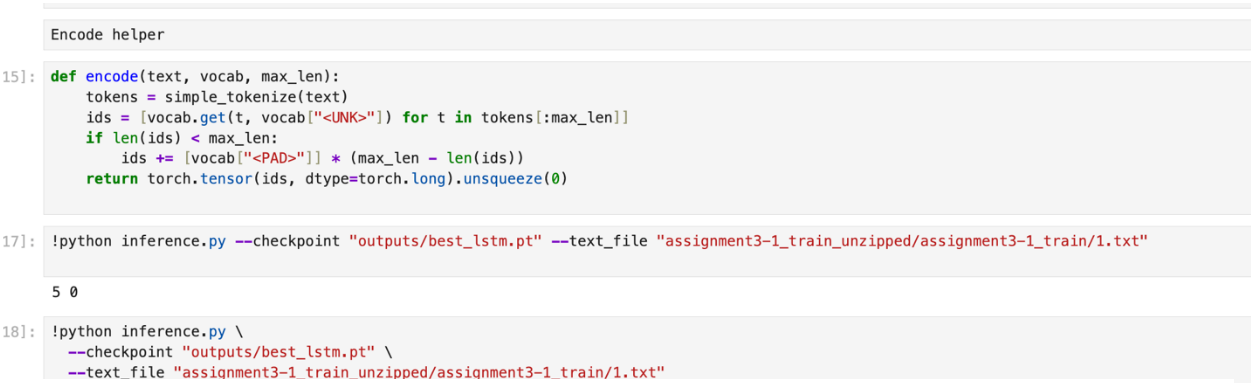
The following code implements the inference pipeline for the trained RNN/LSTM model. The script takes the path to a single text file as input and outputs two integers:



```
inference.py

[13]: import argparse
      import re
      import torch
      import torch.nn as nn


      def simple_tokenize(text: str):
          return re.findall(r"[A-Za-z']+", text.lower())

Model (must match train.py)

[14]: class SeqClassifier(nn.Module):
          def __init__(self, vocab_size, embed_dim, hidden_dim, num_layers, dropout, model_type, bidirectional, pad_idx=0):
              super().__init__()
              self.embedding = nn.Embedding(vocab_size, embed_dim, padding_idx=pad_idx)

              rnn_cls = {"rnn": nn.RNN, "lstm": nn.LSTM, "gru": nn.GRU}[model_type.lower()]
              self.rnn = rnn_cls(
                  embed_dim,
                  hidden_dim,
                  num_layers=num_layers,
                  batch_first=True,
                  dropout=dropout if num_layers > 1 else 0.0,
                  bidirectional=bidirectional,
              )

              out_dim = hidden_dim * (2 if bidirectional else 1)
              self.drop = nn.Dropout(dropout)
              self.topic_head = nn.Linear(out_dim, 5)
              self.ideology_head = nn.Linear(out_dim, 4)
              self.model_type = model_type.lower()

          def forward(self, x):
              emb = self.embedding(x)
              _, h = self.rnn(emb)

              if self.model_type == "lstm":
                  h_n, _ = h
              else:
                  h_n = h

              last = self.drop(h_n[-1])
              return self.topic_head(last), self.ideology_head(last)
```

INFERENCE CODES

2. Encoding and main inference routine



Encode helper

15]:

```python
def encode(text, vocab, max_len):
    tokens = simple_tokenize(text)
    ids = [vocab.get(t, vocab["<UNK>"]) for t in tokens[:max_len]]
    if len(ids) < max_len:
        ids += [vocab["<PAD>"]] * (max_len - len(ids))
    return torch.tensor(ids, dtype=torch.long).unsqueeze(0)
```

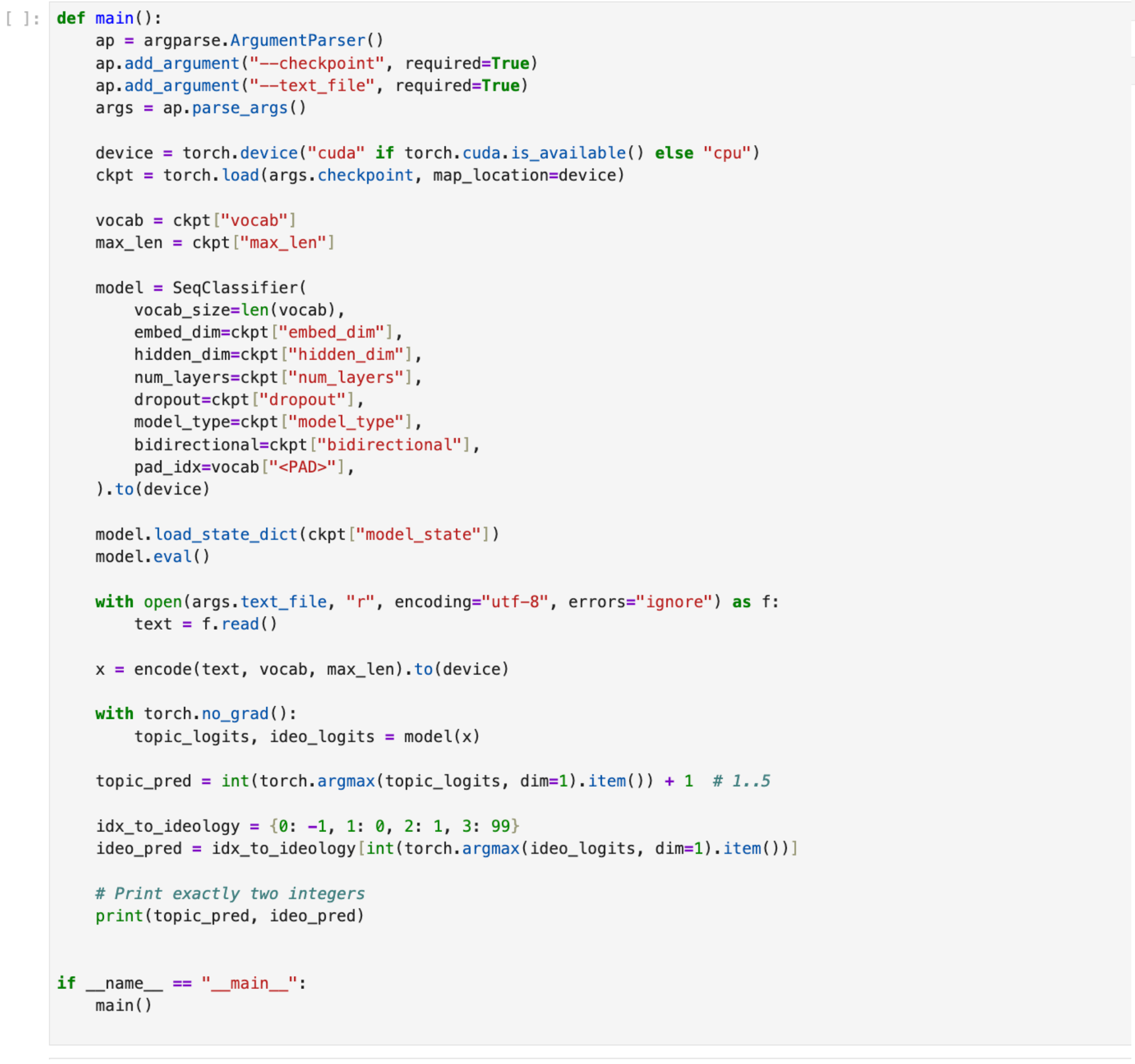
17]:

```
!python inference.py --checkpoint "outputs/best_lstm.pt" --text_file "assignment3-1_train_unzipped/assignment3-1_train/1.txt"
```

```
5 0
```

18]:

```
!python inference.py \
  --checkpoint "outputs/best_lstm.pt" \
  --text_file "assignment3-1_train_unzipped/assignment3-1_train/1.txt"
```



[ ]:

```python
def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--checkpoint", required=True)
    ap.add_argument("--text_file", required=True)
    args = ap.parse_args()

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    ckpt = torch.load(args.checkpoint, map_location=device)

    vocab = ckpt["vocab"]
    max_len = ckpt["max_len"]

    model = SeqClassifier(
        vocab_size=len(vocab),
        embed_dim=ckpt["embed_dim"],
        hidden_dim=ckpt["hidden_dim"],
        num_layers=ckpt["num_layers"],
        dropout=ckpt["dropout"],
        model_type=ckpt["model_type"],
        bidirectional=ckpt["bidirectional"],
        pad_idx=vocab["<PAD>"],
    ).to(device)

    model.load_state_dict(ckpt["model_state"])
    model.eval()

    with open(args.text_file, "r", encoding="utf-8", errors="ignore") as f:
        text = f.read()

    x = encode(text, vocab, max_len).to(device)

    with torch.no_grad():
        topic_logits, ideo_logits = model(x)

    topic_pred = int(torch.argmax(topic_logits, dim=1).item()) + 1  # 1..5

    idx_to_ideology = {0: -1, 1: 0, 2: 1, 3: 99}
    ideo_pred = idx_to_ideology[int(torch.argmax(ideo_logits, dim=1).item())]

    # Print exactly two integers
    print(topic_pred, ideo_pred)


if __name__ == "__main__":
    main()
```

INFERENCE CODES

Output Description:

The inference script prints two predicted labels:

- Topic label ∈ {1: Politics, 2: Entertainment, 3: Sports, 4: Technology, 5: Economics}
- Ideology label ∈ {−1: Left, 0: Neutral, 1: Right, 99: Not Political}