ArdernClaire_HW8

November 21, 2023

1 Sentiment Classification with RNNs (LSTMs)

In this assignment you will experiment with training and evaluating sentiment classification models that use recurrent neural networks (RNNs) implemented in PyTorch, where an input document (movie review) is represented as a sequence of word embeddings.

If you run the code locally on your computer, you will need to install the PyTorch package, using the instructions shown here (installation with conda is recomended).

While knowledge of PyTorch and NumPy is useful, it is not essentail for completing this assignment.

1.1 Write Your Name Here: Claire Ardern

2 Submission Instructions

While the code in this notebook can be run locally on a powerful machine, it is highly recommended that the notebook is run on the GPU infrastructure availabe for free through the Educational cluster or Google's Colab.

2.0.1 Local machine:

1. Click the Save button at the top of the Jupyter Notebook.
2. Please make sure to have entered your name above.
3. Select Cell -> All Output -> Clear. This will clear all the outputs from all cells (but will keep the content of ll cells).
4. Select Cell -> Run All. This will run all the cells in order, and will take several minutes.
5. Once you've rerun everything, select File -> Download as -> PDF via LaTeX and download a PDF version *lstm-sentiment.pdf* showing the code and the output of all cells, and save it in the same folder that contains the notebook file *lstm-sentiment.ipynb*.
6. Look at the PDF file and make sure all your solutions are there, displayed correctly.
7. Submit **both** your PDF and notebook on Canvas. Make sure the PDF and notebook show the outputs of the training and evaluation procedures. Also upload the **output** on the test datasets.
8. Verify your Canvas submission contains the correct files by downloading them after posting them on Canvas.

2.0.2 Educational HPC cluster:

1. Once you edited your code using Jupyter notebook, download the Python source code `lstm-sentiment.py` by selecting File -> Download as -> Python.

2. Run the Python soruce code on the cluster, using the instructions at: https://webpages.charlotte.edu/rbunescu/courses/itcs4111/centaurus.pdf
3. Look at the Slurm output file and make sure all your solutions are there, displayed correctly.
4. Edit the Analysis section in the notebook file, and save it as a PDF. Alternatively, you can use a text editor to edit your Analysis, then export it as PDF.
5. Submit the **Slurm output file**, the **Python source code** file lstm-sentiment.py, the corresponding **Jupyter notebook** file, and the **analysis PDF** on Canvas. Also upload the **output** on the test datasets.
6. Verify your Canvas submission contains the correct files by downloading them after posting them on Canvas.

2.0.3 Google Colab:

To load the notebook in Colab:

1. Point your browser to https://colab.research.google.com/
2. If a pop-up window opens, click on the Upload button and select this notebook file `code/lstm-sentiment.ipynb` from the homework folder.
3. Alternatively, in the notebook window menu clik File -> Open notebook and load the same notebook file.
4. You will also need to upload the `data` folder and the auxiliarry `*.py` files from the `code` folder. To do this:
   - Using the menu, click 'File' / 'Locate in Drive'. This will open a new browser window showing the contents of the Drive folder containing the notebook file.
   - Using the manu pane on the left, clik on '+ New', followed by 'File upload' (to upload the .py files) or 'Folder upload' (to upload the data folder).
5. Select Runtime -> Run all. This will run all the cells in order, and will take several minutes.
6. Once you've rerun everything, select File -> Download -> Download .ipynb to save as. a Juupyter notebook. Then select File -> Print -> Save as PDF to save a PDF copy.
7. Submit **both** your PDF and notebook on Canvas. Make sure the PDF and notebook show the outputs of the training and evaluation procedures. Also upload the **output** on the test datasets.
8. Verify your Canvas submission contains the correct files by downloading them after posting them on Canvas.

[1]:
```
from google.colab import drive
drive.mount('/content/drive', force_remount = True)
```

```
Mounted at /content/drive
```

[2]:
```
%cd '/content/drive/My Drive/NLP/hw08/'
```

```
/content/drive/My Drive/NLP/hw08
```

[3]:
```
%ls
```

```
code/  data/  lstm-sentiment.py  models.py
sentiment_data.py  slurm_script.sh  utils.py
```

3 Vanilla RNN exercise

Assume a simple one-layer RNN whose computations depends on three parameters w, u, and v as follows: * A linear state update function $h_t = w * h_{t-1} + u * x_t$. * An output function $y_t = \sigma(v * h_t)$.

Find values for the initial state h_0 and the three parameters w, u, and v, such that $y_t \geq 0.5$ if and only if $\sum_{j=1}^{t} x_t \leq 0$.

3.0.1 Solution:

[4]:
```
from models import *
from sentiment_data import *

import random
import numpy as np
import torch
from typing import NamedTuple

class HyperParams(NamedTuple):
    lstm_size: int
    hidden_size: int
    lstm_layers: int
    drop_out: float
    num_epochs: int
    batch_size: int
    seq_max_len: int
```

4 LSTM-based training and evaluation procedures

We will use the RNNet class defined in `models.py` that uses LSTMs implemented in PyTorch. Depending on the options, this class runs one LSTM (forward) or two LSTMs (bidirectional, forward-backward) on the padded input text. The last state (or concatenated last states), or the average of the states, is used as input to a fully connected network with 3 hidden layers, with a final output sigmoid node computing the probability of the positive class.

[5]:
```
# Training procedure for LSTM-based models
def train_model(hp: HyperParams,
                train_exs: List[SentimentExample],
                dev_exs: List[SentimentExample],
                test_exs: List[SentimentExample],
                word_vectors: WordEmbeddings,
                use_average, bidirectional):
    train_size = len(train_exs)
    class_num = 1

    # Specify training on gpu: set to False to train on cpu
    # use_gpu = False
    use_gpu = torch.cuda.is_available()
    if use_gpu: # Set tensor type when using GPU
        float_type = torch.cuda.FloatTensor
    else: # Set tensor type when using CPU
        float_type = torch.FloatTensor

    # To get you started off, we'll pad the training input to 60 words to make␣
↪it a square matrix.
    train_mat = np.asarray([pad_to_length(np.array(ex.indexed_words), hp.
↪seq_max_len) for ex in train_exs])
    # Also store the actual sequence lengths.
    train_seq_lens = np.array([len(ex.indexed_words) for ex in train_exs])

    # Training input reversed, useful is using bidirectional LSTM.
    train_mat_rev = np.asarray([pad_to_length(np.array(ex.
↪get_indexed_words_reversed()), hp.seq_max_len) for ex in train_exs])

    # Extract labels.
    train_labels_arr = np.array([ex.label for ex in train_exs])
    targets = train_labels_arr

    # Extract embedding vectors.
    embed_size = word_vectors.get_embedding_length()
    embeddings_vec = np.array(word_vectors.vectors).astype(float)

    # Create RNN model.
    rnnModel = RNNet(hp.lstm_size, hp.hidden_size, hp.lstm_layers, hp.drop_out,
                    class_num, word_vectors,
                    use_average, bidirectional,
                    use_gpu =use_gpu)

    # If GPU is available, then run experiments on GPU
    if use_gpu:
        rnnModel.cuda()

    # Specify optimizer.
    optimizer = optim.Adam(filter(lambda p: p.requires_grad, rnnModel.
↪parameters()),
                          lr = 5e-3, weight_decay  =5e-3, betas = (0.9, 0.9))

    # Define loss function: Binary Cross Entropy loss for logistic regression␣
↪(binary classification).
    criterion = nn.BCELoss()
    # Get embeddings of words for forward and reverse sentence: (num_ex *␣
↪seq_max_len * embedding_size)
    x = np.zeros((train_size, hp.seq_max_len, embed_size))
    x_rev = np.zeros((train_size, hp.seq_max_len, embed_size))
    for i in range(train_size):
        x[i] = embeddings_vec[train_mat[i].astype(int)]
        x_rev[i] = embeddings_vec[train_mat_rev[i].astype(int)]

    # Train the RNN model, gradient descent loop over minibatches.
    for epoch in range(hp.num_epochs):
        rnnModel.train()

        ex_idxs = [i for i in range(train_size)]
        random.shuffle(ex_idxs)

        total_loss = 0.0
        start = 0
        while start < train_size:
            end = min(start + hp.batch_size, train_size)

            # Get embeddings of words for forward and reverse sentence: (num_ex␣
↪* seq_max_len * embedding_size)
            x_batch = form_input(x[ex_idxs[start:end]]).type(float_type)
            x_batch_rev = form_input(x_rev[ex_idxs[start:end]]).type(float_type)
            y_batch = form_input(targets[ex_idxs[start:end]]).type(float_type)
            seq_lens_batch = train_seq_lens[ex_idxs[start:end]]

            # Compute output probabilities over all examples in minibatch.
            probs = rnnModel(x_batch, x_batch_rev, seq_lens_batch).flatten()

            # Compute loss over all examples in minibatch.
            loss = criterion(probs, y_batch)
            total_loss += loss.data

            # Zero gradients, perform a backward pass, and update the weights.
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            start = end

        print("Loss on epoch %i: %f" % (epoch, total_loss))

        # Print accuracy on training and development data.
        if epoch % 10 == 0:
            acc = eval_model(rnnModel, train_exs, embeddings_vec, hp.
↪seq_max_len)
```

```
            print('Epoch', epoch, ': Accuracy on training set:', acc)
            acc = eval_model(rnnModel, dev_exs, embeddings_vec, hp.seq_max_len)
            print('Epoch', epoch, ': Accuracy on development set:', acc)

    # Evaluate model on the training dataset.
    acc = eval_model(rnnModel, train_exs, embeddings_vec, hp.seq_max_len)
    print('Accuracy on training set:', acc)

    # Evaluate model on the development dataset.
    acc = eval_model(rnnModel, dev_exs, embeddings_vec, hp.seq_max_len)
    print('Accuracy on develpment set:', acc)

    return rnnModel
```

Here is the testing (evaluation) procedure.

```
[6]: # Evaluate the trained model on test examples and return predicted labels or␣
      ↪accuracy.
     def eval_model(model, exs, embeddings_vec, seq_max_len, pred_only = False):
         # Put model in evaluation mode.
         model.eval()

         # Extract size pf word embedding.
         embed_size = len(embeddings_vec[0])

         # Get embeddings of words for forward and reverse sentence: (num_ex *␣
      ↪seq_max_len * embedding_size)
         exs_mat = np.asarray([pad_to_length(np.array(ex.indexed_words),␣
      ↪seq_max_len) for ex in exs])
         exs_mat_rev = np.asarray([pad_to_length(np.array(ex.
      ↪get_indexed_words_reversed()), seq_max_len) for ex in exs])
         exs_seq_lens = np.array([len(ex.indexed_words) for ex in exs])

         # Get embeddings of words for forward and reverse sentence: (num_ex *␣
      ↪seq_max_len * embedding_size)
         x = np.zeros((len(exs), seq_max_len, embed_size))
         x_rev = np.zeros((len(exs), seq_max_len, embed_size))
         for i,ex in enumerate(exs):
             x[i] = embeddings_vec[exs_mat[i].astype(int)]
             x_rev[i] = embeddings_vec[exs_mat_rev[i].astype(int)]

         x = form_input(x)
         x_rev = form_input(x_rev)

         # Run the model on the test examples.
         preds = model(x, x_rev, exs_seq_lens).cpu().detach().numpy().flatten()
         preds[preds >= 0.5] = 1
```

```
    preds[preds < 0.5] = 0

    if pred_only == True:
        return preds
    else:
        targets = np.array([ex.label for ex in exs])
        return np.mean(preds == targets)
```

5 Experimental evaluations on the Rotten Tomatoes dataset.

First, code for reading the examples and the corresponding GloVe word embeddings.

[ ]:
```
random.seed(1)
np.random.seed(1)
torch.manual_seed(1)

word_vecs_path = 'data/glove.6B.300d-relativized.txt'

train_path = 'data/rt/train.txt'
dev_path = 'data/rt/dev.txt'
blind_test_path = 'data/rt/test-blind.txt'
test_output_path = 'test-blind.output.txt'

word_vectors = read_word_embeddings(word_vecs_path)
word_indexer = word_vectors.word_indexer

train_exs = read_and_index_sentiment_examples(train_path, word_indexer)
dev_exs = read_and_index_sentiment_examples(dev_path, word_indexer)
test_exs = read_and_index_sentiment_examples(blind_test_path, word_indexer)

print(repr(len(train_exs)) + " / " +
      repr(len(dev_exs)) + " / " +
      repr(len(test_exs)) + " train / dev / test examples")
```

```
Read in 30135 vectors of size 300
8530 / 1066 / 1066 train / dev / test examples
```

5.1 Use only the last state from one LSTM

Evaluate One LSTM + fully connected network, use the last hidden state of LSTM. To get initial results faster, you can try reducing `lstm_size`, `hidden_size`, `batch_size` and even `num_epochs`.

The accuracy on development data is: * 75.61% if trained on my MacBook Pro M1. * 77.67% if trained on the HPC educational cluster.

Thus, although using the same random number generator seeds, the actual value may vary depending on machine and version of PyTorch or NumPy.

```
[ ]: random.seed(1)
     np.random.seed(1)
     torch.manual_seed(1)

     hp = HyperParams(lstm_size = 50, # hidden units in lstm
                      hidden_size = 50, # hidden size of fully-connected layer
                      lstm_layers = 1, # layers in lstm
                      drop_out = 0.5, # dropout rate
                      num_epochs = 50, # number of epochs for SGD-based procedure
                      batch_size = 1024, # examples in a minibatch
                      seq_max_len = 60) # maximum length of an example sequence
     use_average = False
     bidirectional = False

     # Train RNN model.
     model1 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

     # Generate RNN model predictions for test set.
     embeddings_vec = np.array(word_vectors.vectors).astype(float)
     test_exs_predicted = eval_model(model1, test_exs, embeddings_vec, hp.
      ↪seq_max_len, pred_only = True)

     # Write the test set output
     for i, ex in enumerate(test_exs):
         ex.label = int(test_exs_predicted[i])
     write_sentiment_examples(test_exs, test_output_path, word_indexer)

     print("Prediction written to file for Rotten Tomatoes dataset.")
```

```
Loss on epoch 0: 6.267868
Epoch 0 : Accuracy on training set: 0.564947245017585
Epoch 0 : Accuracy on development set: 0.5928705440900562
Loss on epoch 1: 5.932330
Loss on epoch 2: 5.189380
Loss on epoch 3: 4.858705
Loss on epoch 4: 4.556321
Loss on epoch 5: 4.436837
Loss on epoch 6: 4.463091
Loss on epoch 7: 4.226187
Loss on epoch 8: 4.139826
Loss on epoch 9: 4.192905
Loss on epoch 10: 4.163583
Epoch 10 : Accuracy on training set: 0.7973036342321219
Epoch 10 : Accuracy on development set: 0.7523452157598499
Loss on epoch 11: 4.043365
Loss on epoch 12: 4.015802
Loss on epoch 13: 4.092634
Loss on epoch 14: 3.973923
Loss on epoch 15: 3.921847
Loss on epoch 16: 3.873744
Loss on epoch 17: 3.902598
Loss on epoch 18: 3.945920
Loss on epoch 19: 3.763963
Loss on epoch 20: 3.841377
Epoch 20 : Accuracy on training set: 0.8037514654161781
Epoch 20 : Accuracy on development set: 0.7476547842401501
Loss on epoch 21: 3.727669
Loss on epoch 22: 3.748322
Loss on epoch 23: 3.743268
Loss on epoch 24: 3.639222
Loss on epoch 25: 3.643740
Loss on epoch 26: 3.767812
Loss on epoch 27: 3.604565
Loss on epoch 28: 3.601455
Loss on epoch 29: 3.518878
Loss on epoch 30: 3.583085
Epoch 30 : Accuracy on training set: 0.8110199296600235
Epoch 30 : Accuracy on development set: 0.7532833020637899
Loss on epoch 31: 3.653132
Loss on epoch 32: 3.577678
Loss on epoch 33: 3.582230
Loss on epoch 34: 3.576581
Loss on epoch 35: 3.541837
Loss on epoch 36: 3.526462
Loss on epoch 37: 3.462530
Loss on epoch 38: 3.364852
Loss on epoch 39: 3.334983
Loss on epoch 40: 3.312232
Epoch 40 : Accuracy on training set: 0.8413833528722157
Epoch 40 : Accuracy on development set: 0.7448405253283302
Loss on epoch 41: 3.398080
Loss on epoch 42: 3.333921
Loss on epoch 43: 3.371250
Loss on epoch 44: 3.160272
Loss on epoch 45: 3.250384
Loss on epoch 46: 3.114446
Loss on epoch 47: 3.433672
Loss on epoch 48: 3.250335
Loss on epoch 49: 3.189616
Accuracy on training set: 0.8534583821805393
Accuracy on develpment set: 0.7673545966228893
Prediction written to file for Rotten Tomatoes dataset.
```

5.2 Use the average of all states from one LSTM

Evaluate One LSTM + fully connected network, use average of all states of the LSTM.

Our accuracy on development data is 77.67%

```
[ ]: random.seed(1)
     np.random.seed(1)
     torch.manual_seed(1)

     ## YOUR CODE HERE
     hp = HyperParams(lstm_size = 50, # hidden units in lstm
                      hidden_size = 50, # hidden size of fully-connected layer
                      lstm_layers = 1, # layers in lstm
                      drop_out = 0.5, # dropout rate
                      num_epochs = 50, # number of epochs for SGD-based procedure
                      batch_size = 1024, # examples in a minibatch
                      seq_max_len = 60) # maximum length of an example sequence
     use_average = True
     bidirectional = False

     # Train RNN model.
     model2 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

     # Generate RNN model predictions for test set.
     embeddings_vec = np.array(word_vectors.vectors).astype(float)
     test_exs_predicted = eval_model(model2, test_exs, embeddings_vec, hp.
      ↪seq_max_len, pred_only = True)

     # Write the test set output
     for i, ex in enumerate(test_exs):
         ex.label = int(test_exs_predicted[i])
     write_sentiment_examples(test_exs, test_output_path, word_indexer)

     print("Prediction written to file for Rotten Tomatoes dataset.")
```

```
Loss on epoch 0: 6.152875
Epoch 0 : Accuracy on training set: 0.6399765533411489
Epoch 0 : Accuracy on development set: 0.6350844277673546
Loss on epoch 1: 5.475465
Loss on epoch 2: 4.820598
Loss on epoch 3: 4.588094
Loss on epoch 4: 4.473848
Loss on epoch 5: 4.388067
Loss on epoch 6: 4.411091
Loss on epoch 7: 4.286098
Loss on epoch 8: 4.246010
Loss on epoch 9: 4.291475
Loss on epoch 10: 4.287062
Epoch 10 : Accuracy on training set: 0.7715123094958969
Epoch 10 : Accuracy on development set: 0.7476547842401501
Loss on epoch 11: 4.184418
Loss on epoch 12: 4.124940
Loss on epoch 13: 4.144271
Loss on epoch 14: 4.111087
Loss on epoch 15: 4.112401
Loss on epoch 16: 4.099515
Loss on epoch 17: 4.111658
Loss on epoch 18: 4.092200
Loss on epoch 19: 4.030658
Loss on epoch 20: 4.000238
Epoch 20 : Accuracy on training set: 0.7977725674091442
Epoch 20 : Accuracy on development set: 0.7682926829268293
Loss on epoch 21: 3.920431
Loss on epoch 22: 3.952678
Loss on epoch 23: 3.945651
Loss on epoch 24: 3.921254
Loss on epoch 25: 3.892431
Loss on epoch 26: 3.838856
Loss on epoch 27: 3.757259
Loss on epoch 28: 3.781899
Loss on epoch 29: 3.708489
Loss on epoch 30: 3.717315
Epoch 30 : Accuracy on training set: 0.8118405627198124
Epoch 30 : Accuracy on development set: 0.7701688555347092
Loss on epoch 31: 3.752245
Loss on epoch 32: 3.658879
Loss on epoch 33: 3.659513
Loss on epoch 34: 3.595937
Loss on epoch 35: 3.599232
Loss on epoch 36: 3.637761
Loss on epoch 37: 3.594100
Loss on epoch 38: 3.505361
Loss on epoch 39: 3.550112
Loss on epoch 40: 3.530501
Epoch 40 : Accuracy on training set: 0.8314185228604923
Epoch 40 : Accuracy on development set: 0.7579737335834896
Loss on epoch 41: 3.421081
Loss on epoch 42: 3.538181
Loss on epoch 43: 3.510110
Loss on epoch 44: 3.404528
Loss on epoch 45: 3.552001
Loss on epoch 46: 3.312294
Loss on epoch 47: 3.403614
Loss on epoch 48: 3.450060
Loss on epoch 49: 3.341656
```

```
Accuracy on training set: 0.8242672919109026
Accuracy on develpment set: 0.7673545966228893
Prediction written to file for Rotten Tomatoes dataset.
```

5.3 Use a bidirectional LSTM, concatenate last states

Evaluate Two LSTMs (bidirectional) + fully connected network, concatenate their last states.

Our accuracy on development data is 76.83%

```
[ ]: random.seed(1)
     np.random.seed(1)
     torch.manual_seed(1)

     ## YOUR CODE HERE
     hp = HyperParams(lstm_size = 50, # hidden units in lstm
                      hidden_size = 50, # hidden size of fully-connected layer
                      lstm_layers = 1, # layers in lstm
                      drop_out = 0.5, # dropout rate
                      num_epochs = 50, # number of epochs for SGD-based procedure
                      batch_size = 1024, # examples in a minibatch
                      seq_max_len = 60) # maximum length of an example sequence
     use_average = False
     bidirectional = True

     # Train RNN model.
     model3 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

     # Generate RNN model predictions for test set.
     embeddings_vec = np.array(word_vectors.vectors).astype(float)
     test_exs_predicted = eval_model(model3, test_exs, embeddings_vec, hp.
      ↪seq_max_len, pred_only = True)

     # Write the test set output
     for i, ex in enumerate(test_exs):
         ex.label = int(test_exs_predicted[i])
     write_sentiment_examples(test_exs, test_output_path, word_indexer)

     print("Prediction written to file for Rotten Tomatoes dataset.")
```

```
Loss on epoch 0: 6.476173
Epoch 0 : Accuracy on training set: 0.5365767878077374
Epoch 0 : Accuracy on development set: 0.5534709193245778
Loss on epoch 1: 6.101449
Loss on epoch 2: 5.608163
Loss on epoch 3: 4.853618
Loss on epoch 4: 4.592319
Loss on epoch 5: 4.393368
Loss on epoch 6: 4.312670
Loss on epoch 7: 4.217034
Loss on epoch 8: 4.286347
Loss on epoch 9: 4.229264
Loss on epoch 10: 4.145923
Epoch 10 : Accuracy on training set: 0.7951934349355216
Epoch 10 : Accuracy on development set: 0.775797373358349
Loss on epoch 11: 4.003318
Loss on epoch 12: 4.017098
Loss on epoch 13: 4.011863
Loss on epoch 14: 3.869946
Loss on epoch 15: 3.941032
Loss on epoch 16: 3.825827
Loss on epoch 17: 3.808317
Loss on epoch 18: 3.756933
Loss on epoch 19: 3.681215
Loss on epoch 20: 3.727187
Epoch 20 : Accuracy on training set: 0.8232121922626026
Epoch 20 : Accuracy on development set: 0.7645403377110694
Loss on epoch 21: 3.548740
Loss on epoch 22: 3.481202
Loss on epoch 23: 3.460235
Loss on epoch 24: 3.409049
Loss on epoch 25: 3.350418
Loss on epoch 26: 3.444366
Loss on epoch 27: 3.279034
Loss on epoch 28: 3.233062
Loss on epoch 29: 3.158149
Loss on epoch 30: 3.118412
Epoch 30 : Accuracy on training set: 0.8634232121922626
Epoch 30 : Accuracy on development set: 0.7532833020637899
Loss on epoch 31: 3.081005
Loss on epoch 32: 3.045437
Loss on epoch 33: 2.926621
Loss on epoch 34: 2.931365
Loss on epoch 35: 2.843194
Loss on epoch 36: 2.751328
Loss on epoch 37: 2.766973
Loss on epoch 38: 3.081149
Loss on epoch 39: 2.845013
Loss on epoch 40: 2.814105
Epoch 40 : Accuracy on training set: 0.8871043376318875
Epoch 40 : Accuracy on development set: 0.7682926829268293
Loss on epoch 41: 2.694026
Loss on epoch 42: 2.607772
Loss on epoch 43: 2.540644
Loss on epoch 44: 2.479634
Loss on epoch 45: 2.734953
Loss on epoch 46: 2.507866
Loss on epoch 47: 2.340278
Loss on epoch 48: 2.433892
Loss on epoch 49: 2.337925
Accuracy on training set: 0.9131301289566237
Accuracy on develpment set: 0.7523452157598499
Prediction written to file for Rotten Tomatoes dataset.
```

5.4 Use a bidirectional LSTM, concatenate the averages of their states

Evaluate Two LSTMs (bidirectional) + fully connected network, concatenate the averages of their states.

Our accuracy on development data is 77.39%

[ ]:
```
random.seed(1)
np.random.seed(1)
torch.manual_seed(1)

## YOUR CODE HERE
hp = HyperParams(lstm_size = 50, # hidden units in lstm
                 hidden_size = 50, # hidden size of fully-connected layer
                 lstm_layers = 1, # layers in lstm
                 drop_out = 0.5, # dropout rate
                 num_epochs = 50, # number of epochs for SGD-based procedure
                 batch_size = 1024, # examples in a minibatch
                 seq_max_len = 60) # maximum length of an example sequence
use_average = True
bidirectional = True

# Train RNN model.
model4 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
 ↪use_average, bidirectional)

# Generate RNN model predictions for test set.
embeddings_vec = np.array(word_vectors.vectors).astype(float)
test_exs_predicted = eval_model(model4, test_exs, embeddings_vec, hp.
 ↪seq_max_len, pred_only = True)

# Write the test set output
for i, ex in enumerate(test_exs):
    ex.label = int(test_exs_predicted[i])
write_sentiment_examples(test_exs, test_output_path, word_indexer)

print("Prediction written to file for Rotten Tomatoes dataset.")
```

```
Loss on epoch 0: 6.331739
Epoch 0 : Accuracy on training set: 0.5837045720984759
Epoch 0 : Accuracy on development set: 0.5834896810506567
```

```
Loss on epoch 1: 5.841878
Loss on epoch 2: 5.033062
Loss on epoch 3: 4.668347
Loss on epoch 4: 4.512972
Loss on epoch 5: 4.385176
Loss on epoch 6: 4.413477
Loss on epoch 7: 4.308627
Loss on epoch 8: 4.253889
Loss on epoch 9: 4.257465
Loss on epoch 10: 4.199516
Epoch 10 : Accuracy on training set: 0.7808909730363424
Epoch 10 : Accuracy on development set: 0.7420262664165104
Loss on epoch 11: 4.117451
Loss on epoch 12: 4.030150
Loss on epoch 13: 4.066597
Loss on epoch 14: 3.984642
Loss on epoch 15: 4.025867
Loss on epoch 16: 3.963529
Loss on epoch 17: 3.932011
Loss on epoch 18: 3.927685
Loss on epoch 19: 3.879160
Loss on epoch 20: 3.931827
Epoch 20 : Accuracy on training set: 0.8128956623681125
Epoch 20 : Accuracy on development set: 0.7701688555347092
Loss on epoch 21: 3.766663
Loss on epoch 22: 3.720750
Loss on epoch 23: 3.754652
Loss on epoch 24: 3.766137
Loss on epoch 25: 3.812835
Loss on epoch 26: 3.698266
Loss on epoch 27: 3.584020
Loss on epoch 28: 3.678678
Loss on epoch 29: 3.602871
Loss on epoch 30: 3.597236
Epoch 30 : Accuracy on training set: 0.8252051582649472
Epoch 30 : Accuracy on development set: 0.7804878048780488
Loss on epoch 31: 3.597928
Loss on epoch 32: 3.508547
Loss on epoch 33: 3.576733
Loss on epoch 34: 3.542585
Loss on epoch 35: 3.444560
Loss on epoch 36: 3.392123
Loss on epoch 37: 3.388881
Loss on epoch 38: 3.372636
Loss on epoch 39: 3.364726
Loss on epoch 40: 3.339032
Epoch 40 : Accuracy on training set: 0.8467760844079719
Epoch 40 : Accuracy on development set: 0.7879924953095685
```

```
Loss on epoch 41: 3.296636
Loss on epoch 42: 3.493911
Loss on epoch 43: 3.376744
Loss on epoch 44: 3.181008
Loss on epoch 45: 3.323633
Loss on epoch 46: 3.160640
Loss on epoch 47: 3.384176
Loss on epoch 48: 3.221092
Loss on epoch 49: 3.152903
Accuracy on training set: 0.8399765533411488
Accuracy on develpment set: 0.7804878048780488
Prediction written to file for Rotten Tomatoes dataset.
```

5.5 [5111] Average performance and standard deviation

The NN performance can vary depending on the random initialization of its parameters. Train and evaluate each model 10 times, from different random initializations (10 different seeds). Average the accuracy over the 10 runs and compare the performance of the 4 models on the Rotten Tomatoes dataset. Report in your analysis the average and standard deviation for each model.

5.5.1 Model 1 Tests:

- There will be 10 tests here with 10 different seeds, all using model 1.
- Model 1: One LSTM + fully connected network, using the last hidden state of LSTM.

```
[ ]: ## YOUR CODE HERE
seeds = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
accuracies = []

for s in seeds:

  random.seed(s)
  np.random.seed(s)
  torch.manual_seed(s)

  hp = HyperParams(lstm_size = 50, # hidden units in lstm
                   hidden_size = 50, # hidden size of fully-connected layer
                   lstm_layers = 1, # layers in lstm
                   drop_out = 0.5, # dropout rate
                   num_epochs = 50, # number of epochs for SGD-based procedure
                   batch_size = 1024, # examples in a minibatch
                   seq_max_len = 60) # maximum length of an example sequence
  use_average = False
  bidirectional = False

  # Train RNN model.
  model1 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
 ↪use_average, bidirectional)
```

```
  # Evaluate RNN model predictions for test set.
  embeddings_vec = np.array(word_vectors.vectors).astype(float)
  acc = eval_model(model1, test_exs, embeddings_vec, hp.seq_max_len, pred_only␣
↪= False)
  accuracies.append(acc)

avg_acc = np.mean(accuracies)
print('Average accuracy on test set across all 10 evaluations:', avg_acc)
avg_std = np.std(accuracies)
print('Standard deviation:', avg_std)
```

```
Loss on epoch 0: 6.397310
Epoch 0 : Accuracy on training set: 0.5431418522860493
Epoch 0 : Accuracy on development set: 0.5300187617260788
Loss on epoch 1: 6.064509
Loss on epoch 2: 5.493423
Loss on epoch 3: 4.981110
Loss on epoch 4: 4.665425
Loss on epoch 5: 4.585090
Loss on epoch 6: 4.482951
Loss on epoch 7: 4.304576
Loss on epoch 8: 4.165377
Loss on epoch 9: 4.220551
Loss on epoch 10: 4.248222
Epoch 10 : Accuracy on training set: 0.7739742086752638
Epoch 10 : Accuracy on development set: 0.7654784240150094
Loss on epoch 11: 4.077394
Loss on epoch 12: 4.045285
Loss on epoch 13: 3.999397
Loss on epoch 14: 3.952886
Loss on epoch 15: 3.854972
Loss on epoch 16: 3.803097
Loss on epoch 17: 3.905058
Loss on epoch 18: 3.854553
Loss on epoch 19: 3.736169
Loss on epoch 20: 3.858974
Epoch 20 : Accuracy on training set: 0.8077373974208675
Epoch 20 : Accuracy on development set: 0.7729831144465291
Loss on epoch 21: 3.787422
Loss on epoch 22: 3.758644
Loss on epoch 23: 3.731895
Loss on epoch 24: 3.598727
Loss on epoch 25: 3.601509
Loss on epoch 26: 3.638012
Loss on epoch 27: 3.707505
Loss on epoch 28: 3.497229
Loss on epoch 29: 3.501311
Loss on epoch 30: 3.450648
Epoch 30 : Accuracy on training set: 0.8347010550996483
Epoch 30 : Accuracy on development set: 0.7701688555347092
Loss on epoch 31: 3.464190
Loss on epoch 32: 3.448415
Loss on epoch 33: 3.279355
Loss on epoch 34: 3.334016
Loss on epoch 35: 3.301290
Loss on epoch 36: 3.299659
Loss on epoch 37: 3.216300
Loss on epoch 38: 3.162205
Loss on epoch 39: 3.213019
Loss on epoch 40: 3.112714
Epoch 40 : Accuracy on training set: 0.8563892145369285
Epoch 40 : Accuracy on development set: 0.7842401500938087
Loss on epoch 41: 3.130115
Loss on epoch 42: 3.176312
Loss on epoch 43: 3.200650
Loss on epoch 44: 3.093270
Loss on epoch 45: 3.117938
Loss on epoch 46: 3.040054
Loss on epoch 47: 2.966326
Loss on epoch 48: 2.889180
Loss on epoch 49: 3.020883
Accuracy on training set: 0.8718640093786636
Accuracy on develpment set: 0.773921200750469
Loss on epoch 0: 6.267868
Epoch 0 : Accuracy on training set: 0.564947245017585
Epoch 0 : Accuracy on development set: 0.5928705440900562
Loss on epoch 1: 5.932330
Loss on epoch 2: 5.189380
Loss on epoch 3: 4.858705
Loss on epoch 4: 4.556321
Loss on epoch 5: 4.436837
Loss on epoch 6: 4.463091
Loss on epoch 7: 4.226187
Loss on epoch 8: 4.139826
Loss on epoch 9: 4.192905
Loss on epoch 10: 4.163583
Epoch 10 : Accuracy on training set: 0.7973036342321219
Epoch 10 : Accuracy on development set: 0.7523452157598499
Loss on epoch 11: 4.043365
Loss on epoch 12: 4.015802
Loss on epoch 13: 4.092634
Loss on epoch 14: 3.973923
Loss on epoch 15: 3.921847
Loss on epoch 16: 3.873744
Loss on epoch 17: 3.902598
Loss on epoch 18: 3.945920
Loss on epoch 19: 3.763963
Loss on epoch 20: 3.841377
Epoch 20 : Accuracy on training set: 0.8037514654161781
Epoch 20 : Accuracy on development set: 0.7476547842401501
Loss on epoch 21: 3.727669
Loss on epoch 22: 3.748322
Loss on epoch 23: 3.743268
Loss on epoch 24: 3.639222
Loss on epoch 25: 3.643740
Loss on epoch 26: 3.767812
Loss on epoch 27: 3.604565
Loss on epoch 28: 3.601455
Loss on epoch 29: 3.518878
Loss on epoch 30: 3.583085
Epoch 30 : Accuracy on training set: 0.8110199296600235
Epoch 30 : Accuracy on development set: 0.7532833020637899
Loss on epoch 31: 3.653132
Loss on epoch 32: 3.577678
Loss on epoch 33: 3.582230
Loss on epoch 34: 3.576581
Loss on epoch 35: 3.541837
Loss on epoch 36: 3.526462
Loss on epoch 37: 3.462530
Loss on epoch 38: 3.364852
Loss on epoch 39: 3.334983
Loss on epoch 40: 3.312232
Epoch 40 : Accuracy on training set: 0.8413833528722157
Epoch 40 : Accuracy on development set: 0.7448405253283302
Loss on epoch 41: 3.398080
Loss on epoch 42: 3.333921
Loss on epoch 43: 3.371250
Loss on epoch 44: 3.160272
Loss on epoch 45: 3.250384
Loss on epoch 46: 3.114446
Loss on epoch 47: 3.433672
Loss on epoch 48: 3.250335
Loss on epoch 49: 3.189616
Accuracy on training set: 0.8534583821805393
Accuracy on develpment set: 0.7673545966228893
Loss on epoch 0: 6.395678
Epoch 0 : Accuracy on training set: 0.5148886283704572
Epoch 0 : Accuracy on development set: 0.50093808630394
Loss on epoch 1: 6.192025
Loss on epoch 2: 5.691813
Loss on epoch 3: 5.017021
Loss on epoch 4: 4.599822
Loss on epoch 5: 4.431628
Loss on epoch 6: 4.396788
Loss on epoch 7: 4.403067
Loss on epoch 8: 4.258268
Loss on epoch 9: 4.304857
Loss on epoch 10: 4.244149
Epoch 10 : Accuracy on training set: 0.7757327080890973
Epoch 10 : Accuracy on development set: 0.7410881801125704
Loss on epoch 11: 4.174202
Loss on epoch 12: 4.114977
Loss on epoch 13: 4.074571
Loss on epoch 14: 4.031906
Loss on epoch 15: 3.963248
Loss on epoch 16: 4.032993
Loss on epoch 17: 3.864146
Loss on epoch 18: 3.843417
Loss on epoch 19: 3.813267
Loss on epoch 20: 3.801531
Epoch 20 : Accuracy on training set: 0.811957796014068
Epoch 20 : Accuracy on development set: 0.7636022514071295
Loss on epoch 21: 3.741060
Loss on epoch 22: 3.712285
Loss on epoch 23: 3.640734
Loss on epoch 24: 3.602185
Loss on epoch 25: 3.640315
Loss on epoch 26: 3.807474
Loss on epoch 27: 3.576662
Loss on epoch 28: 3.465734
Loss on epoch 29: 3.558282
Loss on epoch 30: 3.520018
Epoch 30 : Accuracy on training set: 0.8201641266119578
Epoch 30 : Accuracy on development set: 0.7617260787992496
Loss on epoch 31: 3.565605
Loss on epoch 32: 3.448687
Loss on epoch 33: 3.430459
Loss on epoch 34: 3.374540
Loss on epoch 35: 3.379696
Loss on epoch 36: 3.286014
Loss on epoch 37: 3.282192
Loss on epoch 38: 3.308931
Loss on epoch 39: 3.271163
Loss on epoch 40: 3.077568
Epoch 40 : Accuracy on training set: 0.855685814771395
Epoch 40 : Accuracy on development set: 0.776735459662289
Loss on epoch 41: 3.258155
Loss on epoch 42: 3.186614
Loss on epoch 43: 3.161302
Loss on epoch 44: 3.073968
Loss on epoch 45: 2.925383
Loss on epoch 46: 3.036043
Loss on epoch 47: 3.035774
Loss on epoch 48: 3.161415
Loss on epoch 49: 2.905850
Accuracy on training set: 0.8713950762016412
Accuracy on develpment set: 0.7579737335834896
Loss on epoch 0: 6.304741
Epoch 0 : Accuracy on training set: 0.5493552168815944
Epoch 0 : Accuracy on development set: 0.5544090056285178
Loss on epoch 1: 5.990075
Loss on epoch 2: 5.303771
Loss on epoch 3: 4.926275
Loss on epoch 4: 4.762162
Loss on epoch 5: 4.507813
Loss on epoch 6: 4.520634
Loss on epoch 7: 4.391179
Loss on epoch 8: 4.255099
Loss on epoch 9: 4.206463
Loss on epoch 10: 4.109718
Epoch 10 : Accuracy on training set: 0.7719812426729191
Epoch 10 : Accuracy on development set: 0.7382739212007504
Loss on epoch 11: 4.215414
Loss on epoch 12: 4.214195
Loss on epoch 13: 4.058351
Loss on epoch 14: 4.078321
Loss on epoch 15: 4.054803
Loss on epoch 16: 3.902553
Loss on epoch 17: 3.908377
Loss on epoch 18: 3.910668
Loss on epoch 19: 3.909931
Loss on epoch 20: 3.818666
Epoch 20 : Accuracy on training set: 0.8106682297772567
Epoch 20 : Accuracy on development set: 0.7570356472795498
Loss on epoch 21: 3.785248
Loss on epoch 22: 3.754106
Loss on epoch 23: 3.823398
Loss on epoch 24: 3.678192
Loss on epoch 25: 3.652429
Loss on epoch 26: 3.663686
Loss on epoch 27: 3.761656
Loss on epoch 28: 3.797965
Loss on epoch 29: 3.622834
Loss on epoch 30: 3.652190
Epoch 30 : Accuracy on training set: 0.8169988276670574
Epoch 30 : Accuracy on development set: 0.7626641651031895
Loss on epoch 31: 3.604678
Loss on epoch 32: 3.494212
Loss on epoch 33: 3.467942
Loss on epoch 34: 3.393777
Loss on epoch 35: 3.365556
Loss on epoch 36: 3.339376
Loss on epoch 37: 3.285629
Loss on epoch 38: 3.493692
Loss on epoch 39: 3.262745
Loss on epoch 40: 3.327220
Epoch 40 : Accuracy on training set: 0.8369284876905041
Epoch 40 : Accuracy on development set: 0.7673545966228893
Loss on epoch 41: 3.248533
Loss on epoch 42: 3.228603
Loss on epoch 43: 3.151531
Loss on epoch 44: 3.249677
Loss on epoch 45: 3.234570
Loss on epoch 46: 3.104342
Loss on epoch 47: 3.088521
Loss on epoch 48: 3.205270
Loss on epoch 49: 2.962363
Accuracy on training set: 0.8674091441969519
Accuracy on develpment set: 0.7701688555347092
Loss on epoch 0: 6.300279
Epoch 0 : Accuracy on training set: 0.5528722157092615
Epoch 0 : Accuracy on development set: 0.5393996247654784
Loss on epoch 1: 5.937208
Loss on epoch 2: 5.372407
Loss on epoch 3: 4.912095
Loss on epoch 4: 4.562885
Loss on epoch 5: 4.475563
Loss on epoch 6: 4.341724
Loss on epoch 7: 4.214330
Loss on epoch 8: 4.189777
Loss on epoch 9: 4.131260
Loss on epoch 10: 4.117317
Epoch 10 : Accuracy on training set: 0.794021101992966
Epoch 10 : Accuracy on development set: 0.7617260787992496
Loss on epoch 11: 4.063996
Loss on epoch 12: 4.071826
Loss on epoch 13: 3.968351
Loss on epoch 14: 3.952926
Loss on epoch 15: 3.984797
Loss on epoch 16: 3.920367
Loss on epoch 17: 3.969038
Loss on epoch 18: 3.972943
Loss on epoch 19: 3.976709
Loss on epoch 20: 3.873947
Epoch 20 : Accuracy on training set: 0.7963657678780773
Epoch 20 : Accuracy on development set: 0.7504690431519699
```

```
Loss on epoch 21: 3.816474
Loss on epoch 22: 3.725000
Loss on epoch 23: 3.735533
Loss on epoch 24: 3.693310
Loss on epoch 25: 3.672524
Loss on epoch 26: 3.634059
Loss on epoch 27: 3.625589
Loss on epoch 28: 3.502872
Loss on epoch 29: 3.495819
Loss on epoch 30: 3.578610
Epoch 30 : Accuracy on training set: 0.8091441969519344
Epoch 30 : Accuracy on development set: 0.7532833020637899
Loss on epoch 31: 3.620534
Loss on epoch 32: 3.475075
Loss on epoch 33: 3.433761
Loss on epoch 34: 3.402654
Loss on epoch 35: 3.364374
Loss on epoch 36: 3.342615
Loss on epoch 37: 3.456892
Loss on epoch 38: 3.328432
Loss on epoch 39: 3.312194
Loss on epoch 40: 3.229784
Epoch 40 : Accuracy on training set: 0.8461899179366941
Epoch 40 : Accuracy on development set: 0.7701688555347092
Loss on epoch 41: 3.180186
Loss on epoch 42: 3.176771
Loss on epoch 43: 3.245642
Loss on epoch 44: 3.064918
Loss on epoch 45: 3.297879
Loss on epoch 46: 3.091605
Loss on epoch 47: 3.029545
Loss on epoch 48: 3.031303
Loss on epoch 49: 3.086655
Accuracy on training set: 0.8490035169988277
Accuracy on develpment set: 0.7701688555347092
Loss on epoch 0: 6.275935
Epoch 0 : Accuracy on training set: 0.5688159437280188
Epoch 0 : Accuracy on development set: 0.5722326454033771
Loss on epoch 1: 5.918039
Loss on epoch 2: 5.366883
Loss on epoch 3: 4.952158
Loss on epoch 4: 4.643641
Loss on epoch 5: 4.514229
Loss on epoch 6: 4.320135
Loss on epoch 7: 4.363358
Loss on epoch 8: 4.228909
Loss on epoch 9: 4.200228
Loss on epoch 10: 4.252742
```

```
Epoch 10 : Accuracy on training set: 0.7867526377491207
Epoch 10 : Accuracy on development set: 0.7673545966228893
Loss on epoch 11: 4.190345
Loss on epoch 12: 4.035844
Loss on epoch 13: 4.017953
Loss on epoch 14: 4.026871
Loss on epoch 15: 3.969001
Loss on epoch 16: 3.919546
Loss on epoch 17: 4.047673
Loss on epoch 18: 3.918150
Loss on epoch 19: 3.877719
Loss on epoch 20: 3.819513
Epoch 20 : Accuracy on training set: 0.809495896834701
Epoch 20 : Accuracy on development set: 0.7682926829268293
Loss on epoch 21: 3.965959
Loss on epoch 22: 3.784651
Loss on epoch 23: 3.759025
Loss on epoch 24: 3.827667
Loss on epoch 25: 3.774070
Loss on epoch 26: 3.761542
Loss on epoch 27: 3.799260
Loss on epoch 28: 3.754666
Loss on epoch 29: 3.602561
Loss on epoch 30: 3.578292
Epoch 30 : Accuracy on training set: 0.8321219226260258
Epoch 30 : Accuracy on development set: 0.7692307692307693
Loss on epoch 31: 3.581038
Loss on epoch 32: 3.569886
Loss on epoch 33: 3.507521
Loss on epoch 34: 3.419951
Loss on epoch 35: 3.447701
Loss on epoch 36: 3.352673
Loss on epoch 37: 3.285342
Loss on epoch 38: 3.346352
Loss on epoch 39: 3.211822
Loss on epoch 40: 3.311207
Epoch 40 : Accuracy on training set: 0.8514654161781946
Epoch 40 : Accuracy on development set: 0.774859287054409
Loss on epoch 41: 3.227826
Loss on epoch 42: 3.158864
Loss on epoch 43: 3.122556
Loss on epoch 44: 3.044036
Loss on epoch 45: 2.978374
Loss on epoch 46: 3.058640
Loss on epoch 47: 2.953350
Loss on epoch 48: 2.920797
Loss on epoch 49: 2.912716
Accuracy on training set: 0.8436107854630716
Accuracy on develpment set: 0.7645403377110694
Loss on epoch 0: 6.200768
Epoch 0 : Accuracy on training set: 0.614419695193435
Epoch 0 : Accuracy on development set: 0.6303939962476548
Loss on epoch 1: 5.673037
Loss on epoch 2: 5.028276
Loss on epoch 3: 4.608032
Loss on epoch 4: 4.576299
Loss on epoch 5: 4.432336
Loss on epoch 6: 4.403045
Loss on epoch 7: 4.304871
Loss on epoch 8: 4.236968
Loss on epoch 9: 4.234956
Loss on epoch 10: 4.269910
Epoch 10 : Accuracy on training set: 0.790504103165299
Epoch 10 : Accuracy on development set: 0.7617260787992496
Loss on epoch 11: 4.262116
Loss on epoch 12: 4.164145
Loss on epoch 13: 4.139437
Loss on epoch 14: 4.104702
Loss on epoch 15: 4.184485
Loss on epoch 16: 4.065075
Loss on epoch 17: 3.910899
Loss on epoch 18: 3.863934
Loss on epoch 19: 3.941867
Loss on epoch 20: 3.860340
Epoch 20 : Accuracy on training set: 0.7978898007033998
Epoch 20 : Accuracy on development set: 0.7682926829268293
Loss on epoch 21: 3.813985
Loss on epoch 22: 3.877155
Loss on epoch 23: 3.771478
Loss on epoch 24: 3.644085
Loss on epoch 25: 3.719486
Loss on epoch 26: 3.680809
Loss on epoch 27: 3.646458
Loss on epoch 28: 3.503067
Loss on epoch 29: 3.515927
Loss on epoch 30: 3.429971
Epoch 30 : Accuracy on training set: 0.836694021101993
Epoch 30 : Accuracy on development set: 0.773921200750469
Loss on epoch 31: 3.621838
Loss on epoch 32: 3.484963
Loss on epoch 33: 3.406063
Loss on epoch 34: 3.496263
Loss on epoch 35: 3.338342
Loss on epoch 36: 3.359093
Loss on epoch 37: 3.305386
Loss on epoch 38: 3.303243
Loss on epoch 39: 3.136923
Loss on epoch 40: 3.414174
Epoch 40 : Accuracy on training set: 0.8406799531066823
Epoch 40 : Accuracy on development set: 0.7786116322701688
Loss on epoch 41: 3.220100
Loss on epoch 42: 3.103242
Loss on epoch 43: 3.104235
Loss on epoch 44: 3.011012
Loss on epoch 45: 3.174007
Loss on epoch 46: 2.979413
Loss on epoch 47: 3.064051
Loss on epoch 48: 2.913937
Loss on epoch 49: 2.807287
Accuracy on training set: 0.8583821805392732
Accuracy on develpment set: 0.774859287054409
Loss on epoch 0: 6.324187
Epoch 0 : Accuracy on training set: 0.5339976553341149
Epoch 0 : Accuracy on development set: 0.5356472795497186
Loss on epoch 1: 6.171686
Loss on epoch 2: 5.692176
Loss on epoch 3: 5.177069
Loss on epoch 4: 4.707744
Loss on epoch 5: 4.540358
Loss on epoch 6: 4.400394
Loss on epoch 7: 4.333890
Loss on epoch 8: 4.288335
Loss on epoch 9: 4.331223
Loss on epoch 10: 4.329996
Epoch 10 : Accuracy on training set: 0.7698710433763188
Epoch 10 : Accuracy on development set: 0.7607879924953096
Loss on epoch 11: 4.252692
Loss on epoch 12: 4.138790
Loss on epoch 13: 4.123224
Loss on epoch 14: 4.061683
Loss on epoch 15: 4.110703
Loss on epoch 16: 4.107508
Loss on epoch 17: 4.051371
Loss on epoch 18: 4.020082
Loss on epoch 19: 3.942309
Loss on epoch 20: 3.978113
Epoch 20 : Accuracy on training set: 0.8072684642438452
Epoch 20 : Accuracy on development set: 0.7664165103189493
Loss on epoch 21: 3.909432
Loss on epoch 22: 3.924498
Loss on epoch 23: 3.883242
Loss on epoch 24: 3.901711
Loss on epoch 25: 3.818079
Loss on epoch 26: 3.838446
Loss on epoch 27: 3.785719
Loss on epoch 28: 3.671713
Loss on epoch 29: 3.754865
Loss on epoch 30: 3.780445
Epoch 30 : Accuracy on training set: 0.8226260257913247
Epoch 30 : Accuracy on development set: 0.7692307692307693
Loss on epoch 31: 3.577147
Loss on epoch 32: 3.588516
Loss on epoch 33: 3.647705
Loss on epoch 34: 3.556658
Loss on epoch 35: 3.556087
Loss on epoch 36: 3.520232
Loss on epoch 37: 3.510361
Loss on epoch 38: 3.412302
Loss on epoch 39: 3.460624
Loss on epoch 40: 3.348735
Epoch 40 : Accuracy on training set: 0.8427901524032825
Epoch 40 : Accuracy on development set: 0.7664165103189493
Loss on epoch 41: 3.312442
Loss on epoch 42: 3.354867
Loss on epoch 43: 3.471507
Loss on epoch 44: 3.364111
Loss on epoch 45: 3.275567
Loss on epoch 46: 3.230300
Loss on epoch 47: 3.285904
Loss on epoch 48: 3.330655
Loss on epoch 49: 3.133458
Accuracy on training set: 0.8539273153575615
Accuracy on develpment set: 0.7636022514071295
Loss on epoch 0: 6.442293
Epoch 0 : Accuracy on training set: 0.5084407971864009
Epoch 0 : Accuracy on development set: 0.5056285178236398
Loss on epoch 1: 6.249653
Loss on epoch 2: 5.981512
Loss on epoch 3: 5.289273
Loss on epoch 4: 5.026931
Loss on epoch 5: 4.645248
Loss on epoch 6: 4.464571
Loss on epoch 7: 4.423369
Loss on epoch 8: 4.342108
Loss on epoch 9: 4.309374
Loss on epoch 10: 4.229925
Epoch 10 : Accuracy on training set: 0.7839390386869871
Epoch 10 : Accuracy on development set: 0.7664165103189493
Loss on epoch 11: 4.149419
Loss on epoch 12: 4.090944
Loss on epoch 13: 4.060359
Loss on epoch 14: 4.068450
Loss on epoch 15: 3.997352
Loss on epoch 16: 4.009990
Loss on epoch 17: 4.039382
Loss on epoch 18: 4.003794
Loss on epoch 19: 3.953954
Loss on epoch 20: 4.017795
Epoch 20 : Accuracy on training set: 0.7953106682297773
Epoch 20 : Accuracy on development set: 0.775797373358349
Loss on epoch 21: 3.961559
Loss on epoch 22: 3.987726
Loss on epoch 23: 3.833696
Loss on epoch 24: 3.774066
Loss on epoch 25: 3.733001
Loss on epoch 26: 3.755417
Loss on epoch 27: 3.698781
Loss on epoch 28: 3.615242
Loss on epoch 29: 3.612598
Loss on epoch 30: 3.630592
Epoch 30 : Accuracy on training set: 0.8328253223915592
Epoch 30 : Accuracy on development set: 0.7692307692307693
Loss on epoch 31: 3.547271
Loss on epoch 32: 3.585500
Loss on epoch 33: 3.616289
Loss on epoch 34: 3.469570
Loss on epoch 35: 3.427986
Loss on epoch 36: 3.428806
Loss on epoch 37: 3.385198
Loss on epoch 38: 3.336885
Loss on epoch 39: 3.436185
Loss on epoch 40: 3.320051
Epoch 40 : Accuracy on training set: 0.848651817116061
Epoch 40 : Accuracy on development set: 0.7673545966228893
Loss on epoch 41: 3.253802
Loss on epoch 42: 3.221512
Loss on epoch 43: 3.233616
Loss on epoch 44: 3.197581
Loss on epoch 45: 3.194651
Loss on epoch 46: 3.010566
Loss on epoch 47: 3.047014
Loss on epoch 48: 3.237294
Loss on epoch 49: 3.204305
Accuracy on training set: 0.8352872215709262
Accuracy on develpment set: 0.7551594746716698
Loss on epoch 0: 6.224954
Epoch 0 : Accuracy on training set: 0.5684642438452521
Epoch 0 : Accuracy on development set: 0.5816135084427767
Loss on epoch 1: 5.820318
Loss on epoch 2: 5.199941
Loss on epoch 3: 4.795958
Loss on epoch 4: 4.604599
Loss on epoch 5: 4.469532
Loss on epoch 6: 4.377968
Loss on epoch 7: 4.252651
Loss on epoch 8: 4.288217
Loss on epoch 9: 4.115288
Loss on epoch 10: 4.082069
Epoch 10 : Accuracy on training set: 0.7903868698710433
Epoch 10 : Accuracy on development set: 0.7439024390243902
Loss on epoch 11: 4.137662
Loss on epoch 12: 3.952554
Loss on epoch 13: 3.922643
Loss on epoch 14: 3.851834
Loss on epoch 15: 3.971657
Loss on epoch 16: 3.817327
Loss on epoch 17: 3.795277
Loss on epoch 18: 3.758717
Loss on epoch 19: 3.721204
Loss on epoch 20: 3.816894
Epoch 20 : Accuracy on training set: 0.7797186400937867
Epoch 20 : Accuracy on development set: 0.7298311444652908
Loss on epoch 21: 3.759292
Loss on epoch 22: 3.751216
Loss on epoch 23: 3.599880
Loss on epoch 24: 3.575762
Loss on epoch 25: 3.517394
Loss on epoch 26: 3.541911
Loss on epoch 27: 3.409451
Loss on epoch 28: 3.572228
Loss on epoch 29: 3.662103
Loss on epoch 30: 3.503873
Epoch 30 : Accuracy on training set: 0.8438452520515827
Epoch 30 : Accuracy on development set: 0.7804878048780488
Loss on epoch 31: 3.452297
Loss on epoch 32: 3.410908
Loss on epoch 33: 3.323812
Loss on epoch 34: 3.368244
Loss on epoch 35: 3.530497
Loss on epoch 36: 3.308987
Loss on epoch 37: 3.324793
Loss on epoch 38: 3.402704
Loss on epoch 39: 3.321598
Loss on epoch 40: 3.378665
Epoch 40 : Accuracy on training set: 0.8246189917936694
Epoch 40 : Accuracy on development set: 0.7570356472795498
Loss on epoch 41: 3.441686
Loss on epoch 42: 3.365516
Loss on epoch 43: 3.207621
Loss on epoch 44: 3.085615
Loss on epoch 45: 3.019816
Loss on epoch 46: 3.008427
Loss on epoch 47: 3.010566
Loss on epoch 48: 2.946026
Loss on epoch 49: 2.942687
Accuracy on training set: 0.8762016412661195
Accuracy on develpment set: 0.7645403377110694
Average accuracy on test set across all 10 evaluations: 0.8383677298311445
Standard deviation: 0.03190886463258
```

5.5.2 Model 2 Tests:

- There will be 10 tests here with 10 different seeds, all using model 2.
- Model 2: One LSTM + fully connected network, using average of all states of the LSTM.

[ ]:
```
seeds = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
accuracies = []

for s in seeds:

  random.seed(s)
  np.random.seed(s)
  torch.manual_seed(s)

  hp = HyperParams(lstm_size = 50, # hidden units in lstm
                   hidden_size = 50, # hidden size of fully-connected layer
                   lstm_layers = 1, # layers in lstm
                   drop_out = 0.5, # dropout rate
                   num_epochs = 50, # number of epochs for SGD-based procedure
                   batch_size = 1024, # examples in a minibatch
                   seq_max_len = 60) # maximum length of an example sequence
  use_average = True
  bidirectional = False

  # Train RNN model.
  model2 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
↪use_average, bidirectional)

  # Generate RNN model predictions for test set.
  embeddings_vec = np.array(word_vectors.vectors).astype(float)
  acc = eval_model(model2, test_exs, embeddings_vec, hp.seq_max_len, pred_only␣
↪= False)
  accuracies.append(acc)

avg_acc = np.mean(accuracies)
print('Average accuracy on test set across all 10 evaluations:', avg_acc)
```

```
avg_std = np.std(accuracies)
print('Standard deviation:', avg_std)
```

```
Loss on epoch 0: 6.255435
Epoch 0 : Accuracy on training set: 0.5974208675263775
Epoch 0 : Accuracy on development set: 0.6041275797373359
Loss on epoch 1: 5.727144
Loss on epoch 2: 5.044729
Loss on epoch 3: 4.693672
Loss on epoch 4: 4.589602
Loss on epoch 5: 4.472541
Loss on epoch 6: 4.399734
Loss on epoch 7: 4.294918
Loss on epoch 8: 4.230246
Loss on epoch 9: 4.284705
Loss on epoch 10: 4.209890
Epoch 10 : Accuracy on training set: 0.774443141852286
Epoch 10 : Accuracy on development set: 0.7476547842401501
Loss on epoch 11: 4.273160
Loss on epoch 12: 4.256356
Loss on epoch 13: 4.169132
Loss on epoch 14: 4.110255
Loss on epoch 15: 4.189195
Loss on epoch 16: 4.166646
Loss on epoch 17: 4.074082
Loss on epoch 18: 4.042634
Loss on epoch 19: 3.987878
Loss on epoch 20: 4.126448
Epoch 20 : Accuracy on training set: 0.7950762016412661
Epoch 20 : Accuracy on development set: 0.7673545966228893
Loss on epoch 21: 4.070434
Loss on epoch 22: 3.949808
Loss on epoch 23: 3.983102
Loss on epoch 24: 3.962345
Loss on epoch 25: 3.941555
Loss on epoch 26: 3.910303
Loss on epoch 27: 3.850723
Loss on epoch 28: 3.851024
Loss on epoch 29: 3.778185
Loss on epoch 30: 3.776885
Epoch 30 : Accuracy on training set: 0.8132473622508792
Epoch 30 : Accuracy on development set: 0.7795497185741088
Loss on epoch 31: 3.732068
Loss on epoch 32: 3.731229
Loss on epoch 33: 3.688191
Loss on epoch 34: 3.694585
Loss on epoch 35: 3.623162
Loss on epoch 36: 3.604882
Loss on epoch 37: 3.542093
Loss on epoch 38: 3.493712
Loss on epoch 39: 3.587601
Loss on epoch 40: 3.464287
Epoch 40 : Accuracy on training set: 0.823915592028136
Epoch 40 : Accuracy on development set: 0.7701688555347092
Loss on epoch 41: 3.474175
Loss on epoch 42: 3.452711
Loss on epoch 43: 3.355658
Loss on epoch 44: 3.406165
Loss on epoch 45: 3.388207
Loss on epoch 46: 3.445300
Loss on epoch 47: 3.402315
Loss on epoch 48: 3.271828
Loss on epoch 49: 3.196667
Accuracy on training set: 0.7937866354044548
Accuracy on develpment set: 0.7166979362101313
Loss on epoch 0: 6.152875
Epoch 0 : Accuracy on training set: 0.6399765533411489
Epoch 0 : Accuracy on development set: 0.6350844277673546
Loss on epoch 1: 5.475465
Loss on epoch 2: 4.820598
Loss on epoch 3: 4.588094
Loss on epoch 4: 4.473848
Loss on epoch 5: 4.388067
Loss on epoch 6: 4.411091
Loss on epoch 7: 4.286098
Loss on epoch 8: 4.246010
Loss on epoch 9: 4.291475
Loss on epoch 10: 4.287062
Epoch 10 : Accuracy on training set: 0.7715123094958969
Epoch 10 : Accuracy on development set: 0.7476547842401501
Loss on epoch 11: 4.184418
Loss on epoch 12: 4.124940
Loss on epoch 13: 4.144271
Loss on epoch 14: 4.111087
Loss on epoch 15: 4.112401
Loss on epoch 16: 4.099515
Loss on epoch 17: 4.111658
Loss on epoch 18: 4.092200
Loss on epoch 19: 4.030658
Loss on epoch 20: 4.000238
Epoch 20 : Accuracy on training set: 0.7977725674091442
Epoch 20 : Accuracy on development set: 0.7682926829268293
Loss on epoch 21: 3.920431
Loss on epoch 22: 3.952678
Loss on epoch 23: 3.945651
Loss on epoch 24: 3.921254
Loss on epoch 25: 3.892431
Loss on epoch 26: 3.838856
Loss on epoch 27: 3.757259
Loss on epoch 28: 3.781899
Loss on epoch 29: 3.708489
Loss on epoch 30: 3.717315
Epoch 30 : Accuracy on training set: 0.8118405627198124
Epoch 30 : Accuracy on development set: 0.7701688555347092
Loss on epoch 31: 3.752245
Loss on epoch 32: 3.658879
Loss on epoch 33: 3.659513
Loss on epoch 34: 3.595937
Loss on epoch 35: 3.599232
Loss on epoch 36: 3.637761
Loss on epoch 37: 3.594100
Loss on epoch 38: 3.505361
Loss on epoch 39: 3.550112
Loss on epoch 40: 3.530501
Epoch 40 : Accuracy on training set: 0.8314185228604923
Epoch 40 : Accuracy on development set: 0.7579737335834896
Loss on epoch 41: 3.421081
Loss on epoch 42: 3.538181
Loss on epoch 43: 3.510110
Loss on epoch 44: 3.404528
Loss on epoch 45: 3.552001
Loss on epoch 46: 3.312294
Loss on epoch 47: 3.403614
Loss on epoch 48: 3.450060
Loss on epoch 49: 3.341656
Accuracy on training set: 0.8242672919109026
Accuracy on develpment set: 0.7673545966228893
Loss on epoch 0: 6.218799
Epoch 0 : Accuracy on training set: 0.6161781946072684
Epoch 0 : Accuracy on development set: 0.6060037523452158
Loss on epoch 1: 5.573567
Loss on epoch 2: 4.846902
Loss on epoch 3: 4.549551
Loss on epoch 4: 4.451869
Loss on epoch 5: 4.410693
Loss on epoch 6: 4.343249
Loss on epoch 7: 4.379559
Loss on epoch 8: 4.277066
Loss on epoch 9: 4.265149
Loss on epoch 10: 4.230400
Epoch 10 : Accuracy on training set: 0.7662368112543962
Epoch 10 : Accuracy on development set: 0.7514071294559099
Loss on epoch 11: 4.225793
Loss on epoch 12: 4.218227
Loss on epoch 13: 4.265088
Loss on epoch 14: 4.120057
Loss on epoch 15: 4.075146
Loss on epoch 16: 4.123872
Loss on epoch 17: 4.048387
Loss on epoch 18: 4.054363
Loss on epoch 19: 4.114965
Loss on epoch 20: 4.039901
Epoch 20 : Accuracy on training set: 0.7954279015240329
Epoch 20 : Accuracy on development set: 0.7579737335834896
Loss on epoch 21: 3.970984
Loss on epoch 22: 3.967073
Loss on epoch 23: 3.965778
Loss on epoch 24: 3.917228
Loss on epoch 25: 3.928777
Loss on epoch 26: 3.855791
Loss on epoch 27: 3.841838
Loss on epoch 28: 3.786291
Loss on epoch 29: 3.748387
Loss on epoch 30: 3.690995
Epoch 30 : Accuracy on training set: 0.7542790152403283
Epoch 30 : Accuracy on development set: 0.7195121951219512
Loss on epoch 31: 3.900245
Loss on epoch 32: 3.688638
Loss on epoch 33: 3.666167
Loss on epoch 34: 3.640515
Loss on epoch 35: 3.640836
Loss on epoch 36: 3.607602
Loss on epoch 37: 3.599078
Loss on epoch 38: 3.649667
Loss on epoch 39: 3.515576
Loss on epoch 40: 3.593829
Epoch 40 : Accuracy on training set: 0.8273153575615475
Epoch 40 : Accuracy on development set: 0.7626641651031895
Loss on epoch 41: 3.555743
Loss on epoch 42: 3.486689
Loss on epoch 43: 3.498145
Loss on epoch 44: 3.426606
Loss on epoch 45: 3.384708
Loss on epoch 46: 3.340016
Loss on epoch 47: 3.436632
Loss on epoch 48: 3.481150
Loss on epoch 49: 3.440256
Accuracy on training set: 0.8281359906213365
Accuracy on develpment set: 0.776735459662289
Loss on epoch 0: 6.210113
Epoch 0 : Accuracy on training set: 0.6157092614302462
Epoch 0 : Accuracy on development set: 0.6210131332082551
Loss on epoch 1: 5.586168
Loss on epoch 2: 4.973027
Loss on epoch 3: 4.621007
Loss on epoch 4: 4.549584
Loss on epoch 5: 4.385182
Loss on epoch 6: 4.431973
Loss on epoch 7: 4.308853
Loss on epoch 8: 4.310535
Loss on epoch 9: 4.278942
Loss on epoch 10: 4.229826
Epoch 10 : Accuracy on training set: 0.7815943728018757
Epoch 10 : Accuracy on development set: 0.7626641651031895
Loss on epoch 11: 4.174849
Loss on epoch 12: 4.214932
Loss on epoch 13: 4.177891
Loss on epoch 14: 4.159770
Loss on epoch 15: 4.213426
Loss on epoch 16: 4.093503
Loss on epoch 17: 4.090383
Loss on epoch 18: 4.062463
Loss on epoch 19: 4.042010
Loss on epoch 20: 4.003931
Epoch 20 : Accuracy on training set: 0.7930832356389215
Epoch 20 : Accuracy on development set: 0.7607879924953096
Loss on epoch 21: 3.991662
Loss on epoch 22: 3.934039
Loss on epoch 23: 3.989770
Loss on epoch 24: 3.950533
Loss on epoch 25: 3.864767
Loss on epoch 26: 3.841696
Loss on epoch 27: 3.958396
Loss on epoch 28: 3.933707
Loss on epoch 29: 3.810684
Loss on epoch 30: 3.906807
Epoch 30 : Accuracy on training set: 0.8032825322391559
Epoch 30 : Accuracy on development set: 0.7579737335834896
Loss on epoch 31: 3.732107
Loss on epoch 32: 3.738102
Loss on epoch 33: 3.716831
Loss on epoch 34: 3.641844
Loss on epoch 35: 3.638404
Loss on epoch 36: 3.593852
Loss on epoch 37: 3.601578
Loss on epoch 38: 3.472710
Loss on epoch 39: 3.535443
Loss on epoch 40: 3.541111
Epoch 40 : Accuracy on training set: 0.835873388042204
Epoch 40 : Accuracy on development set: 0.775797373358349
Loss on epoch 41: 3.429376
Loss on epoch 42: 3.408395
Loss on epoch 43: 3.388535
Loss on epoch 44: 3.403486
Loss on epoch 45: 3.303782
Loss on epoch 46: 3.321065
Loss on epoch 47: 3.312498
Loss on epoch 48: 3.310372
Loss on epoch 49: 3.302201
Accuracy on training set: 0.849706916764361
Accuracy on develpment set: 0.775797373358349
Loss on epoch 0: 6.244310
Epoch 0 : Accuracy on training set: 0.6117233294255568
Epoch 0 : Accuracy on development set: 0.6078799249530957
Loss on epoch 1: 5.568825
Loss on epoch 2: 4.877355
Loss on epoch 3: 4.529917
Loss on epoch 4: 4.402203
Loss on epoch 5: 4.521102
Loss on epoch 6: 4.407759
Loss on epoch 7: 4.307225
Loss on epoch 8: 4.276848
Loss on epoch 9: 4.218545
Loss on epoch 10: 4.384427
Epoch 10 : Accuracy on training set: 0.7732708089097303
Epoch 10 : Accuracy on development set: 0.7495309568480301
Loss on epoch 11: 4.182628
Loss on epoch 12: 4.246563
Loss on epoch 13: 4.222338
Loss on epoch 14: 4.083561
Loss on epoch 15: 4.138008
Loss on epoch 16: 4.057136
Loss on epoch 17: 4.064775
Loss on epoch 18: 4.040193
Loss on epoch 19: 4.003931
Loss on epoch 20: 3.994611
Epoch 20 : Accuracy on training set: 0.7886283704572098
Epoch 20 : Accuracy on development set: 0.7645403377110694
Loss on epoch 21: 3.946128
Loss on epoch 22: 3.938604
Loss on epoch 23: 3.951765
Loss on epoch 24: 3.955328
Loss on epoch 25: 3.900236
Loss on epoch 26: 3.899631
Loss on epoch 27: 3.864934
Loss on epoch 28: 3.804878
Loss on epoch 29: 3.778750
Loss on epoch 30: 3.767152
Epoch 30 : Accuracy on training set: 0.7953106682297773
Epoch 30 : Accuracy on development set: 0.7448405253283302
Loss on epoch 31: 3.850986
Loss on epoch 32: 3.789402
Loss on epoch 33: 3.729885
Loss on epoch 34: 3.694679
Loss on epoch 35: 3.775602
Loss on epoch 36: 3.729817
Loss on epoch 37: 3.682341
Loss on epoch 38: 3.773499
Loss on epoch 39: 3.580800
Loss on epoch 40: 3.593062
Epoch 40 : Accuracy on training set: 0.8267291910902697
Epoch 40 : Accuracy on development set: 0.7626641651031895
Loss on epoch 41: 3.641622
Loss on epoch 42: 3.547961
Loss on epoch 43: 3.551861
Loss on epoch 44: 3.696684
Loss on epoch 45: 3.642309
Loss on epoch 46: 3.519461
Loss on epoch 47: 3.519678
Loss on epoch 48: 3.467992
Loss on epoch 49: 3.423459
Accuracy on training set: 0.8352872215709262
Accuracy on develpment set: 0.774859287054409
Loss on epoch 0: 6.172214
Epoch 0 : Accuracy on training set: 0.6409144196951935
Epoch 0 : Accuracy on development set: 0.650093808630394
Loss on epoch 1: 5.497291
Loss on epoch 2: 4.945138
Loss on epoch 3: 4.710582
Loss on epoch 4: 4.513721
Loss on epoch 5: 4.415395
Loss on epoch 6: 4.340926
Loss on epoch 7: 4.404160
Loss on epoch 8: 4.318070
Loss on epoch 9: 4.280412
Loss on epoch 10: 4.329997
Epoch 10 : Accuracy on training set: 0.7732708089097303
Epoch 10 : Accuracy on development set: 0.7607879924953096
Loss on epoch 11: 4.255350
Loss on epoch 12: 4.293982
Loss on epoch 13: 4.152934
Loss on epoch 14: 4.128675
Loss on epoch 15: 4.215342
Loss on epoch 16: 4.129439
Loss on epoch 17: 4.108799
Loss on epoch 18: 4.121060
Loss on epoch 19: 4.053116
Loss on epoch 20: 4.033873
Epoch 20 : Accuracy on training set: 0.7947245017584994
Epoch 20 : Accuracy on development set: 0.7560975609756098
Loss on epoch 21: 4.095206
Loss on epoch 22: 4.009405
Loss on epoch 23: 3.979660
Loss on epoch 24: 3.893221
Loss on epoch 25: 3.933008
Loss on epoch 26: 3.918423
Loss on epoch 27: 3.895157
Loss on epoch 28: 3.894225
Loss on epoch 29: 3.822487
Loss on epoch 30: 3.783883
Epoch 30 : Accuracy on training set: 0.8107854630715123
Epoch 30 : Accuracy on development set: 0.7720450281425891
Loss on epoch 31: 3.784150
Loss on epoch 32: 3.787360
Loss on epoch 33: 3.790071
Loss on epoch 34: 3.745789
Loss on epoch 35: 3.710001
Loss on epoch 36: 3.601646
Loss on epoch 37: 3.617981
Loss on epoch 38: 3.606791
Loss on epoch 39: 3.590824
Loss on epoch 40: 3.566158
Epoch 40 : Accuracy on training set: 0.82989449003517
Epoch 40 : Accuracy on development set: 0.7720450281425891
Loss on epoch 41: 3.506164
Loss on epoch 42: 3.559134
Loss on epoch 43: 3.527285
Loss on epoch 44: 3.481621
Loss on epoch 45: 3.407176
Loss on epoch 46: 3.493508
Loss on epoch 47: 3.466229
Loss on epoch 48: 3.522461
Loss on epoch 49: 3.387815
Accuracy on training set: 0.8382180539273154
Accuracy on develpment set: 0.7607879924953096
Loss on epoch 0: 6.067056
Epoch 0 : Accuracy on training set: 0.6698710433763189
Epoch 0 : Accuracy on development set: 0.6969981238273921
Loss on epoch 1: 5.241830
Loss on epoch 2: 4.746315
Loss on epoch 3: 4.592268
Loss on epoch 4: 4.356359
Loss on epoch 5: 4.394641
Loss on epoch 6: 4.486432
Loss on epoch 7: 4.331808
Loss on epoch 8: 4.313028
Loss on epoch 9: 4.320816
Loss on epoch 10: 4.223444
Epoch 10 : Accuracy on training set: 0.7803048065650645
Epoch 10 : Accuracy on development set: 0.7607879924953096
Loss on epoch 11: 4.199911
Loss on epoch 12: 4.229369
Loss on epoch 13: 4.281164
Loss on epoch 14: 4.227860
Loss on epoch 15: 4.250382
Loss on epoch 16: 4.140456
Loss on epoch 17: 4.090689
Loss on epoch 18: 4.074325
Loss on epoch 19: 4.101192
Loss on epoch 20: 4.122393
Epoch 20 : Accuracy on training set: 0.7908558030480657
Epoch 20 : Accuracy on development set: 0.7673545966228893
Loss on epoch 21: 4.014748
Loss on epoch 22: 4.016671
Loss on epoch 23: 3.999885
Loss on epoch 24: 3.958031
Loss on epoch 25: 3.998850
Loss on epoch 26: 3.915070
Loss on epoch 27: 3.920503
Loss on epoch 28: 3.811136
Loss on epoch 29: 3.859857
Loss on epoch 30: 3.793482
Epoch 30 : Accuracy on training set: 0.8112543962485346
Epoch 30 : Accuracy on development set: 0.774859287054409
Loss on epoch 31: 3.808733
Loss on epoch 32: 3.810563
Loss on epoch 33: 3.782189
Loss on epoch 34: 3.772382
Loss on epoch 35: 3.767009
Loss on epoch 36: 3.679413
Loss on epoch 37: 3.679751
Loss on epoch 38: 3.642316
Loss on epoch 39: 3.631695
Loss on epoch 40: 3.648042
Epoch 40 : Accuracy on training set: 0.8104337631887456
Epoch 40 : Accuracy on development set: 0.7673545966228893
Loss on epoch 41: 3.626388
Loss on epoch 42: 3.535617
Loss on epoch 43: 3.661841
Loss on epoch 44: 3.493348
Loss on epoch 45: 3.440019
Loss on epoch 46: 3.494378
Loss on epoch 47: 3.455557
Loss on epoch 48: 3.378190
Loss on epoch 49: 3.354270
Accuracy on training set: 0.8150058616647128
Accuracy on develpment set: 0.7504690431519699
Loss on epoch 0: 6.198216
Epoch 0 : Accuracy on training set: 0.6023446658851114
Epoch 0 : Accuracy on development set: 0.5891181988742964
Loss on epoch 1: 5.707810
Loss on epoch 2: 4.954961
Loss on epoch 3: 4.574498
Loss on epoch 4: 4.465042
Loss on epoch 5: 4.432275
Loss on epoch 6: 4.309081
Loss on epoch 7: 4.291248
Loss on epoch 8: 4.279750
Loss on epoch 9: 4.233270
Loss on epoch 10: 4.295443
Epoch 10 : Accuracy on training set: 0.7787807737397421
Epoch 10 : Accuracy on development set: 0.7532833020637899
Loss on epoch 11: 4.229213
Loss on epoch 12: 4.182776
Loss on epoch 13: 4.234574
Loss on epoch 14: 4.156996
Loss on epoch 15: 4.131103
Loss on epoch 16: 4.064485
Loss on epoch 17: 4.082368
Loss on epoch 18: 4.035299
Loss on epoch 19: 3.956428
Loss on epoch 20: 3.945368
Epoch 20 : Accuracy on training set: 0.794021101992966
Epoch 20 : Accuracy on development set: 0.7626641651031895
Loss on epoch 21: 3.933885
Loss on epoch 22: 3.904466
Loss on epoch 23: 3.928635
Loss on epoch 24: 3.947633
Loss on epoch 25: 3.806093
Loss on epoch 26: 3.830980
Loss on epoch 27: 3.832487
Loss on epoch 28: 3.720126
Loss on epoch 29: 3.832490
Loss on epoch 30: 3.765270
Epoch 30 : Accuracy on training set: 0.8124267291910903
Epoch 30 : Accuracy on development set: 0.776735459662289
Loss on epoch 31: 3.702581
Loss on epoch 32: 3.721930
Loss on epoch 33: 3.752652
Loss on epoch 34: 3.641225
Loss on epoch 35: 3.615060
Loss on epoch 36: 3.571743
Loss on epoch 37: 3.487999
Loss on epoch 38: 3.560154
Loss on epoch 39: 3.554635
Loss on epoch 40: 3.503434
Epoch 40 : Accuracy on training set: 0.8322391559202814
Epoch 40 : Accuracy on development set: 0.7729831144465291
Loss on epoch 41: 3.452919
Loss on epoch 42: 3.454313
Loss on epoch 43: 3.550239
Loss on epoch 44: 3.478525
Loss on epoch 45: 3.376242
Loss on epoch 46: 3.341417
Loss on epoch 47: 3.421864
Loss on epoch 48: 3.476609
Loss on epoch 49: 3.427675
Accuracy on training set: 0.8437280187573271
Accuracy on develpment set: 0.7720450281425891
Loss on epoch 0: 6.289442
Epoch 0 : Accuracy on training set: 0.5516998827667058
Epoch 0 : Accuracy on development set: 0.5628517823639775
Loss on epoch 1: 5.890073
Loss on epoch 2: 5.203193
Loss on epoch 3: 4.771000
Loss on epoch 4: 4.485179
Loss on epoch 5: 4.450983
Loss on epoch 6: 4.328522
Loss on epoch 7: 4.332055
Loss on epoch 8: 4.295664
Loss on epoch 9: 4.301430
Loss on epoch 10: 4.252616
Epoch 10 : Accuracy on training set: 0.776905041031653
Epoch 10 : Accuracy on development set: 0.7570356472795498
Loss on epoch 11: 4.207652
Loss on epoch 12: 4.133107
Loss on epoch 13: 4.185940
Loss on epoch 14: 4.114770
Loss on epoch 15: 4.084034
Loss on epoch 16: 4.006382
Loss on epoch 17: 4.101987
Loss on epoch 18: 4.025200
Loss on epoch 19: 3.999928
Loss on epoch 20: 4.040768
Epoch 20 : Accuracy on training set: 0.7886283704572098
Epoch 20 : Accuracy on development set: 0.7514071294559099
Loss on epoch 21: 3.979470
Loss on epoch 22: 3.990639
Loss on epoch 23: 3.962342
Loss on epoch 24: 3.883971
Loss on epoch 25: 3.831182
Loss on epoch 26: 3.751747
Loss on epoch 27: 3.891199
Loss on epoch 28: 3.784416
Loss on epoch 29: 3.731199
Loss on epoch 30: 3.748028
Epoch 30 : Accuracy on training set: 0.8234466588511137
Epoch 30 : Accuracy on development set: 0.7842401500938087
Loss on epoch 31: 3.730973
Loss on epoch 32: 3.684174
Loss on epoch 33: 3.734871
Loss on epoch 34: 3.659078
Loss on epoch 35: 3.637622
Loss on epoch 36: 3.628427
Loss on epoch 37: 3.533036
Loss on epoch 38: 3.597344
Loss on epoch 39: 3.662075
Loss on epoch 40: 3.566120
Epoch 40 : Accuracy on training set: 0.827432590855803
Epoch 40 : Accuracy on development set: 0.7701688555347092
Loss on epoch 41: 3.488456
Loss on epoch 42: 3.562257
Loss on epoch 43: 3.571288
Loss on epoch 44: 3.537329
Loss on epoch 45: 3.452630
Loss on epoch 46: 3.366431
Loss on epoch 47: 3.471961
Loss on epoch 48: 3.461369
Loss on epoch 49: 3.565722
Accuracy on training set: 0.8324736225087925
Accuracy on develpment set: 0.774859287054409
Loss on epoch 0: 6.064033
Epoch 0 : Accuracy on training set: 0.6396248534583822
Epoch 0 : Accuracy on development set: 0.6388367729831145
Loss on epoch 1: 5.308822
Loss on epoch 2: 4.754976
Loss on epoch 3: 4.535662
Loss on epoch 4: 4.446309
Loss on epoch 5: 4.373778
Loss on epoch 6: 4.370652
Loss on epoch 7: 4.309690
Loss on epoch 8: 4.325196
Loss on epoch 9: 4.213505
Loss on epoch 10: 4.237902
Epoch 10 : Accuracy on training set: 0.773622508792497
Epoch 10 : Accuracy on development set: 0.7476547842401501
Loss on epoch 11: 4.227855
Loss on epoch 12: 4.163211
Loss on epoch 13: 4.143941
Loss on epoch 14: 4.266842
Loss on epoch 15: 4.135105
Loss on epoch 16: 4.122495
Loss on epoch 17: 4.053971
Loss on epoch 18: 4.007378
Loss on epoch 19: 4.034956
Loss on epoch 20: 4.032854
Epoch 20 : Accuracy on training set: 0.7996483001172333
Epoch 20 : Accuracy on development set: 0.7720450281425891
Loss on epoch 21: 3.922055
Loss on epoch 22: 3.876254
Loss on epoch 23: 3.851929
Loss on epoch 24: 3.823017
Loss on epoch 25: 3.773855
Loss on epoch 26: 3.762634
Loss on epoch 27: 3.772541
Loss on epoch 28: 3.736093
Loss on epoch 29: 3.713714
Loss on epoch 30: 3.652594
Epoch 30 : Accuracy on training set: 0.8171160609613131
Epoch 30 : Accuracy on development set: 0.7645403377110694
Loss on epoch 31: 3.692573
Loss on epoch 32: 3.581265
Loss on epoch 33: 3.561885
Loss on epoch 34: 3.569354
Loss on epoch 35: 3.604766
Loss on epoch 36: 3.480120
Loss on epoch 37: 3.435906
Loss on epoch 38: 3.411772
Loss on epoch 39: 3.428152
Loss on epoch 40: 3.367844
Epoch 40 : Accuracy on training set: 0.8423212192262602
Epoch 40 : Accuracy on development set: 0.7814258911819888
Loss on epoch 41: 3.356106
Loss on epoch 42: 3.297525
Loss on epoch 43: 3.412756
Loss on epoch 44: 3.288312
Loss on epoch 45: 3.337833
Loss on epoch 46: 3.325853
Loss on epoch 47: 3.237942
Loss on epoch 48: 3.274468
Loss on epoch 49: 3.339104
Accuracy on training set: 0.8382180539273154
Accuracy on develpment set: 0.773921200750469
```

```
Average accuracy on test set across all 10 evaluations: 0.8516885553470919
Standard deviation: 0.045898119314445075
```

5.5.3 Model 3 Tests:

- There will be 10 tests here with 10 different seeds, all using model 3.
- Model 3: Two LSTMs (bidirectional) + fully connected network, concatenating their last states.

```
[ ]: seeds = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
     accuracies = []

     for s in seeds:

       random.seed(s)
       np.random.seed(s)
       torch.manual_seed(s)

       hp = HyperParams(lstm_size = 50, # hidden units in lstm
                        hidden_size = 50, # hidden size of fully-connected layer
                        lstm_layers = 1, # layers in lstm
                        drop_out = 0.5, # dropout rate
                        num_epochs = 50, # number of epochs for SGD-based procedure
                        batch_size = 1024, # examples in a minibatch
                        seq_max_len = 60) # maximum length of an example sequence
       use_average = False
       bidirectional = True

       # Train RNN model.
       model3 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

       # Generate RNN model predictions for test set.
       embeddings_vec = np.array(word_vectors.vectors).astype(float)
       acc = eval_model(model3, test_exs, embeddings_vec, hp.seq_max_len, pred_only␣
      ↪= False)
       accuracies.append(acc)

     avg_acc = np.mean(accuracies)
     print('Average accuracy on test set across all 10 evaluations:', avg_acc)
     avg_std = np.std(accuracies)
     print('Standard deviation:', avg_std)
```

```
Loss on epoch 0: 6.300638
Epoch 0 : Accuracy on training set: 0.5862837045720984
Epoch 0 : Accuracy on development set: 0.5722326454033771
Loss on epoch 1: 5.911338
Loss on epoch 2: 5.041899
Loss on epoch 3: 4.580555
Loss on epoch 4: 4.407141
Loss on epoch 5: 4.251714
Loss on epoch 6: 4.384693
Loss on epoch 7: 4.181728
Loss on epoch 8: 4.041171
Loss on epoch 9: 3.988287
Loss on epoch 10: 4.059657
Epoch 10 : Accuracy on training set: 0.8019929660023447
Epoch 10 : Accuracy on development set: 0.7570356472795498
Loss on epoch 11: 4.084089
Loss on epoch 12: 4.039108
Loss on epoch 13: 3.963974
Loss on epoch 14: 3.838778
Loss on epoch 15: 3.813588
Loss on epoch 16: 3.705499
Loss on epoch 17: 3.741072
Loss on epoch 18: 3.705702
Loss on epoch 19: 3.587873
Loss on epoch 20: 3.586627
Epoch 20 : Accuracy on training set: 0.8186400937866354
Epoch 20 : Accuracy on development set: 0.7673545966228893
Loss on epoch 21: 3.580734
Loss on epoch 22: 3.422700
Loss on epoch 23: 3.431054
Loss on epoch 24: 3.480932
Loss on epoch 25: 3.276970
Loss on epoch 26: 3.255050
Loss on epoch 27: 3.369251
Loss on epoch 28: 3.465249
Loss on epoch 29: 3.228243
Loss on epoch 30: 3.164480
Epoch 30 : Accuracy on training set: 0.8580304806565064
Epoch 30 : Accuracy on development set: 0.7645403377110694
Loss on epoch 31: 3.137098
Loss on epoch 32: 3.052317
Loss on epoch 33: 2.929636
Loss on epoch 34: 3.004447
Loss on epoch 35: 2.931948
Loss on epoch 36: 2.766763
Loss on epoch 37: 2.867903
Loss on epoch 38: 2.810944
Loss on epoch 39: 2.725347
Loss on epoch 40: 2.687881
Epoch 40 : Accuracy on training set: 0.8903868698710434
Epoch 40 : Accuracy on development set: 0.7673545966228893
Loss on epoch 41: 2.623007
Loss on epoch 42: 2.504663
Loss on epoch 43: 2.517618
Loss on epoch 44: 2.443218
Loss on epoch 45: 2.455173
Loss on epoch 46: 2.399876
Loss on epoch 47: 2.343814
Loss on epoch 48: 2.150585
Loss on epoch 49: 2.211618
Accuracy on training set: 0.8494724501758499
Accuracy on develpment set: 0.724202626641651
Loss on epoch 0: 6.476173
Epoch 0 : Accuracy on training set: 0.5365767878077374
Epoch 0 : Accuracy on development set: 0.5534709193245778
Loss on epoch 1: 6.101449
Loss on epoch 2: 5.608163
Loss on epoch 3: 4.853618
Loss on epoch 4: 4.592319
Loss on epoch 5: 4.393368
Loss on epoch 6: 4.312670
Loss on epoch 7: 4.217034
Loss on epoch 8: 4.286347
Loss on epoch 9: 4.229264
Loss on epoch 10: 4.145923
Epoch 10 : Accuracy on training set: 0.7951934349355216
Epoch 10 : Accuracy on development set: 0.775797373358349
Loss on epoch 11: 4.003318
Loss on epoch 12: 4.017098
Loss on epoch 13: 4.011863
Loss on epoch 14: 3.869946
Loss on epoch 15: 3.941032
Loss on epoch 16: 3.825827
Loss on epoch 17: 3.808317
Loss on epoch 18: 3.756933
Loss on epoch 19: 3.681215
Loss on epoch 20: 3.727187
Epoch 20 : Accuracy on training set: 0.8232121922626026
Epoch 20 : Accuracy on development set: 0.7645403377110694
Loss on epoch 21: 3.548740
Loss on epoch 22: 3.481202
Loss on epoch 23: 3.460235
Loss on epoch 24: 3.409049
Loss on epoch 25: 3.350418
Loss on epoch 26: 3.444366
Loss on epoch 27: 3.279034
Loss on epoch 28: 3.233062
Loss on epoch 29: 3.158149
Loss on epoch 30: 3.118412
Epoch 30 : Accuracy on training set: 0.8634232121922626
Epoch 30 : Accuracy on development set: 0.7532833020637899
```

```
Loss on epoch 31: 3.081005
Loss on epoch 32: 3.045437
Loss on epoch 33: 2.926621
Loss on epoch 34: 2.931365
Loss on epoch 35: 2.843194
Loss on epoch 36: 2.751328
Loss on epoch 37: 2.766973
Loss on epoch 38: 3.081149
Loss on epoch 39: 2.845013
Loss on epoch 40: 2.814105
Epoch 40 : Accuracy on training set: 0.8871043376318875
Epoch 40 : Accuracy on development set: 0.7682926829268293
Loss on epoch 41: 2.694026
Loss on epoch 42: 2.607772
Loss on epoch 43: 2.540644
Loss on epoch 44: 2.479634
Loss on epoch 45: 2.734953
Loss on epoch 46: 2.507866
Loss on epoch 47: 2.340278
Loss on epoch 48: 2.433892
Loss on epoch 49: 2.337925
Accuracy on training set: 0.9131301289566237
Accuracy on develpment set: 0.7523452157598499
Loss on epoch 0: 6.386632
Epoch 0 : Accuracy on training set: 0.5305978898007034
Epoch 0 : Accuracy on development set: 0.524390243902439
Loss on epoch 1: 6.057286
Loss on epoch 2: 5.556502
Loss on epoch 3: 5.096416
Loss on epoch 4: 4.646888
Loss on epoch 5: 4.385208
Loss on epoch 6: 4.252962
Loss on epoch 7: 4.296443
Loss on epoch 8: 4.121337
Loss on epoch 9: 4.009259
Loss on epoch 10: 4.020686
Epoch 10 : Accuracy on training set: 0.8018757327080891
Epoch 10 : Accuracy on development set: 0.7711069418386491
Loss on epoch 11: 3.968162
Loss on epoch 12: 3.983465
Loss on epoch 13: 3.864589
Loss on epoch 14: 4.000046
Loss on epoch 15: 3.749303
Loss on epoch 16: 3.820089
Loss on epoch 17: 3.737995
Loss on epoch 18: 3.655791
Loss on epoch 19: 3.633760
Loss on epoch 20: 3.584719
```

```
Epoch 20 : Accuracy on training set: 0.832590855803048
Epoch 20 : Accuracy on development set: 0.7711069418386491
Loss on epoch 21: 3.524306
Loss on epoch 22: 3.487823
Loss on epoch 23: 3.392997
Loss on epoch 24: 3.337689
Loss on epoch 25: 3.400679
Loss on epoch 26: 3.318948
Loss on epoch 27: 3.190341
Loss on epoch 28: 3.421739
Loss on epoch 29: 3.168851
Loss on epoch 30: 3.043844
Epoch 30 : Accuracy on training set: 0.845369284876905
Epoch 30 : Accuracy on development set: 0.7579737335834896
Loss on epoch 31: 3.059685
Loss on epoch 32: 2.999488
Loss on epoch 33: 3.007057
Loss on epoch 34: 2.917996
Loss on epoch 35: 2.937737
Loss on epoch 36: 2.766854
Loss on epoch 37: 2.648748
Loss on epoch 38: 2.991819
Loss on epoch 39: 2.635004
Loss on epoch 40: 2.775266
Epoch 40 : Accuracy on training set: 0.8451348182883939
Epoch 40 : Accuracy on development set: 0.7439024390243902
Loss on epoch 41: 2.757684
Loss on epoch 42: 2.699647
Loss on epoch 43: 2.565097
Loss on epoch 44: 2.389892
Loss on epoch 45: 2.374954
Loss on epoch 46: 2.347880
Loss on epoch 47: 2.492696
Loss on epoch 48: 2.463128
Loss on epoch 49: 2.313983
Accuracy on training set: 0.9064478311840563
Accuracy on develpment set: 0.7598499061913696
Loss on epoch 0: 6.218525
Epoch 0 : Accuracy on training set: 0.6276670574443142
Epoch 0 : Accuracy on development set: 0.6088180112570356
Loss on epoch 1: 5.474661
Loss on epoch 2: 4.963369
Loss on epoch 3: 4.534640
Loss on epoch 4: 4.492376
Loss on epoch 5: 4.294633
Loss on epoch 6: 4.134574
Loss on epoch 7: 4.131169
Loss on epoch 8: 4.025733
Loss on epoch 9: 3.936495
Loss on epoch 10: 3.957772
Epoch 10 : Accuracy on training set: 0.7838218053927315
Epoch 10 : Accuracy on development set: 0.7298311444652908
Loss on epoch 11: 3.912060
Loss on epoch 12: 3.842647
Loss on epoch 13: 3.848004
Loss on epoch 14: 3.800605
Loss on epoch 15: 3.763241
Loss on epoch 16: 3.610835
Loss on epoch 17: 3.584584
Loss on epoch 18: 3.591026
Loss on epoch 19: 3.602980
Loss on epoch 20: 3.556328
Epoch 20 : Accuracy on training set: 0.8323563892145369
Epoch 20 : Accuracy on development set: 0.7682926829268293
Loss on epoch 21: 3.415027
Loss on epoch 22: 3.356055
Loss on epoch 23: 3.498700
Loss on epoch 24: 3.290444
Loss on epoch 25: 3.230192
Loss on epoch 26: 3.228653
Loss on epoch 27: 3.145599
Loss on epoch 28: 3.178984
Loss on epoch 29: 3.027031
Loss on epoch 30: 3.149593
Epoch 30 : Accuracy on training set: 0.8658851113716295
Epoch 30 : Accuracy on development set: 0.7570356472795498
Loss on epoch 31: 3.045141
Loss on epoch 32: 2.990529
Loss on epoch 33: 2.914165
Loss on epoch 34: 2.854967
Loss on epoch 35: 2.749843
Loss on epoch 36: 2.751062
Loss on epoch 37: 2.854511
Loss on epoch 38: 2.601726
Loss on epoch 39: 2.497746
Loss on epoch 40: 2.538450
Epoch 40 : Accuracy on training set: 0.8953106682297772
Epoch 40 : Accuracy on development set: 0.7654784240150094
Loss on epoch 41: 2.570700
Loss on epoch 42: 2.541780
Loss on epoch 43: 2.432111
Loss on epoch 44: 2.396972
Loss on epoch 45: 2.291384
Loss on epoch 46: 2.380248
Loss on epoch 47: 2.266707
Loss on epoch 48: 2.114344
Loss on epoch 49: 2.115088
Accuracy on training set: 0.9126611957796014
Accuracy on develpment set: 0.7626641651031895
Loss on epoch 0: 6.263822
Epoch 0 : Accuracy on training set: 0.590504103165299
Epoch 0 : Accuracy on development set: 0.6088180112570356
Loss on epoch 1: 5.613663
Loss on epoch 2: 4.740492
Loss on epoch 3: 4.388274
Loss on epoch 4: 4.316328
Loss on epoch 5: 4.307429
Loss on epoch 6: 4.155961
Loss on epoch 7: 4.054559
Loss on epoch 8: 4.049514
Loss on epoch 9: 4.163625
Loss on epoch 10: 4.154304
Epoch 10 : Accuracy on training set: 0.8015240328253224
Epoch 10 : Accuracy on development set: 0.7598499061913696
Loss on epoch 11: 3.908084
Loss on epoch 12: 3.939470
Loss on epoch 13: 3.878564
Loss on epoch 14: 3.875630
Loss on epoch 15: 3.754428
Loss on epoch 16: 3.694004
Loss on epoch 17: 3.722009
Loss on epoch 18: 3.627308
Loss on epoch 19: 3.579645
Loss on epoch 20: 3.515019
Epoch 20 : Accuracy on training set: 0.8359906213364595
Epoch 20 : Accuracy on development set: 0.7579737335834896
Loss on epoch 21: 3.600562
Loss on epoch 22: 3.527734
Loss on epoch 23: 3.444338
Loss on epoch 24: 3.460394
Loss on epoch 25: 3.303937
Loss on epoch 26: 3.310645
Loss on epoch 27: 3.201286
Loss on epoch 28: 3.274973
Loss on epoch 29: 3.155078
Loss on epoch 30: 3.102153
Epoch 30 : Accuracy on training set: 0.8661195779601407
Epoch 30 : Accuracy on development set: 0.7617260787992496
Loss on epoch 31: 2.992333
Loss on epoch 32: 3.004630
Loss on epoch 33: 3.044327
Loss on epoch 34: 2.934057
Loss on epoch 35: 3.059412
Loss on epoch 36: 2.842159
Loss on epoch 37: 2.745340
Loss on epoch 38: 2.728608
Loss on epoch 39: 2.716347
Loss on epoch 40: 2.616997
Epoch 40 : Accuracy on training set: 0.8839390386869871
Epoch 40 : Accuracy on development set: 0.7626641651031895
Loss on epoch 41: 2.728354
Loss on epoch 42: 2.635037
Loss on epoch 43: 2.470191
Loss on epoch 44: 2.446432
Loss on epoch 45: 2.578379
Loss on epoch 46: 2.414738
Loss on epoch 47: 2.253408
Loss on epoch 48: 2.326597
Loss on epoch 49: 2.106042
Accuracy on training set: 0.8910902696365768
Accuracy on develpment set: 0.7485928705440901
Loss on epoch 0: 6.110412
Epoch 0 : Accuracy on training set: 0.6440797186400938
Epoch 0 : Accuracy on development set: 0.6407129455909943
Loss on epoch 1: 5.400068
Loss on epoch 2: 4.920683
Loss on epoch 3: 4.596538
Loss on epoch 4: 4.392591
Loss on epoch 5: 4.251840
Loss on epoch 6: 4.172095
Loss on epoch 7: 4.187581
Loss on epoch 8: 4.097312
Loss on epoch 9: 3.983752
Loss on epoch 10: 4.014762
Epoch 10 : Accuracy on training set: 0.8045720984759672
Epoch 10 : Accuracy on development set: 0.7654784240150094
Loss on epoch 11: 4.024738
Loss on epoch 12: 3.846636
Loss on epoch 13: 3.727893
Loss on epoch 14: 3.779161
Loss on epoch 15: 3.811105
Loss on epoch 16: 3.665133
Loss on epoch 17: 3.601709
Loss on epoch 18: 3.654638
Loss on epoch 19: 3.532082
Loss on epoch 20: 3.528154
Epoch 20 : Accuracy on training set: 0.8342321219226261
Epoch 20 : Accuracy on development set: 0.7579737335834896
Loss on epoch 21: 3.476367
Loss on epoch 22: 3.452957
Loss on epoch 23: 3.404655
Loss on epoch 24: 3.220510
Loss on epoch 25: 3.222903
Loss on epoch 26: 3.180127
Loss on epoch 27: 3.230608
Loss on epoch 28: 3.180232
Loss on epoch 29: 3.011119
Loss on epoch 30: 3.000946
Epoch 30 : Accuracy on training set: 0.8499413833528722
Epoch 30 : Accuracy on development set: 0.7729831144465291
Loss on epoch 31: 3.020296
Loss on epoch 32: 2.952338
Loss on epoch 33: 2.879533
Loss on epoch 34: 2.766042
Loss on epoch 35: 2.788216
Loss on epoch 36: 2.614908
Loss on epoch 37: 2.670353
Loss on epoch 38: 2.707315
Loss on epoch 39: 2.506682
Loss on epoch 40: 2.437752
Epoch 40 : Accuracy on training set: 0.8989449003516999
Epoch 40 : Accuracy on development set: 0.7570356472795498
Loss on epoch 41: 2.453788
Loss on epoch 42: 2.391105
Loss on epoch 43: 2.228729
Loss on epoch 44: 2.192369
Loss on epoch 45: 2.381876
Loss on epoch 46: 2.325723
Loss on epoch 47: 2.246162
Loss on epoch 48: 2.156467
Loss on epoch 49: 2.063453
Accuracy on training set: 0.8977725674091442
Accuracy on develpment set: 0.7410881801125704
Loss on epoch 0: 6.097217
Epoch 0 : Accuracy on training set: 0.6492379835873388
Epoch 0 : Accuracy on development set: 0.6538461538461539
Loss on epoch 1: 5.325948
Loss on epoch 2: 4.594061
Loss on epoch 3: 4.406936
Loss on epoch 4: 4.202239
Loss on epoch 5: 4.263628
Loss on epoch 6: 4.253698
Loss on epoch 7: 4.106931
Loss on epoch 8: 4.070292
Loss on epoch 9: 3.966068
Loss on epoch 10: 3.928054
Epoch 10 : Accuracy on training set: 0.8007033997655334
Epoch 10 : Accuracy on development set: 0.7495309568480301
Loss on epoch 11: 4.012674
Loss on epoch 12: 3.893397
Loss on epoch 13: 3.791475
Loss on epoch 14: 3.784196
Loss on epoch 15: 3.811808
Loss on epoch 16: 3.664960
Loss on epoch 17: 3.554224
Loss on epoch 18: 3.615435
Loss on epoch 19: 3.536273
Loss on epoch 20: 3.515534
Epoch 20 : Accuracy on training set: 0.824736225087925
Epoch 20 : Accuracy on development set: 0.7664165103189493
Loss on epoch 21: 3.485724
Loss on epoch 22: 3.604000
Loss on epoch 23: 3.467391
Loss on epoch 24: 3.325102
Loss on epoch 25: 3.270456
Loss on epoch 26: 3.144633
Loss on epoch 27: 3.412879
Loss on epoch 28: 3.254566
Loss on epoch 29: 3.192883
Loss on epoch 30: 2.983846
Epoch 30 : Accuracy on training set: 0.8634232121922626
Epoch 30 : Accuracy on development set: 0.7729831144465291
Loss on epoch 31: 3.017678
Loss on epoch 32: 3.150104
Loss on epoch 33: 3.034754
Loss on epoch 34: 3.007554
Loss on epoch 35: 2.906815
Loss on epoch 36: 2.859430
Loss on epoch 37: 2.822934
Loss on epoch 38: 2.867206
Loss on epoch 39: 2.720915
Loss on epoch 40: 2.551859
Epoch 40 : Accuracy on training set: 0.8569753810082064
Epoch 40 : Accuracy on development set: 0.7307692307692307
Loss on epoch 41: 2.652281
Loss on epoch 42: 2.544193
Loss on epoch 43: 2.603355
Loss on epoch 44: 2.537791
Loss on epoch 45: 2.465037
Loss on epoch 46: 2.464102
Loss on epoch 47: 2.313752
Loss on epoch 48: 2.307062
Loss on epoch 49: 2.190157
Accuracy on training set: 0.9164126611957796
Accuracy on develpment set: 0.7485928705440901
Loss on epoch 0: 6.316004
Epoch 0 : Accuracy on training set: 0.5939038686987105
Epoch 0 : Accuracy on development set: 0.5666041275797373
```

```
Loss on epoch 1: 5.863657
Loss on epoch 2: 5.017568
Loss on epoch 3: 4.545403
Loss on epoch 4: 4.401071
Loss on epoch 5: 4.316947
Loss on epoch 6: 4.178704
Loss on epoch 7: 4.098491
Loss on epoch 8: 4.057248
Loss on epoch 9: 4.150223
Loss on epoch 10: 4.098138
Epoch 10 : Accuracy on training set: 0.785580304806565
Epoch 10 : Accuracy on development set: 0.7636022514071295
Loss on epoch 11: 4.009860
Loss on epoch 12: 3.895650
Loss on epoch 13: 3.875972
Loss on epoch 14: 3.757470
Loss on epoch 15: 3.732984
Loss on epoch 16: 3.691829
Loss on epoch 17: 3.645853
Loss on epoch 18: 3.657995
Loss on epoch 19: 3.495750
Loss on epoch 20: 3.477151
Epoch 20 : Accuracy on training set: 0.8134818288393904
Epoch 20 : Accuracy on development set: 0.7673545966228893
Loss on epoch 21: 3.512752
Loss on epoch 22: 3.449920
Loss on epoch 23: 3.397629
Loss on epoch 24: 3.306304
Loss on epoch 25: 3.249116
Loss on epoch 26: 3.188039
Loss on epoch 27: 3.118275
Loss on epoch 28: 3.021971
Loss on epoch 29: 3.118841
Loss on epoch 30: 2.966795
Epoch 30 : Accuracy on training set: 0.8718640093786636
Epoch 30 : Accuracy on development set: 0.774859287054409
Loss on epoch 31: 2.980131
Loss on epoch 32: 2.851966
Loss on epoch 33: 2.924003
Loss on epoch 34: 2.828818
Loss on epoch 35: 2.886652
Loss on epoch 36: 2.668593
Loss on epoch 37: 2.587332
Loss on epoch 38: 2.693323
Loss on epoch 39: 2.564129
Loss on epoch 40: 2.460624
Epoch 40 : Accuracy on training set: 0.8708089097303634
Epoch 40 : Accuracy on development set: 0.7701688555347092
```

```
Loss on epoch 41: 2.552896
Loss on epoch 42: 2.351044
Loss on epoch 43: 2.472612
Loss on epoch 44: 2.371296
Loss on epoch 45: 2.228473
Loss on epoch 46: 2.231430
Loss on epoch 47: 2.131627
Loss on epoch 48: 2.053564
Loss on epoch 49: 2.241172
Accuracy on training set: 0.9072684642438452
Accuracy on develpment set: 0.7363977485928705
Loss on epoch 0: 6.177367
Epoch 0 : Accuracy on training set: 0.635052754982415
Epoch 0 : Accuracy on development set: 0.6350844277673546
Loss on epoch 1: 5.389721
Loss on epoch 2: 4.847429
Loss on epoch 3: 4.501842
Loss on epoch 4: 4.308172
Loss on epoch 5: 4.239984
Loss on epoch 6: 4.171288
Loss on epoch 7: 4.123831
Loss on epoch 8: 4.026692
Loss on epoch 9: 3.977050
Loss on epoch 10: 3.988073
Epoch 10 : Accuracy on training set: 0.799179366940211
Epoch 10 : Accuracy on development set: 0.7645403377110694
Loss on epoch 11: 3.964995
Loss on epoch 12: 3.826549
Loss on epoch 13: 3.851452
Loss on epoch 14: 3.774999
Loss on epoch 15: 3.648689
Loss on epoch 16: 3.728903
Loss on epoch 17: 3.678722
Loss on epoch 18: 3.668061
Loss on epoch 19: 3.569914
Loss on epoch 20: 3.523002
Epoch 20 : Accuracy on training set: 0.8284876905041032
Epoch 20 : Accuracy on development set: 0.7439024390243902
Loss on epoch 21: 3.507190
Loss on epoch 22: 3.586367
Loss on epoch 23: 3.544229
Loss on epoch 24: 3.437200
Loss on epoch 25: 3.311629
Loss on epoch 26: 3.216375
Loss on epoch 27: 3.414503
Loss on epoch 28: 3.269961
Loss on epoch 29: 3.168693
Loss on epoch 30: 3.244632
```

```
Epoch 30 : Accuracy on training set: 0.8623681125439625
Epoch 30 : Accuracy on development set: 0.7523452157598499
Loss on epoch 31: 3.131260
Loss on epoch 32: 3.089821
Loss on epoch 33: 2.948546
Loss on epoch 34: 2.901821
Loss on epoch 35: 2.890262
Loss on epoch 36: 2.703646
Loss on epoch 37: 2.657961
Loss on epoch 38: 2.715953
Loss on epoch 39: 2.809125
Loss on epoch 40: 2.728253
Epoch 40 : Accuracy on training set: 0.8787807737397421
Epoch 40 : Accuracy on development set: 0.7326454033771107
Loss on epoch 41: 2.517874
Loss on epoch 42: 2.424022
Loss on epoch 43: 2.518949
Loss on epoch 44: 2.489282
Loss on epoch 45: 2.297215
Loss on epoch 46: 2.320327
Loss on epoch 47: 2.336944
Loss on epoch 48: 2.244741
Loss on epoch 49: 2.178064
Accuracy on training set: 0.8974208675263775
Accuracy on develpment set: 0.7589118198874296
Loss on epoch 0: 6.472960
Epoch 0 : Accuracy on training set: 0.54021101992966
Epoch 0 : Accuracy on development set: 0.5553470919324578
Loss on epoch 1: 6.076941
Loss on epoch 2: 5.457554
Loss on epoch 3: 4.720755
Loss on epoch 4: 4.492757
Loss on epoch 5: 4.408018
Loss on epoch 6: 4.235322
Loss on epoch 7: 4.151920
Loss on epoch 8: 4.373504
Loss on epoch 9: 4.138864
Loss on epoch 10: 4.057203
Epoch 10 : Accuracy on training set: 0.7973036342321219
Epoch 10 : Accuracy on development set: 0.7617260787992496
Loss on epoch 11: 4.003553
Loss on epoch 12: 3.837350
Loss on epoch 13: 3.872110
Loss on epoch 14: 3.843992
Loss on epoch 15: 3.761249
Loss on epoch 16: 3.719005
Loss on epoch 17: 3.673764
Loss on epoch 18: 3.649893
Loss on epoch 19: 3.604081
Loss on epoch 20: 3.652135
Epoch 20 : Accuracy on training set: 0.82989449003517
Epoch 20 : Accuracy on development set: 0.7504690431519699
Loss on epoch 21: 3.516937
Loss on epoch 22: 3.489163
Loss on epoch 23: 3.442560
Loss on epoch 24: 3.333412
Loss on epoch 25: 3.405786
Loss on epoch 26: 3.424907
Loss on epoch 27: 3.309357
Loss on epoch 28: 3.262243
Loss on epoch 29: 3.227737
Loss on epoch 30: 3.166924
Epoch 30 : Accuracy on training set: 0.8383352872215709
Epoch 30 : Accuracy on development set: 0.7420262664165104
Loss on epoch 31: 3.173124
Loss on epoch 32: 3.057608
Loss on epoch 33: 3.075955
Loss on epoch 34: 2.984180
Loss on epoch 35: 3.184951
Loss on epoch 36: 3.029595
Loss on epoch 37: 2.893968
Loss on epoch 38: 2.964529
Loss on epoch 39: 2.757266
Loss on epoch 40: 2.720292
Epoch 40 : Accuracy on training set: 0.876905041031653
Epoch 40 : Accuracy on development set: 0.7485928705440901
Loss on epoch 41: 2.769813
Loss on epoch 42: 2.780497
Loss on epoch 43: 2.685184
Loss on epoch 44: 2.619868
Loss on epoch 45: 2.559174
Loss on epoch 46: 2.561932
Loss on epoch 47: 2.361891
Loss on epoch 48: 2.304237
Loss on epoch 49: 2.273037
Accuracy on training set: 0.8818288393903869
Accuracy on develpment set: 0.7589118198874296
Average accuracy on test set across all 10 evaluations: 0.8113508442776736
Standard deviation: 0.03791876662335993
```

5.5.4 Model 4 Tests:

- There will be 10 tests here with 10 different seeds, all using model 4.
- Model 4: Two LSTMs (bidirectional) + fully connected network, concatenating the averages of their states.

```
[ ]: seeds = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
     accuracies = []

     for s in seeds:

       random.seed(s)
       np.random.seed(s)
       torch.manual_seed(s)

       hp = HyperParams(lstm_size = 50, # hidden units in lstm
                        hidden_size = 50, # hidden size of fully-connected layer
                        lstm_layers = 1, # layers in lstm
                        drop_out = 0.5, # dropout rate
                        num_epochs = 50, # number of epochs for SGD-based procedure
                        batch_size = 1024, # examples in a minibatch
                        seq_max_len = 60) # maximum length of an example sequence
       use_average = True
       bidirectional = True

       # Train RNN model.
       model4 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

       # Generate RNN model predictions for test set.
       embeddings_vec = np.array(word_vectors.vectors).astype(float)
       acc = eval_model(model4, test_exs, embeddings_vec, hp.seq_max_len, pred_only␣
      ↪= False)

       accuracies.append(acc)

     avg_acc = np.mean(accuracies)
     print('Average accuracy on test set across all 10 evaluations:', avg_acc)
     avg_std = np.std(accuracies)
     print('Standard deviation:', avg_std)
```

```
Loss on epoch 0: 6.234572
Epoch 0 : Accuracy on training set: 0.6252051582649473
Epoch 0 : Accuracy on development set: 0.6369606003752345
Loss on epoch 1: 5.705043
Loss on epoch 2: 4.976670
Loss on epoch 3: 4.625502
Loss on epoch 4: 4.508529
Loss on epoch 5: 4.343558
Loss on epoch 6: 4.453292
Loss on epoch 7: 4.254796
Loss on epoch 8: 4.128486
Loss on epoch 9: 4.127715
Loss on epoch 10: 4.148503
Epoch 10 : Accuracy on training set: 0.7962485345838218
Epoch 10 : Accuracy on development set: 0.7598499061913696
Loss on epoch 11: 4.067836
Loss on epoch 12: 4.063529
Loss on epoch 13: 4.073436
Loss on epoch 14: 4.003458
Loss on epoch 15: 4.005667
Loss on epoch 16: 3.986571
Loss on epoch 17: 3.939664
Loss on epoch 18: 3.878471
Loss on epoch 19: 3.837375
Loss on epoch 20: 3.847113
Epoch 20 : Accuracy on training set: 0.7981242672919109
Epoch 20 : Accuracy on development set: 0.7645403377110694
Loss on epoch 21: 3.849042
Loss on epoch 22: 3.739303
Loss on epoch 23: 3.764688
Loss on epoch 24: 3.756403
Loss on epoch 25: 3.666021
Loss on epoch 26: 3.606484
Loss on epoch 27: 3.601793
Loss on epoch 28: 3.588073
Loss on epoch 29: 3.578010
Loss on epoch 30: 3.525254
Epoch 30 : Accuracy on training set: 0.8211019929660024
Epoch 30 : Accuracy on development set: 0.7711069418386491
Loss on epoch 31: 3.543552
Loss on epoch 32: 3.496245
Loss on epoch 33: 3.389245
Loss on epoch 34: 3.408916
Loss on epoch 35: 3.416168
Loss on epoch 36: 3.320776
Loss on epoch 37: 3.365682
Loss on epoch 38: 3.314095
Loss on epoch 39: 3.313909
Loss on epoch 40: 3.267124
Epoch 40 : Accuracy on training set: 0.8477139507620164
Epoch 40 : Accuracy on development set: 0.7823639774859287
Loss on epoch 41: 3.284508
Loss on epoch 42: 3.282822
Loss on epoch 43: 3.158080
Loss on epoch 44: 3.242368
Loss on epoch 45: 3.292938
Loss on epoch 46: 3.191373
Loss on epoch 47: 3.092793
Loss on epoch 48: 3.088878
Loss on epoch 49: 3.295131
```

```
Accuracy on training set: 0.860609613130129
Accuracy on develpment set: 0.7908067542213884
Loss on epoch 0: 6.331739
Epoch 0 : Accuracy on training set: 0.5837045720984759
Epoch 0 : Accuracy on development set: 0.5834896810506567
Loss on epoch 1: 5.841878
Loss on epoch 2: 5.033062
Loss on epoch 3: 4.668347
Loss on epoch 4: 4.512972
Loss on epoch 5: 4.385176
Loss on epoch 6: 4.413477
Loss on epoch 7: 4.308627
Loss on epoch 8: 4.253889
Loss on epoch 9: 4.257465
Loss on epoch 10: 4.199516
Epoch 10 : Accuracy on training set: 0.7808909730363424
Epoch 10 : Accuracy on development set: 0.7420262664165104
Loss on epoch 11: 4.117451
Loss on epoch 12: 4.030150
Loss on epoch 13: 4.066597
Loss on epoch 14: 3.984642
Loss on epoch 15: 4.025867
Loss on epoch 16: 3.963529
Loss on epoch 17: 3.932011
Loss on epoch 18: 3.927685
Loss on epoch 19: 3.879160
Loss on epoch 20: 3.931827
Epoch 20 : Accuracy on training set: 0.8128956623681125
Epoch 20 : Accuracy on development set: 0.7701688555347092
Loss on epoch 21: 3.766663
Loss on epoch 22: 3.720750
Loss on epoch 23: 3.754652
Loss on epoch 24: 3.766137
Loss on epoch 25: 3.812835
Loss on epoch 26: 3.698266
Loss on epoch 27: 3.584020
Loss on epoch 28: 3.678678
Loss on epoch 29: 3.602871
Loss on epoch 30: 3.597236
Epoch 30 : Accuracy on training set: 0.8252051582649472
Epoch 30 : Accuracy on development set: 0.7804878048780488
Loss on epoch 31: 3.597928
Loss on epoch 32: 3.508547
Loss on epoch 33: 3.576733
Loss on epoch 34: 3.542585
Loss on epoch 35: 3.444560
Loss on epoch 36: 3.392123
Loss on epoch 37: 3.388881
Loss on epoch 38: 3.372636
Loss on epoch 39: 3.364726
Loss on epoch 40: 3.339032
Epoch 40 : Accuracy on training set: 0.8467760844079719
Epoch 40 : Accuracy on development set: 0.7879924953095685
Loss on epoch 41: 3.296636
Loss on epoch 42: 3.493911
Loss on epoch 43: 3.376744
Loss on epoch 44: 3.181008
Loss on epoch 45: 3.323633
Loss on epoch 46: 3.160640
Loss on epoch 47: 3.384176
Loss on epoch 48: 3.221092
Loss on epoch 49: 3.152903
Accuracy on training set: 0.8399765533411488
Accuracy on develpment set: 0.7804878048780488
Loss on epoch 0: 6.313270
Epoch 0 : Accuracy on training set: 0.5640093786635404
Epoch 0 : Accuracy on development set: 0.5647279549718575
Loss on epoch 1: 5.939829
Loss on epoch 2: 5.258513
Loss on epoch 3: 4.737614
Loss on epoch 4: 4.485034
Loss on epoch 5: 4.411281
Loss on epoch 6: 4.348779
Loss on epoch 7: 4.432864
Loss on epoch 8: 4.302586
Loss on epoch 9: 4.250588
Loss on epoch 10: 4.214500
Epoch 10 : Accuracy on training set: 0.7719812426729191
Epoch 10 : Accuracy on development set: 0.7363977485928705
Loss on epoch 11: 4.166607
Loss on epoch 12: 4.107593
Loss on epoch 13: 4.189721
Loss on epoch 14: 4.041376
Loss on epoch 15: 3.979395
Loss on epoch 16: 4.030958
Loss on epoch 17: 3.904432
Loss on epoch 18: 3.900747
Loss on epoch 19: 3.968254
Loss on epoch 20: 3.917381
Epoch 20 : Accuracy on training set: 0.8010550996483001
Epoch 20 : Accuracy on development set: 0.7532833020637899
Loss on epoch 21: 3.833262
Loss on epoch 22: 3.800579
Loss on epoch 23: 3.775752
Loss on epoch 24: 3.677173
Loss on epoch 25: 3.747891
Loss on epoch 26: 3.745095
Loss on epoch 27: 3.683056
Loss on epoch 28: 3.568304
Loss on epoch 29: 3.586902
Loss on epoch 30: 3.567074
Epoch 30 : Accuracy on training set: 0.7676436107854631
Epoch 30 : Accuracy on development set: 0.726078799249531
Loss on epoch 31: 3.739871
Loss on epoch 32: 3.476727
Loss on epoch 33: 3.551818
Loss on epoch 34: 3.445897
Loss on epoch 35: 3.488275
Loss on epoch 36: 3.450354
Loss on epoch 37: 3.347149
Loss on epoch 38: 3.460235
Loss on epoch 39: 3.385205
Loss on epoch 40: 3.244319
Epoch 40 : Accuracy on training set: 0.8506447831184056
Epoch 40 : Accuracy on development set: 0.7607879924953096
Loss on epoch 41: 3.261369
Loss on epoch 42: 3.325414
Loss on epoch 43: 3.391697
Loss on epoch 44: 3.326430
Loss on epoch 45: 3.241421
Loss on epoch 46: 3.133819
Loss on epoch 47: 3.140623
Loss on epoch 48: 3.207544
Loss on epoch 49: 3.113834
Accuracy on training set: 0.8599062133645955
Accuracy on develpment set: 0.7682926829268293
Loss on epoch 0: 6.069588
Epoch 0 : Accuracy on training set: 0.6805392731535757
Epoch 0 : Accuracy on development set: 0.6735459662288931
Loss on epoch 1: 5.236228
Loss on epoch 2: 4.632560
Loss on epoch 3: 4.573735
Loss on epoch 4: 4.314031
Loss on epoch 5: 4.262049
Loss on epoch 6: 4.209303
Loss on epoch 7: 4.246139
Loss on epoch 8: 4.173232
Loss on epoch 9: 4.097980
Loss on epoch 10: 4.032389
Epoch 10 : Accuracy on training set: 0.7859320046893318
Epoch 10 : Accuracy on development set: 0.7420262664165104
Loss on epoch 11: 4.127875
Loss on epoch 12: 4.097173
Loss on epoch 13: 4.077072
Loss on epoch 14: 4.004371
Loss on epoch 15: 4.025640
Loss on epoch 16: 3.968973
Loss on epoch 17: 3.902504
Loss on epoch 18: 3.888079
Loss on epoch 19: 3.851709
Loss on epoch 20: 3.801801
Epoch 20 : Accuracy on training set: 0.8182883939038686
Epoch 20 : Accuracy on development set: 0.7711069418386491
Loss on epoch 21: 3.754722
Loss on epoch 22: 3.798394
Loss on epoch 23: 3.844881
Loss on epoch 24: 3.708288
Loss on epoch 25: 3.782957
Loss on epoch 26: 3.662960
Loss on epoch 27: 3.627079
Loss on epoch 28: 3.634371
Loss on epoch 29: 3.554722
Loss on epoch 30: 3.619610
Epoch 30 : Accuracy on training set: 0.8257913247362251
Epoch 30 : Accuracy on development set: 0.7598499061913696
Loss on epoch 31: 3.495815
Loss on epoch 32: 3.578044
Loss on epoch 33: 3.501634
Loss on epoch 34: 3.515421
Loss on epoch 35: 3.461111
Loss on epoch 36: 3.371326
Loss on epoch 37: 3.395339
Loss on epoch 38: 3.384828
Loss on epoch 39: 3.481383
Loss on epoch 40: 3.397082
Epoch 40 : Accuracy on training set: 0.838569753810082
Epoch 40 : Accuracy on development set: 0.774859287054409
Loss on epoch 41: 3.312369
Loss on epoch 42: 3.294364
Loss on epoch 43: 3.276907
Loss on epoch 44: 3.383742
Loss on epoch 45: 3.364066
Loss on epoch 46: 3.195979
Loss on epoch 47: 3.166161
Loss on epoch 48: 3.225129
Loss on epoch 49: 3.160308
Accuracy on training set: 0.8583821805392732
Accuracy on develpment set: 0.7833020637898687
Loss on epoch 0: 6.059280
Epoch 0 : Accuracy on training set: 0.6645955451348183
Epoch 0 : Accuracy on development set: 0.6735459662288931
Loss on epoch 1: 5.276364
Loss on epoch 2: 4.722493
Loss on epoch 3: 4.535439
Loss on epoch 4: 4.400410
Loss on epoch 5: 4.309704
Loss on epoch 6: 4.334202
Loss on epoch 7: 4.225851
Loss on epoch 8: 4.223785
Loss on epoch 9: 4.211031
Loss on epoch 10: 4.126351
Epoch 10 : Accuracy on training set: 0.7906213364595546
Epoch 10 : Accuracy on development set: 0.7551594746716698
Loss on epoch 11: 4.024813
Loss on epoch 12: 4.027923
Loss on epoch 13: 3.946006
Loss on epoch 14: 3.981783
Loss on epoch 15: 3.884500
Loss on epoch 16: 3.853927
Loss on epoch 17: 3.866323
Loss on epoch 18: 3.866594
Loss on epoch 19: 3.882132
Loss on epoch 20: 3.824106
Epoch 20 : Accuracy on training set: 0.8053927315357562
Epoch 20 : Accuracy on development set: 0.7570356472795498
Loss on epoch 21: 3.752429
Loss on epoch 22: 3.713567
Loss on epoch 23: 3.707286
Loss on epoch 24: 3.768318
Loss on epoch 25: 3.653508
Loss on epoch 26: 3.633421
Loss on epoch 27: 3.614515
Loss on epoch 28: 3.559916
Loss on epoch 29: 3.523976
Loss on epoch 30: 3.531236
Epoch 30 : Accuracy on training set: 0.8268464243845252
Epoch 30 : Accuracy on development set: 0.7645403377110694
Loss on epoch 31: 3.499393
Loss on epoch 32: 3.434605
Loss on epoch 33: 3.470021
Loss on epoch 34: 3.362320
Loss on epoch 35: 3.434997
Loss on epoch 36: 3.459614
Loss on epoch 37: 3.405660
Loss on epoch 38: 3.316580
Loss on epoch 39: 3.250591
Loss on epoch 40: 3.291003
Epoch 40 : Accuracy on training set: 0.8542790152403282
Epoch 40 : Accuracy on development set: 0.7711069418386491
Loss on epoch 41: 3.341086
Loss on epoch 42: 3.217415
Loss on epoch 43: 3.240530
Loss on epoch 44: 3.204670
Loss on epoch 45: 3.096390
Loss on epoch 46: 3.137612
Loss on epoch 47: 3.147516
Loss on epoch 48: 3.206074
Loss on epoch 49: 3.237676
Accuracy on training set: 0.8643610785463072
Accuracy on develpment set: 0.7917448405253283
Loss on epoch 0: 6.006958
Epoch 0 : Accuracy on training set: 0.6695193434935521
Epoch 0 : Accuracy on development set: 0.6876172607879925
Loss on epoch 1: 5.270878
Loss on epoch 2: 4.763756
Loss on epoch 3: 4.498759
Loss on epoch 4: 4.390533
Loss on epoch 5: 4.358018
Loss on epoch 6: 4.220098
Loss on epoch 7: 4.237512
Loss on epoch 8: 4.139883
Loss on epoch 9: 4.064391
Loss on epoch 10: 4.232679
Epoch 10 : Accuracy on training set: 0.7852286049237983
Epoch 10 : Accuracy on development set: 0.7626641651031895
Loss on epoch 11: 4.111954
Loss on epoch 12: 3.995545
Loss on epoch 13: 3.923762
Loss on epoch 14: 3.964053
Loss on epoch 15: 3.990491
Loss on epoch 16: 4.013078
Loss on epoch 17: 3.934021
Loss on epoch 18: 3.992269
Loss on epoch 19: 3.922098
Loss on epoch 20: 3.874815
Epoch 20 : Accuracy on training set: 0.8055099648300117
Epoch 20 : Accuracy on development set: 0.7551594746716698
Loss on epoch 21: 3.910912
Loss on epoch 22: 3.777623
Loss on epoch 23: 3.773080
Loss on epoch 24: 3.830022
Loss on epoch 25: 3.736078
Loss on epoch 26: 3.707374
Loss on epoch 27: 3.774467
Loss on epoch 28: 3.729052
Loss on epoch 29: 3.601551
Loss on epoch 30: 3.635984
Epoch 30 : Accuracy on training set: 0.8191090269636577
Epoch 30 : Accuracy on development set: 0.773921200750469
Loss on epoch 31: 3.590880
Loss on epoch 32: 3.600557
Loss on epoch 33: 3.624837
Loss on epoch 34: 3.629059
Loss on epoch 35: 3.519657
Loss on epoch 36: 3.534396
Loss on epoch 37: 3.490864
Loss on epoch 38: 3.614528
Loss on epoch 39: 3.433182
Loss on epoch 40: 3.462320
Epoch 40 : Accuracy on training set: 0.8425556858147714
Epoch 40 : Accuracy on development set: 0.7833020637898687
Loss on epoch 41: 3.380836
Loss on epoch 42: 3.494417
Loss on epoch 43: 3.433235
Loss on epoch 44: 3.380583
Loss on epoch 45: 3.311068
Loss on epoch 46: 3.348398
Loss on epoch 47: 3.302131
Loss on epoch 48: 3.228887
Loss on epoch 49: 3.166035
Accuracy on training set: 0.8562719812426729
Accuracy on develpment set: 0.7833020637898687
Loss on epoch 0: 6.068959
Epoch 0 : Accuracy on training set: 0.6797186400937867
Epoch 0 : Accuracy on development set: 0.6866791744840526
Loss on epoch 1: 5.256824
Loss on epoch 2: 4.689192
Loss on epoch 3: 4.511512
Loss on epoch 4: 4.465474
Loss on epoch 5: 4.336626
Loss on epoch 6: 4.260099
Loss on epoch 7: 4.207058
Loss on epoch 8: 4.189287
Loss on epoch 9: 4.166413
Loss on epoch 10: 4.145687
Epoch 10 : Accuracy on training set: 0.7894490035169989
Epoch 10 : Accuracy on development set: 0.7692307692307693
Loss on epoch 11: 4.081679
Loss on epoch 12: 4.011095
Loss on epoch 13: 4.078334
Loss on epoch 14: 3.971930
Loss on epoch 15: 3.937208
Loss on epoch 16: 3.926242
Loss on epoch 17: 3.927316
Loss on epoch 18: 3.862450
Loss on epoch 19: 3.868272
Loss on epoch 20: 3.808552
Epoch 20 : Accuracy on training set: 0.8069167643610785
Epoch 20 : Accuracy on development set: 0.7645403377110694
Loss on epoch 21: 3.809841
Loss on epoch 22: 3.899294
Loss on epoch 23: 3.768418
Loss on epoch 24: 3.707531
Loss on epoch 25: 3.747471
Loss on epoch 26: 3.712695
Loss on epoch 27: 3.676268
Loss on epoch 28: 3.636425
Loss on epoch 29: 3.618091
Loss on epoch 30: 3.601172
Epoch 30 : Accuracy on training set: 0.8302461899179366
Epoch 30 : Accuracy on development set: 0.7692307692307693
Loss on epoch 31: 3.560081
Loss on epoch 32: 3.632407
Loss on epoch 33: 3.510438
Loss on epoch 34: 3.535148
Loss on epoch 35: 3.483736
Loss on epoch 36: 3.472527
Loss on epoch 37: 3.474875
Loss on epoch 38: 3.440298
Loss on epoch 39: 3.350836
Loss on epoch 40: 3.547415
Epoch 40 : Accuracy on training set: 0.8450175849941384
Epoch 40 : Accuracy on development set: 0.773921200750469
Loss on epoch 41: 3.336366
Loss on epoch 42: 3.245949
Loss on epoch 43: 3.521398
Loss on epoch 44: 3.352930
Loss on epoch 45: 3.280346
Loss on epoch 46: 3.318261
Loss on epoch 47: 3.268709
Loss on epoch 48: 3.189712
Loss on epoch 49: 3.202332
Accuracy on training set: 0.8140679953106682
Accuracy on develpment set: 0.7617260787992496
Loss on epoch 0: 6.107529
Epoch 0 : Accuracy on training set: 0.6774912075029308
Epoch 0 : Accuracy on development set: 0.6819887429643527
Loss on epoch 1: 5.290294
Loss on epoch 2: 4.722012
Loss on epoch 3: 4.498312
Loss on epoch 4: 4.365027
Loss on epoch 5: 4.296637
Loss on epoch 6: 4.202307
Loss on epoch 7: 4.162228
Loss on epoch 8: 4.137473
Loss on epoch 9: 4.215216
Loss on epoch 10: 4.273950
Epoch 10 : Accuracy on training set: 0.777725674091442
Epoch 10 : Accuracy on development set: 0.7532833020637899
Loss on epoch 11: 4.177293
Loss on epoch 12: 4.058360
Loss on epoch 13: 4.053787
Loss on epoch 14: 4.006847
Loss on epoch 15: 3.983759
Loss on epoch 16: 3.857514
Loss on epoch 17: 3.883886
Loss on epoch 18: 3.899296
Loss on epoch 19: 3.811007
Loss on epoch 20: 3.793742
Epoch 20 : Accuracy on training set: 0.8124267291910903
Epoch 20 : Accuracy on development set: 0.7664165103189493
Loss on epoch 21: 3.827559
Loss on epoch 22: 3.802891
Loss on epoch 23: 3.812030
Loss on epoch 24: 3.730242
Loss on epoch 25: 3.737639
Loss on epoch 26: 3.616800
Loss on epoch 27: 3.630085
Loss on epoch 28: 3.619795
Loss on epoch 29: 3.709431
Loss on epoch 30: 3.666655
Epoch 30 : Accuracy on training set: 0.8287221570926143
Epoch 30 : Accuracy on development set: 0.7598499061913696
Loss on epoch 31: 3.478485
Loss on epoch 32: 3.477273
Loss on epoch 33: 3.639387
Loss on epoch 34: 3.605138
Loss on epoch 35: 3.560323
Loss on epoch 36: 3.573511
Loss on epoch 37: 3.409921
Loss on epoch 38: 3.494375
Loss on epoch 39: 3.501552
Loss on epoch 40: 3.493513
Epoch 40 : Accuracy on training set: 0.842672919109027
Epoch 40 : Accuracy on development set: 0.775797373358349
Loss on epoch 41: 3.353297
Loss on epoch 42: 3.321701
Loss on epoch 43: 3.325164
Loss on epoch 44: 3.408230
Loss on epoch 45: 3.270799
Loss on epoch 46: 3.304308
Loss on epoch 47: 3.223829
Loss on epoch 48: 3.165009
Loss on epoch 49: 3.282956
Accuracy on training set: 0.82989449003517
Accuracy on develpment set: 0.7617260787992496
Loss on epoch 0: 6.121671
Epoch 0 : Accuracy on training set: 0.6316529894490035
Epoch 0 : Accuracy on development set: 0.6407129455909943
Loss on epoch 1: 5.385486
Loss on epoch 2: 4.866867
Loss on epoch 3: 4.554539
Loss on epoch 4: 4.508377
Loss on epoch 5: 4.420247
Loss on epoch 6: 4.313962
Loss on epoch 7: 4.305667
Loss on epoch 8: 4.230549
Loss on epoch 9: 4.255131
Loss on epoch 10: 4.163539
Epoch 10 : Accuracy on training set: 0.788042203985932
Epoch 10 : Accuracy on development set: 0.7598499061913696
Loss on epoch 11: 4.094065
Loss on epoch 12: 4.025465
Loss on epoch 13: 3.994657
Loss on epoch 14: 4.081552
Loss on epoch 15: 3.964447
Loss on epoch 16: 3.904579
Loss on epoch 17: 3.961298
Loss on epoch 18: 3.889319
Loss on epoch 19: 3.896912
Loss on epoch 20: 3.958447
Epoch 20 : Accuracy on training set: 0.8093786635404455
Epoch 20 : Accuracy on development set: 0.775797373358349
Loss on epoch 21: 3.831643
Loss on epoch 22: 3.859622
Loss on epoch 23: 3.866823
Loss on epoch 24: 3.791225
Loss on epoch 25: 3.697219
Loss on epoch 26: 3.642029
Loss on epoch 27: 3.745254
Loss on epoch 28: 3.645500
Loss on epoch 29: 3.636575
Loss on epoch 30: 3.684114
Epoch 30 : Accuracy on training set: 0.8125439624853459
Epoch 30 : Accuracy on development set: 0.7645403377110694
Loss on epoch 31: 3.616281
Loss on epoch 32: 3.635184
Loss on epoch 33: 3.641367
Loss on epoch 34: 3.617249
Loss on epoch 35: 3.608766
Loss on epoch 36: 3.547536
Loss on epoch 37: 3.456306
Loss on epoch 38: 3.434091
Loss on epoch 39: 3.490180
Loss on epoch 40: 3.514411
Epoch 40 : Accuracy on training set: 0.8314185228604923
Epoch 40 : Accuracy on development set: 0.7701688555347092
Loss on epoch 41: 3.414677
Loss on epoch 42: 3.395082
Loss on epoch 43: 3.429076
Loss on epoch 44: 3.418113
Loss on epoch 45: 3.347549
Loss on epoch 46: 3.211751
Loss on epoch 47: 3.408018
Loss on epoch 48: 3.343232
Loss on epoch 49: 3.404593
Accuracy on training set: 0.849706916764361
Accuracy on develpment set: 0.773921200750469
Loss on epoch 0: 6.350413
Epoch 0 : Accuracy on training set: 0.5713950762016413
Epoch 0 : Accuracy on development set: 0.5787992495309568
Loss on epoch 1: 5.845367
Loss on epoch 2: 5.057069
Loss on epoch 3: 4.628642
Loss on epoch 4: 4.473840
Loss on epoch 5: 4.443087
Loss on epoch 6: 4.265932
Loss on epoch 7: 4.195521
Loss on epoch 8: 4.440676
Loss on epoch 9: 4.223155
Loss on epoch 10: 4.170138
Epoch 10 : Accuracy on training set: 0.7920281359906214
Epoch 10 : Accuracy on development set: 0.7598499061913696
Loss on epoch 11: 4.118472
Loss on epoch 12: 4.022291
Loss on epoch 13: 3.992166
Loss on epoch 14: 3.984035
Loss on epoch 15: 4.036170
Loss on epoch 16: 3.953011
Loss on epoch 17: 3.888670
Loss on epoch 18: 3.863259
Loss on epoch 19: 3.869575
Loss on epoch 20: 3.991618
Epoch 20 : Accuracy on training set: 0.8139507620164127
Epoch 20 : Accuracy on development set: 0.7664165103189493
Loss on epoch 21: 3.803633
Loss on epoch 22: 3.736144
Loss on epoch 23: 3.715740
Loss on epoch 24: 3.771406
Loss on epoch 25: 3.795650
Loss on epoch 26: 3.812025
Loss on epoch 27: 3.755770
Loss on epoch 28: 3.655171
Loss on epoch 29: 3.598972
Loss on epoch 30: 3.695205
Epoch 30 : Accuracy on training set: 0.8078546307151231
Epoch 30 : Accuracy on development set: 0.7504690431519699
Loss on epoch 31: 3.779064
Loss on epoch 32: 3.562251
Loss on epoch 33: 3.532666
Loss on epoch 34: 3.566604
Loss on epoch 35: 3.619196
Loss on epoch 36: 3.551960
Loss on epoch 37: 3.516354
Loss on epoch 38: 3.526670
Loss on epoch 39: 3.516828
Loss on epoch 40: 3.577577
Epoch 40 : Accuracy on training set: 0.8063305978898007
Epoch 40 : Accuracy on development set: 0.7729831144465291
Loss on epoch 41: 3.606577
Loss on epoch 42: 3.537511
Loss on epoch 43: 3.523566
Loss on epoch 44: 3.399225
Loss on epoch 45: 3.418050
Loss on epoch 46: 3.352947
Loss on epoch 47: 3.335878
Loss on epoch 48: 3.324419
Loss on epoch 49: 3.265950
Accuracy on training set: 0.8474794841735053
Accuracy on develpment set: 0.7701688555347092
Average accuracy on test set across all 10 evaluations: 0.90234521575985
Standard deviation: 0.04104157062343849
```

6 Experimental evaluations on the IMDB dataset.

Run the same 4 evaluations on the IMDB dataset.

[ ]:
```
train_path = 'data/imdb/train.txt'
dev_path = 'data/imdb/dev.txt'
test_path = 'data/imdb/test.txt'

test_output_path = 'test-imdb.output.txt'

## YOUR CODE HERE
word_vectors = read_word_embeddings(word_vecs_path)
```

```
word_indexer = word_vectors.word_indexer

train_exs = read_and_index_sentiment_examples(train_path, word_indexer)
dev_exs = read_and_index_sentiment_examples(dev_path, word_indexer)
test_exs = read_and_index_sentiment_examples(test_path, word_indexer)

print(repr(len(train_exs)) + " / " +
      repr(len(dev_exs)) + " / " +
      repr(len(test_exs)) + " train / dev / test examples")
```

```
Read in 30135 vectors of size 300
1500 / 1500 / 1500 train / dev / test examples
```

6.0.1 Model 1 Evaluation:

```
[ ]: random.seed(1)
np.random.seed(1)
torch.manual_seed(1)

hp = HyperParams(lstm_size = 50, # hidden units in lstm
                 hidden_size = 50, # hidden size of fully-connected layer
                 lstm_layers = 1, # layers in lstm
                 drop_out = 0.5, # dropout rate
                 num_epochs = 50, # number of epochs for SGD-based procedure
                 batch_size = 1024, # examples in a minibatch
                 seq_max_len = 60) # maximum length of an example sequence
use_average = False
bidirectional = False

# Train RNN model.
model1 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
 ↪use_average, bidirectional)

# Generate RNN model predictions for test set.
embeddings_vec = np.array(word_vectors.vectors).astype(float)
test_exs_predicted = eval_model(model1, test_exs, embeddings_vec, hp.
 ↪seq_max_len, pred_only = True)

# Write the test set output
for i, ex in enumerate(test_exs):
    ex.label = int(test_exs_predicted[i])
write_sentiment_examples(test_exs, test_output_path, word_indexer)

print("Prediction written to file for IMDB dataset.")
```

```
Loss on epoch 0: 1.453330
Epoch 0 : Accuracy on training set: 0.5213333333333333
Epoch 0 : Accuracy on development set: 0.49
Loss on epoch 1: 1.397238
Loss on epoch 2: 1.394922
Loss on epoch 3: 1.384270
Loss on epoch 4: 1.381265
Loss on epoch 5: 1.375511
Loss on epoch 6: 1.355744
Loss on epoch 7: 1.354589
Loss on epoch 8: 1.338645
Loss on epoch 9: 1.325564
Loss on epoch 10: 1.310576
Epoch 10 : Accuracy on training set: 0.63
Epoch 10 : Accuracy on development set: 0.5306666666666666
Loss on epoch 11: 1.307331
Loss on epoch 12: 1.287133
Loss on epoch 13: 1.279851
Loss on epoch 14: 1.267624
Loss on epoch 15: 1.241794
Loss on epoch 16: 1.211682
Loss on epoch 17: 1.201796
Loss on epoch 18: 1.185211
Loss on epoch 19: 1.165342
Loss on epoch 20: 1.164737
Epoch 20 : Accuracy on training set: 0.6926666666666667
Epoch 20 : Accuracy on development set: 0.6126666666666667
Loss on epoch 21: 1.148687
Loss on epoch 22: 1.138891
Loss on epoch 23: 1.100161
Loss on epoch 24: 1.082157
Loss on epoch 25: 1.057653
Loss on epoch 26: 1.006938
Loss on epoch 27: 1.044793
Loss on epoch 28: 1.036073
Loss on epoch 29: 1.004239
Loss on epoch 30: 1.027270
Epoch 30 : Accuracy on training set: 0.698
Epoch 30 : Accuracy on development set: 0.6086666666666667
Loss on epoch 31: 1.085254
Loss on epoch 32: 1.023671
Loss on epoch 33: 0.961800
Loss on epoch 34: 0.978977
Loss on epoch 35: 0.943840
Loss on epoch 36: 0.922381
Loss on epoch 37: 0.900915
Loss on epoch 38: 0.895853
Loss on epoch 39: 0.877834
Loss on epoch 40: 0.882706
Epoch 40 : Accuracy on training set: 0.8253333333333334
Epoch 40 : Accuracy on development set: 0.6353333333333333
Loss on epoch 41: 0.831574
Loss on epoch 42: 0.797713
Loss on epoch 43: 0.800961
Loss on epoch 44: 0.745474
Loss on epoch 45: 0.779768
Loss on epoch 46: 0.795925
Loss on epoch 47: 0.755777
Loss on epoch 48: 0.678322
Loss on epoch 49: 0.690266
Accuracy on training set: 0.8313333333333334
Accuracy on develpment set: 0.6313333333333333
Prediction written to file for IMDB dataset.
```

6.0.2 Model 2 Evaluation:

```
[ ]: random.seed(1)
     np.random.seed(1)
     torch.manual_seed(1)

     ## YOUR CODE HERE
     hp = HyperParams(lstm_size = 50, # hidden units in lstm
                      hidden_size = 50, # hidden size of fully-connected layer
                      lstm_layers = 1, # layers in lstm
                      drop_out = 0.5, # dropout rate
                      num_epochs = 50, # number of epochs for SGD-based procedure
                      batch_size = 1024, # examples in a minibatch
                      seq_max_len = 60) # maximum length of an example sequence
     use_average = True
     bidirectional = False

     # Train RNN model.
     model2 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

     # Generate RNN model predictions for test set.
     embeddings_vec = np.array(word_vectors.vectors).astype(float)
     test_exs_predicted = eval_model(model2, test_exs, embeddings_vec, hp.
      ↪seq_max_len, pred_only = True)

     # Write the test set output
     for i, ex in enumerate(test_exs):
         ex.label = int(test_exs_predicted[i])
     write_sentiment_examples(test_exs, test_output_path, word_indexer)

     print("Prediction written to file for IMDB dataset.")
```

```
Loss on epoch 0: 1.458653
```

```
Epoch 0 : Accuracy on training set: 0.5186666666666667
Epoch 0 : Accuracy on development set: 0.49666666666666665
Loss on epoch 1: 1.401299
Loss on epoch 2: 1.400150
Loss on epoch 3: 1.387287
Loss on epoch 4: 1.391608
Loss on epoch 5: 1.387948
Loss on epoch 6: 1.371438
Loss on epoch 7: 1.375570
Loss on epoch 8: 1.364308
Loss on epoch 9: 1.348404
Loss on epoch 10: 1.333620
Epoch 10 : Accuracy on training set: 0.6133333333333333
Epoch 10 : Accuracy on development set: 0.61
Loss on epoch 11: 1.312866
Loss on epoch 12: 1.284701
Loss on epoch 13: 1.276582
Loss on epoch 14: 1.259226
Loss on epoch 15: 1.250175
Loss on epoch 16: 1.204935
Loss on epoch 17: 1.201647
Loss on epoch 18: 1.160905
Loss on epoch 19: 1.119385
Loss on epoch 20: 1.111832
Epoch 20 : Accuracy on training set: 0.7186666666666667
Epoch 20 : Accuracy on development set: 0.7053333333333334
Loss on epoch 21: 1.088056
Loss on epoch 22: 1.112120
Loss on epoch 23: 1.105768
Loss on epoch 24: 1.077549
Loss on epoch 25: 1.081267
Loss on epoch 26: 1.027753
Loss on epoch 27: 1.058153
Loss on epoch 28: 1.091065
Loss on epoch 29: 1.035319
Loss on epoch 30: 1.057394
Epoch 30 : Accuracy on training set: 0.7666666666666667
Epoch 30 : Accuracy on development set: 0.7233333333333334
Loss on epoch 31: 1.038187
Loss on epoch 32: 0.989528
Loss on epoch 33: 1.000830
Loss on epoch 34: 0.977862
Loss on epoch 35: 0.989094
Loss on epoch 36: 1.015077
Loss on epoch 37: 0.961882
Loss on epoch 38: 0.929449
Loss on epoch 39: 0.977028
Loss on epoch 40: 0.905599
```

```
Epoch 40 : Accuracy on training set: 0.782
Epoch 40 : Accuracy on development set: 0.732
Loss on epoch 41: 0.924071
Loss on epoch 42: 0.950483
Loss on epoch 43: 0.930951
Loss on epoch 44: 0.981632
Loss on epoch 45: 0.926642
Loss on epoch 46: 0.876290
Loss on epoch 47: 0.935196
Loss on epoch 48: 0.943648
Loss on epoch 49: 0.909405
Accuracy on training set: 0.7953333333333333
Accuracy on develpment set: 0.7166666666666667
Prediction written to file for IMDB dataset.
```

6.0.3 Model 3 Evaluation:

```
[ ]: random.seed(1)
     np.random.seed(1)
     torch.manual_seed(1)

     ## YOUR CODE HERE
     hp = HyperParams(lstm_size = 50, # hidden units in lstm
                      hidden_size = 50, # hidden size of fully-connected layer
                      lstm_layers = 1, # layers in lstm
                      drop_out = 0.5, # dropout rate
                      num_epochs = 50, # number of epochs for SGD-based procedure
                      batch_size = 1024, # examples in a minibatch
                      seq_max_len = 60) # maximum length of an example sequence
     use_average = False
     bidirectional = True

     # Train RNN model.
     model3 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

     # Generate RNN model predictions for test set.
     embeddings_vec = np.array(word_vectors.vectors).astype(float)
     test_exs_predicted = eval_model(model3, test_exs, embeddings_vec, hp.
      ↪seq_max_len, pred_only = True)

     # Write the test set output
     for i, ex in enumerate(test_exs):
         ex.label = int(test_exs_predicted[i])
     write_sentiment_examples(test_exs, test_output_path, word_indexer)

     print("Prediction written to file for IMDB dataset.")
```

```
Loss on epoch 0: 1.584871
Epoch 0 : Accuracy on training set: 0.5053333333333333
Epoch 0 : Accuracy on development set: 0.504
Loss on epoch 1: 1.424024
Loss on epoch 2: 1.397311
Loss on epoch 3: 1.371477
Loss on epoch 4: 1.371694
Loss on epoch 5: 1.377411
Loss on epoch 6: 1.355382
Loss on epoch 7: 1.350052
Loss on epoch 8: 1.328987
Loss on epoch 9: 1.309001
Loss on epoch 10: 1.289134
Epoch 10 : Accuracy on training set: 0.642
Epoch 10 : Accuracy on development set: 0.5553333333333333
Loss on epoch 11: 1.273834
Loss on epoch 12: 1.237192
Loss on epoch 13: 1.220540
Loss on epoch 14: 1.184478
Loss on epoch 15: 1.177365
Loss on epoch 16: 1.116286
Loss on epoch 17: 1.091381
Loss on epoch 18: 1.058752
Loss on epoch 19: 1.018903
Loss on epoch 20: 1.008936
Epoch 20 : Accuracy on training set: 0.7746666666666666
Epoch 20 : Accuracy on development set: 0.648
Loss on epoch 21: 0.944952
Loss on epoch 22: 1.041088
Loss on epoch 23: 1.051450
Loss on epoch 24: 0.995997
Loss on epoch 25: 0.996276
Loss on epoch 26: 0.995251
Loss on epoch 27: 0.922184
Loss on epoch 28: 0.935127
Loss on epoch 29: 0.868734
Loss on epoch 30: 0.804316
Epoch 30 : Accuracy on training set: 0.8473333333333334
Epoch 30 : Accuracy on development set: 0.6713333333333333
Loss on epoch 31: 0.882988
Loss on epoch 32: 0.828649
Loss on epoch 33: 0.817542
Loss on epoch 34: 0.759968
Loss on epoch 35: 0.778511
Loss on epoch 36: 0.727214
Loss on epoch 37: 0.655972
Loss on epoch 38: 0.665667
Loss on epoch 39: 0.629431
Loss on epoch 40: 0.614519
Epoch 40 : Accuracy on training set: 0.8913333333333333
Epoch 40 : Accuracy on development set: 0.6653333333333333
Loss on epoch 41: 0.559164
Loss on epoch 42: 0.510009
Loss on epoch 43: 0.483380
Loss on epoch 44: 0.560413
Loss on epoch 45: 0.521942
Loss on epoch 46: 0.434643
Loss on epoch 47: 0.396630
Loss on epoch 48: 0.344345
Loss on epoch 49: 0.302382
Accuracy on training set: 0.9566666666666667
Accuracy on develpment set: 0.6846666666666666
Prediction written to file for IMDB dataset.
```

6.0.4 Model 4 Evaluation:

```
[ ]: random.seed(1)
     np.random.seed(1)
     torch.manual_seed(1)

     ## YOUR CODE HERE
     hp = HyperParams(lstm_size = 50, # hidden units in lstm
                      hidden_size = 50, # hidden size of fully-connected layer
                      lstm_layers = 1, # layers in lstm
                      drop_out = 0.5, # dropout rate
                      num_epochs = 50, # number of epochs for SGD-based procedure
                      batch_size = 1024, # examples in a minibatch
                      seq_max_len = 60) # maximum length of an example sequence
     use_average = True
     bidirectional = True

     # Train RNN model.
     model4 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

     # Generate RNN model predictions for test set.
     embeddings_vec = np.array(word_vectors.vectors).astype(float)
     test_exs_predicted = eval_model(model4, test_exs, embeddings_vec, hp.
      ↪seq_max_len, pred_only = True)

     # Write the test set output
     for i, ex in enumerate(test_exs):
         ex.label = int(test_exs_predicted[i])
     write_sentiment_examples(test_exs, test_output_path, word_indexer)
```

```python
print("Prediction written to file for IMDB dataset.")
```

```
Loss on epoch 0: 1.586349
Epoch 0 : Accuracy on training set: 0.49666666666666665
Epoch 0 : Accuracy on development set: 0.498
Loss on epoch 1: 1.425531
Loss on epoch 2: 1.404426
Loss on epoch 3: 1.385035
Loss on epoch 4: 1.386975
Loss on epoch 5: 1.392177
Loss on epoch 6: 1.378059
Loss on epoch 7: 1.380755
Loss on epoch 8: 1.370604
Loss on epoch 9: 1.360894
Loss on epoch 10: 1.361191
Epoch 10 : Accuracy on training set: 0.6086666666666667
Epoch 10 : Accuracy on development set: 0.6293333333333333
Loss on epoch 11: 1.343627
Loss on epoch 12: 1.323903
Loss on epoch 13: 1.303303
Loss on epoch 14: 1.283770
Loss on epoch 15: 1.240168
Loss on epoch 16: 1.189801
Loss on epoch 17: 1.182481
Loss on epoch 18: 1.111729
Loss on epoch 19: 1.071498
Loss on epoch 20: 1.032303
Epoch 20 : Accuracy on training set: 0.702
Epoch 20 : Accuracy on development set: 0.6873333333333334
Loss on epoch 21: 1.091102
Loss on epoch 22: 0.986199
Loss on epoch 23: 0.957805
Loss on epoch 24: 0.971484
Loss on epoch 25: 0.956278
Loss on epoch 26: 0.904816
Loss on epoch 27: 0.876601
Loss on epoch 28: 0.891765
Loss on epoch 29: 0.810309
Loss on epoch 30: 0.798981
Epoch 30 : Accuracy on training set: 0.794
Epoch 30 : Accuracy on development set: 0.7733333333333333
Loss on epoch 31: 0.836227
Loss on epoch 32: 0.806355
Loss on epoch 33: 0.831540
Loss on epoch 34: 0.772063
Loss on epoch 35: 0.777711
Loss on epoch 36: 0.750377
Loss on epoch 37: 0.726027
Loss on epoch 38: 0.732040
Loss on epoch 39: 0.780060
Loss on epoch 40: 0.787553
Epoch 40 : Accuracy on training set: 0.8266666666666667
Epoch 40 : Accuracy on development set: 0.7766666666666666
Loss on epoch 41: 0.813430
Loss on epoch 42: 0.733097
Loss on epoch 43: 0.715621
Loss on epoch 44: 0.750937
Loss on epoch 45: 0.692192
Loss on epoch 46: 0.639561
Loss on epoch 47: 0.640958
Loss on epoch 48: 0.635280
Loss on epoch 49: 0.602479
Accuracy on training set: 0.882
Accuracy on develpment set: 0.7866666666666666
Prediction written to file for IMDB dataset.
```

6.1 [5111] Cross-domain performance

Compare the performance of the Bidirectional LSTM with state averaging on the IMDB test set in two scenarios:

1. The model is trained on the IMDB training data.
2. The model is trained on the Rotten Tomatoes data.

```
[ ]: ## YOUR CODE HERE

     ## Setting up data:
     train_path_imdb = 'data/imdb/train.txt'
     train_path_rt = 'data/rt/train.txt'
     dev_path = 'data/imdb/dev.txt'
     test_path = 'data/imdb/test.txt'

     test_output_path = 'test-imdb.output.txt'

     ## YOUR CODE HERE
     word_vectors = read_word_embeddings(word_vecs_path)
     word_indexer = word_vectors.word_indexer

     train_exs_imdb = read_and_index_sentiment_examples(train_path_imdb,␣
       ↪word_indexer)
     train_exs_rt = read_and_index_sentiment_examples(train_path_rt, word_indexer)
     dev_exs = read_and_index_sentiment_examples(dev_path, word_indexer)
     test_exs = read_and_index_sentiment_examples(test_path, word_indexer)

     print(repr(len(train_exs)) + " / " +
           repr(len(dev_exs)) + " / " +
           repr(len(test_exs)) + " train / dev / test examples")
```

```
Read in 30135 vectors of size 300
1500 / 1500 / 1500 train / dev / test examples
```

6.1.1 1. Training on IMDB training data

```
[ ]: random.seed(1)
     np.random.seed(1)
     torch.manual_seed(1)

     ## YOUR CODE HERE
     hp = HyperParams(lstm_size = 50, # hidden units in lstm
                      hidden_size = 50, # hidden size of fully-connected layer
                      lstm_layers = 1, # layers in lstm
                      drop_out = 0.5, # dropout rate
                      num_epochs = 50, # number of epochs for SGD-based procedure
                      batch_size = 1024, # examples in a minibatch
                      seq_max_len = 60) # maximum length of an example sequence
     use_average = True
     bidirectional = True

     # Train RNN model.
     model4 = train_model(hp, train_exs_imdb, dev_exs, test_exs, word_vectors,␣
      ↪use_average, bidirectional)

     # Generate RNN model predictions for test set.
     embeddings_vec = np.array(word_vectors.vectors).astype(float)
     test_exs_predicted = eval_model(model4, test_exs, embeddings_vec, hp.
      ↪seq_max_len, pred_only = True)
     accuracy = eval_model(model4, test_exs, embeddings_vec, hp.seq_max_len,␣
      ↪pred_only = False)

     print("Predictions saved. Accuracy on test data: ", accuracy)
```

```
Loss on epoch 0: 1.586349
Epoch 0 : Accuracy on training set: 0.49666666666666665
Epoch 0 : Accuracy on development set: 0.498
Loss on epoch 1: 1.425531
Loss on epoch 2: 1.404426
Loss on epoch 3: 1.385035
Loss on epoch 4: 1.386975
Loss on epoch 5: 1.392177
Loss on epoch 6: 1.378059
Loss on epoch 7: 1.380755
Loss on epoch 8: 1.370604
Loss on epoch 9: 1.360894
Loss on epoch 10: 1.361191
Epoch 10 : Accuracy on training set: 0.6086666666666667
Epoch 10 : Accuracy on development set: 0.6293333333333333
Loss on epoch 11: 1.343627
Loss on epoch 12: 1.323903
Loss on epoch 13: 1.303303
Loss on epoch 14: 1.283770
Loss on epoch 15: 1.240168
Loss on epoch 16: 1.189801
Loss on epoch 17: 1.182481
Loss on epoch 18: 1.111729
Loss on epoch 19: 1.071498
Loss on epoch 20: 1.032303
Epoch 20 : Accuracy on training set: 0.702
Epoch 20 : Accuracy on development set: 0.6873333333333334
Loss on epoch 21: 1.091102
Loss on epoch 22: 0.986199
Loss on epoch 23: 0.957805
Loss on epoch 24: 0.971484
Loss on epoch 25: 0.956278
Loss on epoch 26: 0.904816
Loss on epoch 27: 0.876601
Loss on epoch 28: 0.891765
Loss on epoch 29: 0.810309
Loss on epoch 30: 0.798981
Epoch 30 : Accuracy on training set: 0.794
Epoch 30 : Accuracy on development set: 0.7733333333333333
Loss on epoch 31: 0.836227
Loss on epoch 32: 0.806355
Loss on epoch 33: 0.831540
Loss on epoch 34: 0.772063
Loss on epoch 35: 0.777711
Loss on epoch 36: 0.750377
Loss on epoch 37: 0.726027
Loss on epoch 38: 0.732040
Loss on epoch 39: 0.780060
Loss on epoch 40: 0.787553
Epoch 40 : Accuracy on training set: 0.8266666666666667
Epoch 40 : Accuracy on development set: 0.7766666666666666
Loss on epoch 41: 0.813430
Loss on epoch 42: 0.733097
Loss on epoch 43: 0.715621
Loss on epoch 44: 0.750937
Loss on epoch 45: 0.692192
Loss on epoch 46: 0.639561
Loss on epoch 47: 0.640958
Loss on epoch 48: 0.635280
Loss on epoch 49: 0.602479
Accuracy on training set: 0.882
Accuracy on develpment set: 0.7866666666666666
Predictions saved. Accuracy on test data:  0.7666666666666667
```

6.1.2 2. Training on Rotten Tomatoes data

```
[ ]: random.seed(1)
np.random.seed(1)
torch.manual_seed(1)

## YOUR CODE HERE
hp = HyperParams(lstm_size = 50, # hidden units in lstm
                 hidden_size = 50, # hidden size of fully-connected layer
                 lstm_layers = 1, # layers in lstm
                 drop_out = 0.5, # dropout rate
                 num_epochs = 50, # number of epochs for SGD-based procedure
                 batch_size = 1024, # examples in a minibatch
                 seq_max_len = 60) # maximum length of an example sequence
use_average = True
bidirectional = True

# Train RNN model.
model4 = train_model(hp, train_exs_rt, dev_exs, test_exs, word_vectors,␣
 ↪use_average, bidirectional)

# Generate RNN model predictions for test set.
embeddings_vec = np.array(word_vectors.vectors).astype(float)
test_exs_predicted = eval_model(model4, test_exs, embeddings_vec, hp.
 ↪seq_max_len, pred_only = True)
accuracy = eval_model(model4, test_exs, embeddings_vec, hp.seq_max_len,␣
 ↪pred_only = False)

print("Predictions saved. Accuracy on test data: ", accuracy)
```

```
Loss on epoch 0: 6.331739
Epoch 0 : Accuracy on training set: 0.5837045720984759
Epoch 0 : Accuracy on development set: 0.5526666666666666
Loss on epoch 1: 5.841878
Loss on epoch 2: 5.033062
Loss on epoch 3: 4.668347
Loss on epoch 4: 4.512972
Loss on epoch 5: 4.385176
Loss on epoch 6: 4.413477
Loss on epoch 7: 4.308627
Loss on epoch 8: 4.253889
Loss on epoch 9: 4.257465
Loss on epoch 10: 4.199516
Epoch 10 : Accuracy on training set: 0.7808909730363424
Epoch 10 : Accuracy on development set: 0.784
Loss on epoch 11: 4.117451
Loss on epoch 12: 4.030150
Loss on epoch 13: 4.066597
Loss on epoch 14: 3.984642
Loss on epoch 15: 4.025867
Loss on epoch 16: 3.963529
Loss on epoch 17: 3.932011
Loss on epoch 18: 3.927685
Loss on epoch 19: 3.879160
Loss on epoch 20: 3.931827
Epoch 20 : Accuracy on training set: 0.8128956623681125
Epoch 20 : Accuracy on development set: 0.736
Loss on epoch 21: 3.766663
Loss on epoch 22: 3.720750
Loss on epoch 23: 3.754652
Loss on epoch 24: 3.766137
Loss on epoch 25: 3.812835
Loss on epoch 26: 3.698266
Loss on epoch 27: 3.584020
Loss on epoch 28: 3.678678
Loss on epoch 29: 3.602871
Loss on epoch 30: 3.597236
Epoch 30 : Accuracy on training set: 0.8252051582649472
Epoch 30 : Accuracy on development set: 0.74
Loss on epoch 31: 3.597928
Loss on epoch 32: 3.508547
Loss on epoch 33: 3.576733
Loss on epoch 34: 3.542585
Loss on epoch 35: 3.444560
Loss on epoch 36: 3.392123
Loss on epoch 37: 3.388881
Loss on epoch 38: 3.372636
Loss on epoch 39: 3.364726
Loss on epoch 40: 3.339032
Epoch 40 : Accuracy on training set: 0.8467760844079719
Epoch 40 : Accuracy on development set: 0.7293333333333333
Loss on epoch 41: 3.296636
Loss on epoch 42: 3.493911
Loss on epoch 43: 3.376744
Loss on epoch 44: 3.181008
Loss on epoch 45: 3.323633
Loss on epoch 46: 3.160640
Loss on epoch 47: 3.384176
Loss on epoch 48: 3.221092
Loss on epoch 49: 3.152903
```

```
Accuracy on training set: 0.8399765533411488
Accuracy on develpment set: 0.6586666666666666
Predictions saved. Accuracy on test data:  0.6346666666666667
```

6.2 Bonus points

Anything extra goes here.

```
[8]: train_path = 'data/imdb/train.txt'
     dev_path = 'data/imdb/dev.txt'
     test_path = 'data/imdb/test.txt'
     word_vecs_path = 'data/glove.6B.300d-relativized.txt'

     test_output_path = 'test-imdb_bonus.output.txt'

     ## YOUR CODE HERE
     word_vectors = read_word_embeddings(word_vecs_path)
     word_indexer = word_vectors.word_indexer

     train_exs = read_and_index_sentiment_examples(train_path, word_indexer)
     dev_exs = read_and_index_sentiment_examples(dev_path, word_indexer)
     test_exs = read_and_index_sentiment_examples(test_path, word_indexer)

     print(repr(len(train_exs)) + " / " +
           repr(len(dev_exs)) + " / " +
           repr(len(test_exs)) + " train / dev / test examples")
```

```
Read in 30135 vectors of size 300
1500 / 1500 / 1500 train / dev / test examples
```

```
[10]: seeds = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
      accuracies = []

      for s in seeds:

        random.seed(s)
        np.random.seed(s)
        torch.manual_seed(s)

        hp = HyperParams(lstm_size = 50, # hidden units in lstm
                         hidden_size = 50, # hidden size of fully-connected layer
                         lstm_layers = 1, # layers in lstm
                         drop_out = 0.5, # dropout rate
                         num_epochs = 50, # number of epochs for SGD-based procedure
                         batch_size = 1024, # examples in a minibatch
                         seq_max_len = 60) # maximum length of an example sequence
        use_average = True
        bidirectional = True
```

```
  # Train RNN model.
  model4 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
↪use_average, bidirectional)

  # Generate RNN model predictions for test set.
  embeddings_vec = np.array(word_vectors.vectors).astype(float)
  test_exs_predicted = eval_model(model4, test_exs, embeddings_vec, hp.
↪seq_max_len, pred_only = True)
  acc = eval_model(model4, test_exs, embeddings_vec, hp.seq_max_len, pred_only␣
↪= False)

  accuracies.append(acc)

avg_acc = np.mean(accuracies)
print('Average accuracy on test set across all 10 evaluations:', avg_acc)
avg_std = np.std(accuracies)
print('Standard deviation:', avg_std)
print("Accuracies by seed: ", accuracies)
```

```
Loss on epoch 0: 1.481982
Epoch 0 : Accuracy on training set: 0.486
Epoch 0 : Accuracy on development set: 0.48866666666666667
Loss on epoch 1: 1.400995
Loss on epoch 2: 1.406399
Loss on epoch 3: 1.381155
Loss on epoch 4: 1.399291
Loss on epoch 5: 1.386583
Loss on epoch 6: 1.367662
Loss on epoch 7: 1.366127
Loss on epoch 8: 1.354667
Loss on epoch 9: 1.353192
Loss on epoch 10: 1.326881
Epoch 10 : Accuracy on training set: 0.632
Epoch 10 : Accuracy on development set: 0.632
Loss on epoch 11: 1.311159
Loss on epoch 12: 1.257324
Loss on epoch 13: 1.228301
Loss on epoch 14: 1.192894
Loss on epoch 15: 1.132780
Loss on epoch 16: 1.113165
Loss on epoch 17: 1.056873
Loss on epoch 18: 1.043456
Loss on epoch 19: 1.081172
Loss on epoch 20: 1.032774
Epoch 20 : Accuracy on training set: 0.7993333333333333
Epoch 20 : Accuracy on development set: 0.7786666666666666
```

```
Loss on epoch 21: 0.984801
Loss on epoch 22: 0.970217
Loss on epoch 23: 0.946757
Loss on epoch 24: 0.883174
Loss on epoch 25: 0.841264
Loss on epoch 26: 0.823028
Loss on epoch 27: 0.776420
Loss on epoch 28: 0.829437
Loss on epoch 29: 0.780328
Loss on epoch 30: 0.777196
Epoch 30 : Accuracy on training set: 0.858
Epoch 30 : Accuracy on development set: 0.792
Loss on epoch 31: 0.741546
Loss on epoch 32: 0.761225
Loss on epoch 33: 0.692864
Loss on epoch 34: 0.852025
Loss on epoch 35: 0.732317
Loss on epoch 36: 0.747871
Loss on epoch 37: 0.726735
Loss on epoch 38: 0.761829
Loss on epoch 39: 0.731040
Loss on epoch 40: 0.736718
Epoch 40 : Accuracy on training set: 0.8646666666666667
Epoch 40 : Accuracy on development set: 0.7826666666666666
Loss on epoch 41: 0.746797
Loss on epoch 42: 0.732642
Loss on epoch 43: 0.699260
Loss on epoch 44: 0.661625
Loss on epoch 45: 0.635557
Loss on epoch 46: 0.709161
Loss on epoch 47: 0.628458
Loss on epoch 48: 0.604694
Loss on epoch 49: 0.610573
Accuracy on training set: 0.8853333333333333
Accuracy on develpment set: 0.786
Loss on epoch 0: 1.586349
Epoch 0 : Accuracy on training set: 0.49666666666666665
Epoch 0 : Accuracy on development set: 0.498
Loss on epoch 1: 1.425531
Loss on epoch 2: 1.404426
Loss on epoch 3: 1.385035
Loss on epoch 4: 1.386975
Loss on epoch 5: 1.392177
Loss on epoch 6: 1.378059
Loss on epoch 7: 1.380755
Loss on epoch 8: 1.370604
Loss on epoch 9: 1.360894
Loss on epoch 10: 1.361191
```

```
Epoch 10 : Accuracy on training set: 0.6086666666666667
Epoch 10 : Accuracy on development set: 0.6293333333333333
Loss on epoch 11: 1.343627
Loss on epoch 12: 1.323903
Loss on epoch 13: 1.303303
Loss on epoch 14: 1.283770
Loss on epoch 15: 1.240168
Loss on epoch 16: 1.189801
Loss on epoch 17: 1.182481
Loss on epoch 18: 1.111729
Loss on epoch 19: 1.071498
Loss on epoch 20: 1.032303
Epoch 20 : Accuracy on training set: 0.702
Epoch 20 : Accuracy on development set: 0.6873333333333334
Loss on epoch 21: 1.091102
Loss on epoch 22: 0.986199
Loss on epoch 23: 0.957805
Loss on epoch 24: 0.971484
Loss on epoch 25: 0.956278
Loss on epoch 26: 0.904816
Loss on epoch 27: 0.876601
Loss on epoch 28: 0.891765
Loss on epoch 29: 0.810309
Loss on epoch 30: 0.798981
Epoch 30 : Accuracy on training set: 0.794
Epoch 30 : Accuracy on development set: 0.7733333333333333
Loss on epoch 31: 0.836227
Loss on epoch 32: 0.806355
Loss on epoch 33: 0.831540
Loss on epoch 34: 0.772063
Loss on epoch 35: 0.777711
Loss on epoch 36: 0.750377
Loss on epoch 37: 0.726027
Loss on epoch 38: 0.732040
Loss on epoch 39: 0.780060
Loss on epoch 40: 0.787553
Epoch 40 : Accuracy on training set: 0.8266666666666667
Epoch 40 : Accuracy on development set: 0.7766666666666666
Loss on epoch 41: 0.813430
Loss on epoch 42: 0.733097
Loss on epoch 43: 0.715621
Loss on epoch 44: 0.750937
Loss on epoch 45: 0.692192
Loss on epoch 46: 0.639561
Loss on epoch 47: 0.640958
Loss on epoch 48: 0.635280
Loss on epoch 49: 0.602479
Accuracy on training set: 0.882
Accuracy on develpment set: 0.7866666666666666
Loss on epoch 0: 1.519911
Epoch 0 : Accuracy on training set: 0.49733333333333335
Epoch 0 : Accuracy on development set: 0.52
Loss on epoch 1: 1.420963
Loss on epoch 2: 1.399583
Loss on epoch 3: 1.386322
Loss on epoch 4: 1.391595
Loss on epoch 5: 1.383292
Loss on epoch 6: 1.380847
Loss on epoch 7: 1.381919
Loss on epoch 8: 1.374650
Loss on epoch 9: 1.372366
Loss on epoch 10: 1.367901
Epoch 10 : Accuracy on training set: 0.578
Epoch 10 : Accuracy on development set: 0.6046666666666667
Loss on epoch 11: 1.364213
Loss on epoch 12: 1.350145
Loss on epoch 13: 1.328023
Loss on epoch 14: 1.288507
Loss on epoch 15: 1.247406
Loss on epoch 16: 1.215853
Loss on epoch 17: 1.190416
Loss on epoch 18: 1.157581
Loss on epoch 19: 1.144075
Loss on epoch 20: 1.102413
Epoch 20 : Accuracy on training set: 0.7553333333333333
Epoch 20 : Accuracy on development set: 0.7666666666666667
Loss on epoch 21: 1.088241
Loss on epoch 22: 1.011230
Loss on epoch 23: 1.004115
Loss on epoch 24: 0.977331
Loss on epoch 25: 0.944103
Loss on epoch 26: 0.940980
Loss on epoch 27: 0.901608
Loss on epoch 28: 0.853884
Loss on epoch 29: 0.799597
Loss on epoch 30: 0.907269
Epoch 30 : Accuracy on training set: 0.8246666666666667
Epoch 30 : Accuracy on development set: 0.7753333333333333
Loss on epoch 31: 0.840919
Loss on epoch 32: 0.829576
Loss on epoch 33: 0.827049
Loss on epoch 34: 0.781011
Loss on epoch 35: 0.770719
Loss on epoch 36: 0.792228
Loss on epoch 37: 0.770799
Loss on epoch 38: 0.775575
Loss on epoch 39: 0.774961
Loss on epoch 40: 0.729625
Epoch 40 : Accuracy on training set: 0.84
Epoch 40 : Accuracy on development set: 0.7786666666666666
Loss on epoch 41: 0.730152
Loss on epoch 42: 0.706213
Loss on epoch 43: 0.751518
Loss on epoch 44: 0.674845
Loss on epoch 45: 0.705328
Loss on epoch 46: 0.670591
Loss on epoch 47: 0.670965
Loss on epoch 48: 0.646647
Loss on epoch 49: 0.712385
Accuracy on training set: 0.844
Accuracy on develpment set: 0.7733333333333333
Loss on epoch 0: 1.438542
Epoch 0 : Accuracy on training set: 0.5446666666666666
Epoch 0 : Accuracy on development set: 0.53
Loss on epoch 1: 1.399656
Loss on epoch 2: 1.398727
Loss on epoch 3: 1.370237
Loss on epoch 4: 1.375899
Loss on epoch 5: 1.363138
Loss on epoch 6: 1.363774
Loss on epoch 7: 1.329967
Loss on epoch 8: 1.324161
Loss on epoch 9: 1.301518
Loss on epoch 10: 1.259328
Epoch 10 : Accuracy on training set: 0.68
Epoch 10 : Accuracy on development set: 0.684
Loss on epoch 11: 1.199866
Loss on epoch 12: 1.137927
Loss on epoch 13: 1.139731
Loss on epoch 14: 1.058346
Loss on epoch 15: 1.042055
Loss on epoch 16: 0.996019
Loss on epoch 17: 1.045835
Loss on epoch 18: 1.012110
Loss on epoch 19: 0.973730
Loss on epoch 20: 0.938438
Epoch 20 : Accuracy on training set: 0.7866666666666666
Epoch 20 : Accuracy on development set: 0.7673333333333333
Loss on epoch 21: 0.865304
Loss on epoch 22: 0.916384
Loss on epoch 23: 0.922813
Loss on epoch 24: 0.944782
Loss on epoch 25: 0.911733
Loss on epoch 26: 0.840329
Loss on epoch 27: 0.852562
Loss on epoch 28: 0.869850
Loss on epoch 29: 0.851738
Loss on epoch 30: 0.894680
Epoch 30 : Accuracy on training set: 0.8046666666666666
Epoch 30 : Accuracy on development set: 0.7846666666666666
Loss on epoch 31: 0.891546
Loss on epoch 32: 0.823058
Loss on epoch 33: 0.812507
Loss on epoch 34: 0.799075
Loss on epoch 35: 0.744968
Loss on epoch 36: 0.731151
Loss on epoch 37: 0.755201
Loss on epoch 38: 0.770863
Loss on epoch 39: 0.726483
Loss on epoch 40: 0.703847
Epoch 40 : Accuracy on training set: 0.8606666666666667
Epoch 40 : Accuracy on development set: 0.7933333333333333
Loss on epoch 41: 0.680851
Loss on epoch 42: 0.670606
Loss on epoch 43: 0.676946
Loss on epoch 44: 0.628461
Loss on epoch 45: 0.751701
Loss on epoch 46: 0.643794
Loss on epoch 47: 0.668844
Loss on epoch 48: 0.637758
Loss on epoch 49: 0.663146
Accuracy on training set: 0.83
Accuracy on develpment set: 0.7733333333333333
Loss on epoch 0: 1.582921
Epoch 0 : Accuracy on training set: 0.5186666666666667
Epoch 0 : Accuracy on development set: 0.49466666666666664
Loss on epoch 1: 1.402367
Loss on epoch 2: 1.429407
Loss on epoch 3: 1.384374
Loss on epoch 4: 1.388065
Loss on epoch 5: 1.390472
Loss on epoch 6: 1.376821
Loss on epoch 7: 1.381059
Loss on epoch 8: 1.380923
Loss on epoch 9: 1.365732
Loss on epoch 10: 1.369926
Epoch 10 : Accuracy on training set: 0.5773333333333334
Epoch 10 : Accuracy on development set: 0.592
Loss on epoch 11: 1.354480
Loss on epoch 12: 1.338732
Loss on epoch 13: 1.312763
Loss on epoch 14: 1.278409
Loss on epoch 15: 1.218435
Loss on epoch 16: 1.234303
Loss on epoch 17: 1.173498
Loss on epoch 18: 1.131045
Loss on epoch 19: 1.101930
Loss on epoch 20: 1.034000
Epoch 20 : Accuracy on training set: 0.772
Epoch 20 : Accuracy on development set: 0.7786666666666666
Loss on epoch 21: 1.029811
Loss on epoch 22: 0.965537
Loss on epoch 23: 0.982255
Loss on epoch 24: 0.980842
Loss on epoch 25: 0.949238
Loss on epoch 26: 0.901382
Loss on epoch 27: 0.864771
Loss on epoch 28: 0.858153
Loss on epoch 29: 0.849574
Loss on epoch 30: 0.857712
Epoch 30 : Accuracy on training set: 0.7746666666666666
Epoch 30 : Accuracy on development set: 0.756
Loss on epoch 31: 0.866490
Loss on epoch 32: 0.842638
Loss on epoch 33: 0.824803
Loss on epoch 34: 0.807273
Loss on epoch 35: 0.824110
Loss on epoch 36: 0.790786
Loss on epoch 37: 0.794980
Loss on epoch 38: 0.795153
Loss on epoch 39: 0.746498
Loss on epoch 40: 0.720798
Epoch 40 : Accuracy on training set: 0.85
Epoch 40 : Accuracy on development set: 0.7913333333333333
Loss on epoch 41: 0.709065
Loss on epoch 42: 0.711949
Loss on epoch 43: 0.645738
Loss on epoch 44: 0.797846
Loss on epoch 45: 0.765962
Loss on epoch 46: 0.751778
Loss on epoch 47: 0.711883
Loss on epoch 48: 0.719296
Loss on epoch 49: 0.695967
Accuracy on training set: 0.8153333333333334
Accuracy on develpment set: 0.7733333333333333
Loss on epoch 0: 1.425593
Epoch 0 : Accuracy on training set: 0.532
Epoch 0 : Accuracy on development set: 0.5286666666666666
Loss on epoch 1: 1.381007
Loss on epoch 2: 1.377631
Loss on epoch 3: 1.367567
Loss on epoch 4: 1.359963
Loss on epoch 5: 1.349248
Loss on epoch 6: 1.350603
Loss on epoch 7: 1.329710
Loss on epoch 8: 1.292472
Loss on epoch 9: 1.266982
Loss on epoch 10: 1.202337
Epoch 10 : Accuracy on training set: 0.6893333333333334
Epoch 10 : Accuracy on development set: 0.6926666666666667
Loss on epoch 11: 1.151494
Loss on epoch 12: 1.126657
Loss on epoch 13: 1.120005
Loss on epoch 14: 1.072054
Loss on epoch 15: 1.032179
Loss on epoch 16: 0.991009
Loss on epoch 17: 1.011917
Loss on epoch 18: 1.000584
Loss on epoch 19: 1.072506
Loss on epoch 20: 1.022312
Epoch 20 : Accuracy on training set: 0.788
Epoch 20 : Accuracy on development set: 0.768
Loss on epoch 21: 0.927067
Loss on epoch 22: 0.994714
Loss on epoch 23: 0.917446
Loss on epoch 24: 0.911151
Loss on epoch 25: 0.944886
Loss on epoch 26: 0.903168
Loss on epoch 27: 0.830446
Loss on epoch 28: 0.873118
Loss on epoch 29: 0.830702
Loss on epoch 30: 0.797134
Epoch 30 : Accuracy on training set: 0.8246666666666667
Epoch 30 : Accuracy on development set: 0.7853333333333333
Loss on epoch 31: 0.765179
Loss on epoch 32: 0.751524
Loss on epoch 33: 0.734486
Loss on epoch 34: 0.718726
Loss on epoch 35: 0.865293
Loss on epoch 36: 0.790019
Loss on epoch 37: 0.793434
Loss on epoch 38: 0.769043
Loss on epoch 39: 0.725073
Loss on epoch 40: 0.697822
Epoch 40 : Accuracy on training set: 0.8333333333333334
Epoch 40 : Accuracy on development set: 0.784
Loss on epoch 41: 0.730412
Loss on epoch 42: 0.655365
Loss on epoch 43: 0.676250
Loss on epoch 44: 0.654759
Loss on epoch 45: 0.665818
Loss on epoch 46: 0.689650
Loss on epoch 47: 0.678236
Loss on epoch 48: 0.656240
Loss on epoch 49: 0.658717
Accuracy on training set: 0.8953333333333333
Accuracy on develpment set: 0.796
Loss on epoch 0: 1.392667
Epoch 0 : Accuracy on training set: 0.516
Epoch 0 : Accuracy on development set: 0.5133333333333333
Loss on epoch 1: 1.391096
Loss on epoch 2: 1.387093
Loss on epoch 3: 1.354417
Loss on epoch 4: 1.348331
Loss on epoch 5: 1.346931
Loss on epoch 6: 1.321099
Loss on epoch 7: 1.290333
Loss on epoch 8: 1.269951
Loss on epoch 9: 1.236527
Loss on epoch 10: 1.206758
Epoch 10 : Accuracy on training set: 0.6973333333333334
Epoch 10 : Accuracy on development set: 0.698
Loss on epoch 11: 1.168650
Loss on epoch 12: 1.153437
Loss on epoch 13: 1.075049
Loss on epoch 14: 1.047565
Loss on epoch 15: 1.032062
Loss on epoch 16: 1.028239
Loss on epoch 17: 0.952992
Loss on epoch 18: 1.070398
Loss on epoch 19: 1.023586
Loss on epoch 20: 0.939880
Epoch 20 : Accuracy on training set: 0.7806666666666666
Epoch 20 : Accuracy on development set: 0.7713333333333333
Loss on epoch 21: 0.918702
Loss on epoch 22: 0.897548
Loss on epoch 23: 0.896112
Loss on epoch 24: 0.888643
Loss on epoch 25: 0.837767
Loss on epoch 26: 0.813554
Loss on epoch 27: 0.842212
Loss on epoch 28: 0.800871
Loss on epoch 29: 0.774925
Loss on epoch 30: 0.772834
Epoch 30 : Accuracy on training set: 0.838
Epoch 30 : Accuracy on development set: 0.7873333333333333
```

```
Loss on epoch 31: 0.782076
Loss on epoch 32: 0.773979
Loss on epoch 33: 0.735623
Loss on epoch 34: 0.712131
Loss on epoch 35: 0.716493
Loss on epoch 36: 0.662604
Loss on epoch 37: 0.636504
Loss on epoch 38: 0.768248
Loss on epoch 39: 0.672472
Loss on epoch 40: 0.656496
Epoch 40 : Accuracy on training set: 0.8746666666666667
Epoch 40 : Accuracy on development set: 0.792
Loss on epoch 41: 0.719605
Loss on epoch 42: 0.718974
Loss on epoch 43: 0.679723
Loss on epoch 44: 0.623542
Loss on epoch 45: 0.709218
Loss on epoch 46: 0.635072
Loss on epoch 47: 0.628064
Loss on epoch 48: 0.627685
Loss on epoch 49: 0.557566
Accuracy on training set: 0.882
Accuracy on develpment set: 0.7886666666666666
Loss on epoch 0: 1.414679
Epoch 0 : Accuracy on training set: 0.5286666666666666
Epoch 0 : Accuracy on development set: 0.5073333333333333
Loss on epoch 1: 1.392639
Loss on epoch 2: 1.372767
Loss on epoch 3: 1.375938
Loss on epoch 4: 1.353339
Loss on epoch 5: 1.365352
Loss on epoch 6: 1.333591
Loss on epoch 7: 1.322835
Loss on epoch 8: 1.279954
Loss on epoch 9: 1.264452
Loss on epoch 10: 1.199511
Epoch 10 : Accuracy on training set: 0.6886666666666666
Epoch 10 : Accuracy on development set: 0.6973333333333334
Loss on epoch 11: 1.163608
Loss on epoch 12: 1.137069
Loss on epoch 13: 1.147114
Loss on epoch 14: 1.066821
Loss on epoch 15: 1.049882
Loss on epoch 16: 1.019609
Loss on epoch 17: 0.950207
Loss on epoch 18: 0.935612
Loss on epoch 19: 0.975455
Loss on epoch 20: 0.926123
```

```
Epoch 20 : Accuracy on training set: 0.8113333333333334
Epoch 20 : Accuracy on development set: 0.788
Loss on epoch 21: 0.922995
Loss on epoch 22: 0.893760
Loss on epoch 23: 0.925652
Loss on epoch 24: 0.900946
Loss on epoch 25: 0.861356
Loss on epoch 26: 0.782589
Loss on epoch 27: 0.764770
Loss on epoch 28: 0.757702
Loss on epoch 29: 0.854769
Loss on epoch 30: 0.775857
Epoch 30 : Accuracy on training set: 0.8586666666666667
Epoch 30 : Accuracy on development set: 0.7946666666666666
Loss on epoch 31: 0.785042
Loss on epoch 32: 0.792095
Loss on epoch 33: 0.693168
Loss on epoch 34: 0.673764
Loss on epoch 35: 0.688746
Loss on epoch 36: 0.953162
Loss on epoch 37: 0.751155
Loss on epoch 38: 0.829105
Loss on epoch 39: 0.780908
Loss on epoch 40: 0.722027
Epoch 40 : Accuracy on training set: 0.8366666666666667
Epoch 40 : Accuracy on development set: 0.7753333333333333
Loss on epoch 41: 0.723650
Loss on epoch 42: 0.674860
Loss on epoch 43: 0.695032
Loss on epoch 44: 0.622490
Loss on epoch 45: 0.627411
Loss on epoch 46: 0.616823
Loss on epoch 47: 0.648393
Loss on epoch 48: 0.614189
Loss on epoch 49: 0.611977
Accuracy on training set: 0.838
Accuracy on develpment set: 0.774
Loss on epoch 0: 1.422986
Epoch 0 : Accuracy on training set: 0.5
Epoch 0 : Accuracy on development set: 0.5213333333333333
Loss on epoch 1: 1.408642
Loss on epoch 2: 1.383976
Loss on epoch 3: 1.388192
Loss on epoch 4: 1.371315
Loss on epoch 5: 1.372909
Loss on epoch 6: 1.351616
Loss on epoch 7: 1.348557
Loss on epoch 8: 1.317351
Loss on epoch 9: 1.305502
Loss on epoch 10: 1.286543
Epoch 10 : Accuracy on training set: 0.6766666666666666
Epoch 10 : Accuracy on development set: 0.6953333333333334
Loss on epoch 11: 1.260059
Loss on epoch 12: 1.218374
Loss on epoch 13: 1.168785
Loss on epoch 14: 1.160314
Loss on epoch 15: 1.106963
Loss on epoch 16: 1.040474
Loss on epoch 17: 1.020485
Loss on epoch 18: 1.022103
Loss on epoch 19: 0.983415
Loss on epoch 20: 0.965146
Epoch 20 : Accuracy on training set: 0.7653333333333333
Epoch 20 : Accuracy on development set: 0.7413333333333333
Loss on epoch 21: 0.959964
Loss on epoch 22: 0.889029
Loss on epoch 23: 0.860837
Loss on epoch 24: 0.884458
Loss on epoch 25: 0.898918
Loss on epoch 26: 0.867009
Loss on epoch 27: 0.835637
Loss on epoch 28: 0.816926
Loss on epoch 29: 0.806164
Loss on epoch 30: 0.786363
Epoch 30 : Accuracy on training set: 0.814
Epoch 30 : Accuracy on development set: 0.7813333333333333
Loss on epoch 31: 0.774328
Loss on epoch 32: 0.758040
Loss on epoch 33: 0.795393
Loss on epoch 34: 0.731830
Loss on epoch 35: 0.769793
Loss on epoch 36: 0.720051
Loss on epoch 37: 0.706279
Loss on epoch 38: 0.694843
Loss on epoch 39: 0.749894
Loss on epoch 40: 0.681242
Epoch 40 : Accuracy on training set: 0.854
Epoch 40 : Accuracy on development set: 0.7846666666666666
Loss on epoch 41: 0.657692
Loss on epoch 42: 0.648487
Loss on epoch 43: 0.622939
Loss on epoch 44: 0.587902
Loss on epoch 45: 0.945862
Loss on epoch 46: 0.758731
Loss on epoch 47: 0.705344
Loss on epoch 48: 0.770833
Loss on epoch 49: 0.690589
Accuracy on training set: 0.8153333333333334
Accuracy on develpment set: 0.7693333333333333
Loss on epoch 0: 1.483990
Epoch 0 : Accuracy on training set: 0.5213333333333333
Epoch 0 : Accuracy on development set: 0.5493333333333333
Loss on epoch 1: 1.429934
Loss on epoch 2: 1.391415
Loss on epoch 3: 1.377372
Loss on epoch 4: 1.378537
Loss on epoch 5: 1.368426
Loss on epoch 6: 1.367807
Loss on epoch 7: 1.364103
Loss on epoch 8: 1.355358
Loss on epoch 9: 1.340760
Loss on epoch 10: 1.324147
Epoch 10 : Accuracy on training set: 0.66
Epoch 10 : Accuracy on development set: 0.654
Loss on epoch 11: 1.281471
Loss on epoch 12: 1.251351
Loss on epoch 13: 1.209924
Loss on epoch 14: 1.168602
Loss on epoch 15: 1.160807
Loss on epoch 16: 1.128679
Loss on epoch 17: 1.074380
Loss on epoch 18: 1.061683
Loss on epoch 19: 1.047371
Loss on epoch 20: 1.010857
Epoch 20 : Accuracy on training set: 0.774
Epoch 20 : Accuracy on development set: 0.7526666666666667
Loss on epoch 21: 0.932146
Loss on epoch 22: 0.870960
Loss on epoch 23: 0.950069
Loss on epoch 24: 0.900991
Loss on epoch 25: 0.926304
Loss on epoch 26: 0.904729
Loss on epoch 27: 0.880145
Loss on epoch 28: 0.849986
Loss on epoch 29: 0.823490
Loss on epoch 30: 0.791671
Epoch 30 : Accuracy on training set: 0.78
Epoch 30 : Accuracy on development set: 0.752
Loss on epoch 31: 0.839884
Loss on epoch 32: 0.839993
Loss on epoch 33: 0.805376
Loss on epoch 34: 0.805877
Loss on epoch 35: 0.747359
Loss on epoch 36: 0.726494
Loss on epoch 37: 0.713395
Loss on epoch 38: 0.699241
Loss on epoch 39: 0.756584
Loss on epoch 40: 0.732845
Epoch 40 : Accuracy on training set: 0.8686666666666667
Epoch 40 : Accuracy on development set: 0.788
Loss on epoch 41: 0.699022
Loss on epoch 42: 0.650944
Loss on epoch 43: 0.659776
Loss on epoch 44: 0.664921
Loss on epoch 45: 0.692557
Loss on epoch 46: 0.651573
Loss on epoch 47: 0.646979
Loss on epoch 48: 0.642300
Loss on epoch 49: 0.644762
Accuracy on training set: 0.806
Accuracy on develpment set: 0.7493333333333333
Average accuracy on test set across all 10 evaluations: 0.7605333333333334
Standard deviation: 0.013524627741847668
Accuracies by seed:  [0.7733333333333333, 0.7666666666666667,
0.7533333333333333, 0.766, 0.7546666666666667, 0.77, 0.7786666666666666, 0.76,
0.7546666666666667, 0.728]
```

[19]:
```python
random.seed(6)
np.random.seed(6)
torch.manual_seed(6)

## YOUR CODE HERE
hp = HyperParams(lstm_size = 50, # hidden units in lstm
                 hidden_size = 50, # hidden size of fully-connected layer
                 lstm_layers = 1, # layers in lstm
                 drop_out = 0.5, # dropout rate
                 num_epochs = 50, # number of epochs for SGD-based procedure
                 batch_size = 1024, # examples in a minibatch
                 seq_max_len = 60) # maximum length of an example sequence
use_average = True
bidirectional = True

# Train RNN model.
model4 = train_model(hp, train_exs, dev_exs, test_exs, word_vectors,␣
 ↪use_average, bidirectional)

# Generate RNN model predictions for test set.
embeddings_vec = np.array(word_vectors.vectors).astype(float)
test_exs_predicted = eval_model(model4, test_exs, embeddings_vec, hp.
 ↪seq_max_len, pred_only = True)
```

```
accuracy = eval_model(model4, test_exs, embeddings_vec, hp.seq_max_len,␣
 ↪pred_only = False)

print("Predictions saved. Accuracy on test data: ", accuracy)
```

```
Loss on epoch 0: 1.392667
Epoch 0 : Accuracy on training set: 0.516
Epoch 0 : Accuracy on development set: 0.5133333333333333
Loss on epoch 1: 1.391096
Loss on epoch 2: 1.387093
Loss on epoch 3: 1.354417
Loss on epoch 4: 1.348331
Loss on epoch 5: 1.346931
Loss on epoch 6: 1.321099
Loss on epoch 7: 1.290333
Loss on epoch 8: 1.269951
Loss on epoch 9: 1.236527
Loss on epoch 10: 1.206758
Epoch 10 : Accuracy on training set: 0.6973333333333334
Epoch 10 : Accuracy on development set: 0.698
Loss on epoch 11: 1.168650
Loss on epoch 12: 1.153437
Loss on epoch 13: 1.075049
Loss on epoch 14: 1.047565
Loss on epoch 15: 1.032062
Loss on epoch 16: 1.028239
Loss on epoch 17: 0.952992
Loss on epoch 18: 1.070398
Loss on epoch 19: 1.023586
Loss on epoch 20: 0.939880
Epoch 20 : Accuracy on training set: 0.7806666666666666
Epoch 20 : Accuracy on development set: 0.7713333333333333
Loss on epoch 21: 0.918702
Loss on epoch 22: 0.897548
Loss on epoch 23: 0.896112
Loss on epoch 24: 0.888643
Loss on epoch 25: 0.837767
Loss on epoch 26: 0.813554
Loss on epoch 27: 0.842212
Loss on epoch 28: 0.800871
Loss on epoch 29: 0.774925
Loss on epoch 30: 0.772834
Epoch 30 : Accuracy on training set: 0.838
Epoch 30 : Accuracy on development set: 0.7873333333333333
Loss on epoch 31: 0.782076
Loss on epoch 32: 0.773979
Loss on epoch 33: 0.735623
Loss on epoch 34: 0.712131
Loss on epoch 35: 0.716493
Loss on epoch 36: 0.662604
Loss on epoch 37: 0.636504
Loss on epoch 38: 0.768248
Loss on epoch 39: 0.672472
Loss on epoch 40: 0.656496
Epoch 40 : Accuracy on training set: 0.8746666666666667
Epoch 40 : Accuracy on development set: 0.792
Loss on epoch 41: 0.719605
Loss on epoch 42: 0.718974
Loss on epoch 43: 0.679723
Loss on epoch 44: 0.623542
Loss on epoch 45: 0.709218
Loss on epoch 46: 0.635072
Loss on epoch 47: 0.628064
Loss on epoch 48: 0.627685
Loss on epoch 49: 0.557566
Accuracy on training set: 0.882
Accuracy on develpment set: 0.7886666666666666
Predictions saved. Accuracy on test data:  0.7786666666666666
```

6.2.1 Error Analysis

[20]:
```
testX = []
testY = []
with open('data/imdb/test.txt', mode = 'r', encoding = 'utf-8') as file:
    for line in file:
        [label, text] = line.rstrip().split(' ', maxsplit = 1)
        testX.append(text)
        testY.append(label)
```

[21]:
```
for idx, ex in enumerate(testY):
  if ex == 'pos':
    testY[idx] = 1.0
  if ex == 'neg':
    testY[idx] = 0.0
```

[22]:
```
count = 0
for i in range(len(testY)):
    if testY[i] != test_exs_predicted[i] and count < 10:
        print("Test case ", i, " with class ", testY[i], " was incorrectly␣
 ↪classified as ", test_exs_predicted[i])
        count += 1
```

```
Test case  0  with class  1.0  was incorrectly classified as  0.0
Test case  2  with class  posI  was incorrectly classified as  1.0
Test case  14  with class  1.0  was incorrectly classified as  0.0
Test case  18  with class  1.0  was incorrectly classified as  0.0
Test case  24  with class  1.0  was incorrectly classified as  0.0
Test case  25  with class  1.0  was incorrectly classified as  0.0
Test case  26  with class  1.0  was incorrectly classified as  0.0
Test case  29  with class  1.0  was incorrectly classified as  0.0
Test case  33  with class  1.0  was incorrectly classified as  0.0
Test case  36  with class  1.0  was incorrectly classified as  0.0
```

[23]:
```
print(testX[14])
print("True label: ", testY[14])
print("Predicted label: ", test_exs_predicted[14])
```

```
Shot entirely on location in Bulgaria, The Man With The Screaming Brain is a
hilarious love story between two rich ugly-American types and a murderous hotel
maid gypsy. <br /><br />William Cole and his wife Jackie arrive in Bulgaria on a
business trip and catch a cab driven by hustler Yegor. Things start to go awry
when Tatoya, the maid, murders Yegor and William and a mad scientist implants a
piece of Yegor's brain in William's head. Robots eventually become involved, as
do gypsies with broken fingers, head injuries, Bruce Campbell riding a pink
Vespa with prissy little streamers, and All-Of-Me-style physical comedy by a
character at war with a voice in his brain who controls half of his body.<br
/><br />The Man With The Screaming Brain is an incredibly funny film. It has the
most hilarious tracking shot I have ever seen (when Bruce Campbell's character,
fresh from the lab and complete with giant forehead scar and blue hospital
pajamas, runs into a square and scares a crowd of people) and a falling-down-
the-steps murder scene that had the entire test screening audience screaming
laughing. The whole thing is a damn riot from beginning to end and I would
recommend it to any fan of physical comedy, Bruce Campbell, or B-movies in
general.
True label:  1.0
Predicted label:  0.0
```

[24]:
```
print(testX[24])
print("True label: ", testY[24])
print("Predicted label: ", test_exs_predicted[24])
```

```
I really enjoyed "Random Hearts". It was shocking to see such a low rating on
IMDB, but chacun a son gout and all that. I am a big fan of Harrison Ford, but I
do have to admit that he was ill cast in this movie, and the reason I gave it 9
instead of 10. Kristin Scott Thomas, though, was just wonderful. She was
believable and beautiful, in spite of being made up for half the movie like
she'd been crying for days. Could "Random Hearts" have been better? Sure, but it
is very much worth seeing as it is.
True label:  1.0
Predicted label:  0.0
```

6.3 Analysis

Include an analysis of the results that you obtained in the experiments above. Also compare with the sentiment classification performance from previous assignments and explain the difference in accuracy. Show the results using table(s).

6.3.1 Experimental Evaluations on the Rotten Tomatoes Dataset

Models	Model 1	Model 2	Model 3	Model 4
Train Accuracy	85.34%	82.42%	91.31%	83.99%
Dev Accuracy	76.73%	76.73%	75.23%	78.04%

Model 1 This model was made with one LSTM + a fully connected network, using the last hidden state of LSTM. Model 1 was not the best performing model of all the models that were used. The development accuracy of 76.73% is not a bad score, though there is likely a model that could do much better. It seems that the performance was about the same as Model 2, though there was a higher training accuracy for Model 1.

Model 2 This model was made with one LSTM + a fully connected network, using average of all states of the LSTM. Model 2 was not the best performing model of all the models that were used. The development accuracy of 76.73% is not a bad score, though there is likely a model that could do much better. It seems that the performance was about the same as Model 1, though there was a higher training accuracy for Model 1.

Model 3 This model was made with two LSTMs (bidirectional) + a fully connected network, concatenating their last states. Model 3 was not the best performing model of all the models that were used. The development accuracy of 75.23% was the lowest of all training accuracies recorded. However, the training accuracy of 91.31% was the highest training accrurancy recorded among all four models.

Model 4 This model was made with two LSTMs (bidirectional) + a fully connected network, concatenating the averages of their states. Model 4 was the best performing model of all the models that were used. The training accuracy of 83.99% was not better than the other models. However, the development accuracy of 78.04% was better than all of the other models by over 1%.

It would be interesting to see how these models may compete with different datasets or with different initializations. Perhaps one model was at an advantage over another because of the starting point at which the weights were set before training. We will observe this in the analysis for the following experiments.

6.3.2 Results of Randomized Initializations (different seeds)

Models	Model 1	Model 2	Model 3	Model 4
Average Accuracy	83.83%	85.16%	81.13%	90.02%
Standard Deviation	0.0319	0.0458	0.0379	0.0410

Model 1 On average, this model performed the second worst out of all of the models. The average accuracy was based on testing accuracy, so it is more extensive than the Model 1 results discussed in the previous section of analysis which ranked this model as the second-best model (in a tie with Model 2). This model also had the lowest standard deviation of all the models. So, it seems that this model will result in the second worst results regardless of model initialization, though some small improvements could be seen in some cases of initialization.

These results, in combination with the Model 2 results below, seem to support the results seen in the previous section of analysis. Although, here we get some clarification as to which model is better out of the two tied models in the previous section of analysis.

Model 2 On average, this model performed the second best out of all of the models. The average accuracy was based on testing accuracy, so it is more extensive than the Model 2 results discussed in the previous section of analysis which ranked this model as the second-best model (in a tie with Model 1). This model also had the highest standard deviation of all the models, which means that this model has the most variable results and depends more heavily on the model initialization than the rest of the models, though the accuracy will still not change much in any case.

These results, in combination with the Model 1 results above, seem to support the results seen in the previous section of analysis. Although, here we get some clarification as to which model is better out of the two tied models in the previous section of analysis.

Model 3 On average, this model performed the worst out of all of the models. The average accuracy was based on testing accuracy, so it is more extensive than the Model 3 results discussed in the previous section of analysis, though this section also ranked this model as the worst performing model. This model had a fairly average standard deviation in comparison with the other models. So, it seems that this model will likely result in the worst results regardless of model initialization, though some small improvements could be seen in some cases of initialization.

These results seem to support the Model 3 results seen in the previous section of analysis.

Model 4 On average, this model performed the best out of all of the models. The average accuracy was based on testing accuracy, so it is more extensive than the Model 4 results discussed in the previous section of analysis, though this section also ranked this model as the best performing model. This model had a slightly larger standard deviation in comparison with the other models, though not the highest. So, it seems that the model initialization does have an impact on model performance, though not a large one.

These results seem to support the Model 4 results seen in the previous section of analysis.

6.3.3 Experimental Evaluations on the IMDB dataset

Models	Model 1	Model 2	Model 3	Model 4
Train Accuracy	83.13%	79.53%	95.67%	88.20%
Dev Accuracy	63.13%	71.67%	68.46%	78.66%

Model 1 This model performed the worst out of all of the models when using the IMDB dataset. With a development accuracy of 63.13%, this model performed around 5% worse than the next-best

model. This is a significant difference and is much larger than the differences seen between the models when using the Rotten Tomatoes dataset. However, this model was consistently ranked as one of the worst models across all of the experiments, so these results seem consistent. One explanation for the decrease in performance with this dataset could be the drastic decrease in data samples that are provided in the IMDB dataset. The less data the model has to train on, the worse its performance will be.

Model 2 This model performed the second best out of all of the models when using the IMDB dataset. With a development accuracy of 71.67%, this model surpasses the performance of Models 1 and 3, which is consistent with the results seen in previous experiments. However, the performance on this IMDB dataset is worse than what was achieved when using the Rotten Tomatoes dataset. One explanation for the decrease in performance could be the drastic decrease in data samples that are provided in the IMDB dataset in comparison to the Rotten Tomatoes dataset. The less data the model has to train on, the worse its performance will be.

Model 3 This model performed the second worst out of all of the models when using the IMDB dataset. With a development accuracy of 68.46%, this model performed better than Model 1, which is a change in the results from previous experiments. It could be that this model simply works better for this specific dataset. However, this performance is still significantly worse than the performance achieved when using the Rotten Tomatoes dataset. One explanation for the decrease in performance could be the drastic decrease in data samples that are provided in the IMDB dataset in comparison to the Rotten Tomatoes dataset. The less data the model has to train on, the worse its performance will be.

Model 4 This model performed the best out of all of the models when using the IMDB dataset. With a development accuracy of 78.66%, this model performed around 7% better than the next-best model. This is a significant difference in model performance and supports the idea that this model is the best for this sentiment classification task, as seen in the previous sections of analysis. In fact, this model performed better on the IMDB dataset than it did on the Rotten Tomatoes dataset, which is surprising considering the drastic decrease in the amount of data samples. It could be that this model simply works slightly better for this specific dataset.

All of the results discussed in this section seem to remain fairly consistent with the results seen in the previous sections of analysis.

6.3.4 Cross-Domain Performance Evaluation

Training Data	IMDB	Rotten Tomatoes
Train Accuracy	88.20%	83.39%
Dev Accuracy	78.66%	65.86%
Test Accuracy	76.67%	63.46%

Now that we know which model performs the best for this task, we can investigate how much of an impact the training data has on the model. As mentioned in the previous section of evaluation, more data usually ensures better performance. However, especially in the case of NLP, this might not always be true. It is also helpful for the training data to be consistent with and similar to the development and testing data. This is seen in the results table above.

When we train our model on the Rotten Tomatoes dataset before testing it on the IMDB dataset, the performance is drastically worse. A difference of over 10% is significant, and is likely due to the difference between the Rotten Tomatoes data samples and the IMDB data samples. So, although the Rotten Tomatoes dataset has a drastically larger amount of data, this does not necessarily ensure better performance since the IMDB data will have different characteristics.

When we train our model on the IMDB dataset before testing it on the IMDB dataset, the performance is drastically better. An improvement of 10% is significant, and is likely due to the similarity between the training and testing data which are both within the IMDB dataset. Although the model is seeing less training samples here, the training samples will have more characteristics that are consistent with the testing samples than if we had trained on the Rotten Tomatoes data. In other words, the model has learned more about the characteristics that it is likely to see in the testing data, rather than chracteristics that are not likely to appear in the testing data.

So, the takeaway from this section of the analysis is that although I had previously said "the less data the model has to train on, the worse its performance will be," this statement was made with the assumption that the training and testing data are quite similar. Inconsistencies between the training and testing data will have a significant impact on model performance.

6.3.5 Comparison with Sentiment Classification in Previous Assignments

Assignment	Homework 4	Homework 5	Homework 8
Model	Naive Bayes (Binary)	Logistic Regression	RNN
Test Accuracy	80.7%	82.2%	77.86%

6.3.6 Homework 4

In previous homework assignments we have used the IMDB data to train and evaluate the performance of several machine learning models. More specifically, in Homework 4, we found that the binary multinomial bayes model performed the best out of several implementations of the Naive Bayes model. In this homework assignment, we used feature engineering to attempt to improve the performance of the Multinomial Naive Bayes model. However, despite all of our efforts to provide more features and different combinations of features for this model, the Binary model outperformed. The final accuracy that was achived with the binary multinomial bayes model was 80.7%. This is not a bad score, especially for such a simple model, but as we progressed through the homework assignments, we found more intricate models that proved to have better performance.

6.3.7 Homework 5

In Homework 5, we investigated the logistic regression model. Again, we used feature engineering in an attempt to find the best combination of features to achieve the greatest accuracy score. In the end, the logistic regression model that used the features lexicon_features_negation, word_features, lexicon_features, len_feature, and deictic_feature, resulted in the best performance. This model achieved an accuracy score of 82.2% which is higher than what was achieved in the previous homework assignment (Homework 4). This is likely due to the increased intricacy of the model that is able to find a decision boundary to classify the datapoints rather than calculating probabilities for every word in the vocabulary.

6.3.8 Homework 8

In this homework assignment, we investigated different RNN models. These models are much more sophisticated than the models used in previous assignments. The recurrent nature of the neural network is meant to improve performance by incrimentally increasing the data input to the model. However, during the testing process, we originally used a seed of 1 for our best model (Model 4) and only achieved a test accuracy of 76.67%, which is lower than the models used in both Homeworks 4 and 5. In an attempt to further improve on this accuracy, in the bonus section, a set of 10 different seeds were used to see which seed produced the best results. Even with the best seed of 6, this model only achieved a test accuracy of 77.86%, which is still lower than both the Naive Bayes and Logistic Regression model scores. This is somewhat surpising.

It is important to note that when training and testing with the Rotten Tomatoes dataset, which had a drastically larger amount of data samples, an average accuracy of 90.02% was achieved with Model 4. This could indicate that given a larger amount of data in the case of using the IMDB dataset, the performance of Model 4 could have greatly improved. This more sophisticated model may simply require a larger amount of data to produce more accurate predictions. It would be interesting to investigate this further provided the access to more IMDB data. In theory, the increased amount of data would allow the RNN model (Model 4) to surpass the accruacy score of the models used in Homeworks 4 and 5.

Additionally, in the bonus section, error analysis was performed to observe a few cases where this model made incorrect predictions. In the first case that is shown, test case 14, Model 4 incorrectly assigns a negative label to a data sample with a positive sentiment. The same situation can be seen in test case 24, which is shown immediately after. Reading the contents of these data samples, it seems obvious that these models should be assigned a positive label. It is not clear why the RNN model has incorrectly classified these samples. Again, this indicates that an increased amount of data samples could provide the model with a better chance of learning the correct characteristics for positive and negative labels.