Introduction to NLP

Assignment-2

Submitted by - Ishit Bansal (2021101083)

Report

Feed Forward Neural Network POS Tagging

Hyperparameter Tuning

- Configuration 1

  embedding_dim = 64
  hidden_dim = 128
  hidden_layers = 5
  p = 2
  s = 3
  activation = nn.Tanh()

```python
train_dataset = POSDatatset_FFNN(train_filepath, p, s)
training_args = {'vocab': train_dataset.vocab, 'tags': train_dataset.tags,
                 'words_index': train_dataset.words_index, 'tags_index': train_dataset.tags_index}
dev_dataset = POSDatatset_FFNN(dev_filepath, p, s, training_args)

input_dim = len(train_dataset.vocab)
output_dim = len(train_dataset.tags)
embedding_dim = 64
hidden_dim = 128
hidden_layers = 5
p = 2
s = 3
activation = nn.Tanh()

model1 = FFNN_Tagger(input_dim, embedding_dim, hidden_dim,
                     output_dim, p, s, hidden_layers, activation)
tagger = POS_Tagger()
tagger.train(train_dataset, model=model1)

print(tagger.model)
accuracy, recall, f1_micro, f1_macro, confusion_mat = tagger.evaluate(
    dev_dataset)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("F1 micro:", f1_micro)
print("F1 macro:", f1_macro)
print('Confusion matrix', confusion_mat)

table.add_row([hidden_layers, hidden_dim, embedding_dim, activation, accuracy])
```
✓ 26.7s Python

Dev set evaluation metrics :

```
FFNN_Tagger(
  (embedding): Embedding(515, 64)
  (activation): Tanh()
  (ffnn): Sequential(
    (linear1): Linear(in_features=384, out_features=128, bias=True)
    (act1): Tanh()
    (linear2): Linear(in_features=128, out_features=128, bias=True)
    (act2): Tanh()
    (linear3): Linear(in_features=128, out_features=128, bias=True)
    (act3): Tanh()
    (linear4): Linear(in_features=128, out_features=128, bias=True)
    (act4): Tanh()
    (linear5): Linear(in_features=128, out_features=13, bias=True)
  )
)
Accuracy: 0.9739536284251732
Recall: 0.9739536284251732
F1 micro: 0.9739536284251732
F1 macro: 0.9540229766524841
Confusion matrix [[ 195    0   12    1    0    1    0   14    2    0    0    2    0]
 [   0 1405    0    0    0    3    0    1    0    6    0    0    0]
 [   4    0   48    1    0    0    0    3    0    0    0    2    1]
 [   0    0    0  251    0    0    0    2    0    0    0    0   13]
 [   0    0    0    0  107    0    0    0    0    0    0    0    0]
 [   0   18    0    0    0  544    0    0    1    0    5    0    0]
 [   0    0    0    0    0    0   35    0    0    0    0    0    0]
 [   2    0    2    0    0    0    0 1125    0    0    0   13    1]
 [   1    0    0    0    0    0    0    1  129    0    0    0    0]
 [   0    6    0    0    0    0    0    0    0   67    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0  413    0    1]
 [   1    1    1    0    1    0    0    5    2    0    0 1537    3]
 [   1   30    0    0    0    0    0    6    3    0    0    0  613]]
```

- Configuration 2

embedding_dim = 256
hidden_dim = 64
hidden_layers = 3
p = 2
s = 3
activation = nn.LeakyReLU()

```python
train_dataset = POSDatatset_FFNN(train_filepath, p, s)
training_args = {'vocab': train_dataset.vocab, 'tags': train_dataset.tags,
                 'words_index': train_dataset.words_index, 'tags_index': train_dataset.tags_index}
dev_dataset = POSDatatset_FFNN(dev_filepath, p, s, training_args)

input_dim = len(train_dataset.vocab)
output_dim = len(train_dataset.tags)
embedding_dim=256
hidden_dim=64
hidden_layers=3
p=2
s=3
activation=nn.LeakyReLU()

model2=FFNN_Tagger(input_dim,embedding_dim,hidden_dim,output_dim,p,s,hidden_layers,activation)
tagger = POS_Tagger()
tagger.train(train_dataset,model=model2)

print(tagger.model)
accuracy,recall,f1_micro,f1_macro,confusion_mat=tagger.evaluate(dev_dataset)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("F1 micro:", f1_micro)
print("F1 macro:", f1_macro)
print('Confusion matrix', confusion_mat)

table.add_row([hidden_layers,hidden_dim,embedding_dim,activation,accuracy])
```

✓ 25.6s Python

Dev set evaluation metrics :

```
FFNN_Tagger(
  (embedding): Embedding(515, 256)
  (activation): LeakyReLU(negative_slope=0.01)
  (ffnn): Sequential(
    (linear1): Linear(in_features=1536, out_features=64, bias=True)
    (act1): LeakyReLU(negative_slope=0.01)
    (linear2): Linear(in_features=64, out_features=64, bias=True)
    (act2): LeakyReLU(negative_slope=0.01)
    (linear3): Linear(in_features=64, out_features=13, bias=True)
  )
)
Accuracy: 0.9753086419753086
Recall: 0.9753086419753086
F1 micro: 0.9753086419753086
F1 macro: 0.9533101882735436
Confusion matrix [[ 196    0   12    0    0    0    0   15    1    0    0    2    1]
 [    0 1409    1    0    0    0    0    0    0    5    0    0    0]
 [    3    1   51    1    0    0    0    2    0    0    0    0    1]
 [    0    1    0  251    0    0    0    0    0    0    0    0   14]
 [    0    0    0    0  106    1    0    0    0    0    0    0    0]
 [    0   17    0    0    0  542    0    1    0    0    8    0    0]
 [    0    0    0    0    0    0   35    0    0    0    0    0    0]
 [    1    1    0    0    0    0    1 1123    1    0    0   10    6]
 [    0    0    1    0    0    0    0    1  129    0    0    0    0]
 [    0   12    0    0    0    0    0    0    0   61    0    0    0]
 [    0    0    0    0    0    1    0    0    0    0  412    0    1]
 [    2    0    1    0    0    0    0    2    0    0    0 1544    2]
 [    8   20    0    0    0    0    0    5    1    0    0    0  619]]
```

- Configuration 3

  embedding_dim = 256
  hidden_dim = 128
  hidden_layers = 4
  p = 2
  s = 3
  activation = nn.ReLU()

```python
train_dataset = POSDatatset_FFNN(train_filepath, p, s)
training_args = {'vocab': train_dataset.vocab, 'tags': train_dataset.tags,
                 'words_index': train_dataset.words_index, 'tags_index': train_dataset.tags_index}
dev_dataset = POSDatatset_FFNN(dev_filepath, p, s, training_args)

input_dim = len(train_dataset.vocab)
output_dim = len(train_dataset.tags)
embedding_dim=256
hidden_dim=128
hidden_layers=4
p=2
s=3
activation=nn.ReLU()

model3=FFNN_Tagger(input_dim,embedding_dim,hidden_dim,output_dim,p,s,hidden_layers,activation)
tagger = POS_Tagger()
tagger.train(train_dataset,model=model3)

print(tagger.model)
accuracy,recall,f1_micro,f1_macro,confusion_mat=tagger.evaluate(dev_dataset)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("F1 micro:", f1_micro)
print("F1 macro:", f1_macro)
print('Confusion matrix', confusion_mat)

table.add_row([hidden_layers,hidden_dim,embedding_dim,activation,accuracy])
```

✓ 40.6s Python

Dev set evaluation metrics :

```
FFNN_Tagger(
  (embedding): Embedding(515, 256)
  (activation): ReLU()
  (ffnn): Sequential(
    (linear1): Linear(in_features=1536, out_features=128, bias=True)
    (act1): ReLU()
    (linear2): Linear(in_features=128, out_features=128, bias=True)
    (act2): ReLU()
    (linear3): Linear(in_features=128, out_features=128, bias=True)
    (act3): ReLU()
    (linear4): Linear(in_features=128, out_features=13, bias=True)
  )
)
Accuracy: 0.9793736826257151
Recall: 0.9793736826257151
F1 micro: 0.9793736826257152
F1 macro: 0.9561330825004544
Confusion matrix [[ 203    0    9    0    1    0    1   11    0    0    0    1    1]
 [   0 1406    0    0    0    0    0    0    1    6    2    0    0]
 [   5    0   48    1    0    0    0    2    2    0    1    0    0]
 [   0    1    0  259    0    0    0    0    0    0    1    0    5]
 [   0    3    0    0  104    0    0    0    0    0    0    0    0]
 [   0   15    0    1    0  539    0    0    1    0   12    0    0]
 [   0    0    0    0    0    0   34    0    0    0    0    0    1]
 [   0    0    0    0    0    0    0 1130    1    0    0   10    2]
 [   0    0    0    0    0    0    0    1  126    0    0    2    2]
 [   0    6    0    0    0    0    0    0    0   67    0    0    0]
 [   0    0    0    0    0    1    0    0    0    0  412    0    1]
 [   0    0    2    0    0    0    0    4    0    0    0 1543    2]
 [   0    4    1    8    0    0    0    5    0    0    1    0  634]]
```

Combining accuracies of all 3 configurations:

hidden layers	hidden dim	embedding size	activation	accuracy
5	128	64	Tanh()	0.9739536284251732
3	64	256	LeakyReLU(negative_slope=0.01)	0.9753086419753086
4	128	256	ReLU()	0.9793736826257151

We observe that the best configuration is Configuration 3 with hyperparameters:
embedding_dim = 256
hidden_dim = 128
hidden_layers = 4
p = 2
s = 3
activation = nn.ReLU()
epochs = 10
learning rate = 0.001
batch size=32

Testing on Best Configuration

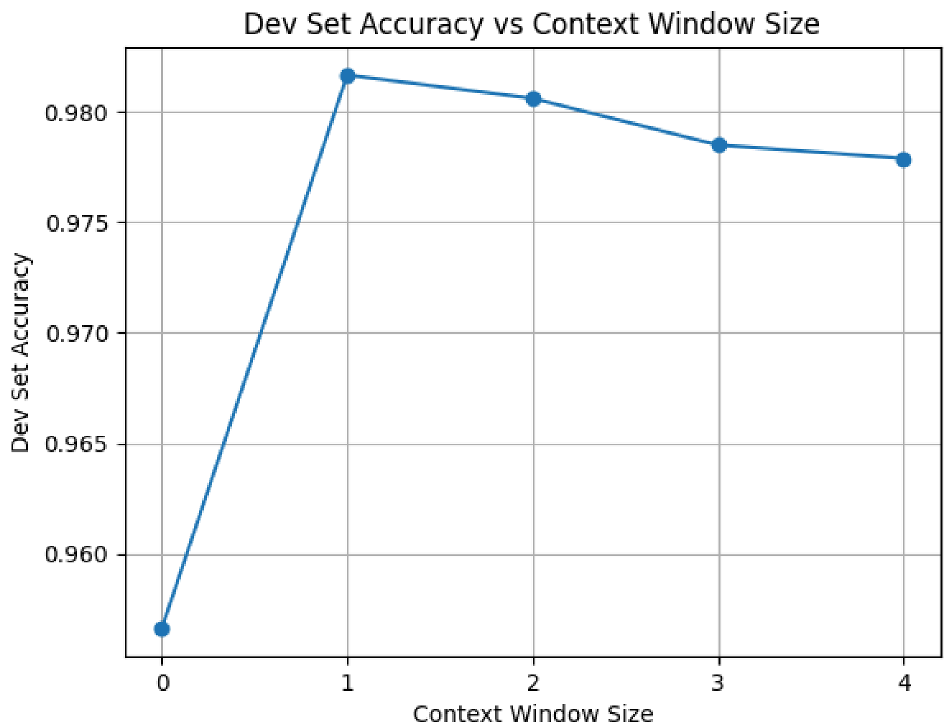
```python
train_dataset = POSDatatset_FFNN(train_filepath, p, s)
training_args = {'vocab': train_dataset.vocab, 'tags': train_dataset.tags,
                 'words_index': train_dataset.words_index, 'tags_index': train_dataset.tags_index}
test_dataset = POSDatatset_FFNN(test_filepath, p, s, training_args)

input_dim = len(train_dataset.vocab)
output_dim = len(train_dataset.tags)
embedding_dim = 256
hidden_dim = 128
hidden_layers = 4
p = 2
s = 3
activation = nn.ReLU()

best_model = FFNN_Tagger(input_dim, embedding_dim, hidden_dim,
                         output_dim, p, s, hidden_layers, activation)
tagger = POS_Tagger()
tagger.train(train_dataset, model=best_model)

print(tagger.model)
accuracy, recall, f1_micro, f1_macro, confusion_mat = tagger.evaluate(
    test_dataset)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("F1 micro:", f1_micro)
print("F1 macro:", f1_macro)
print('Confusion matrix', confusion_mat)
```
✓ 34.5s Python

Test set evaluation metrics :

```
FFNN_Tagger(
  (embedding): Embedding(515, 256)
  (activation): ReLU()
  (ffnn): Sequential(
    (linear1): Linear(in_features=1536, out_features=128, bias=True)
    (act1): ReLU()
    (linear2): Linear(in_features=128, out_features=128, bias=True)
    (act2): ReLU()
    (linear3): Linear(in_features=128, out_features=128, bias=True)
    (act3): ReLU()
    (linear4): Linear(in_features=128, out_features=13, bias=True)
  )
)
Accuracy: 0.9790273556231003
Recall: 0.9790273556231003
F1 micro: 0.9790273556231003
F1 macro: 0.9563300739670564
Confusion matrix [[ 207    0    4    0    0    0    0    3    0    0    0
  [   1 1428    1    0    0    0    0    0    0    1    2    0    1]
  [  13    2   53    0    0    0    0    2    0    0    0    4    2]
  [   0    0    0  253    0    0    0    1    0    0    0    0    2]
  [   0    0    0    0  109    0    0    0    0    0    0    0    0]
  [   0    1    0    0    0  502    1    1    0    0    4    2    1]
  [   0    0    0    0    0    0   35    0    0    0    0    0    1]
  [   1    0    0    0    0    0    0 1152    1    0    0    6    6]
  [   4    0    0    0    0    0    0    3  114    0    1    1    4]
  [   0    4    0    0    0    0    0    0    0   52    0    0    0]
  [   0    0    1    0    0    3    0    0    1    0  387    0    0]
  [   3    0    0    0    0    0    0    5    2    0    0 1550    7]
  [   4   13    1    2    0    0    0    7    0    0    0    2  600]]
```

Graphs

Graphs for context_window ∈ {0...4} vs dev set accuracy, where p = s = context_window for one such configuration :



Code for graph in jupyter notebook

LSTM POS Tagging

- Configuration 1

  embedding_dim = 64
  hidden_dim = 128
  stacks = 2
  bidirectional = False

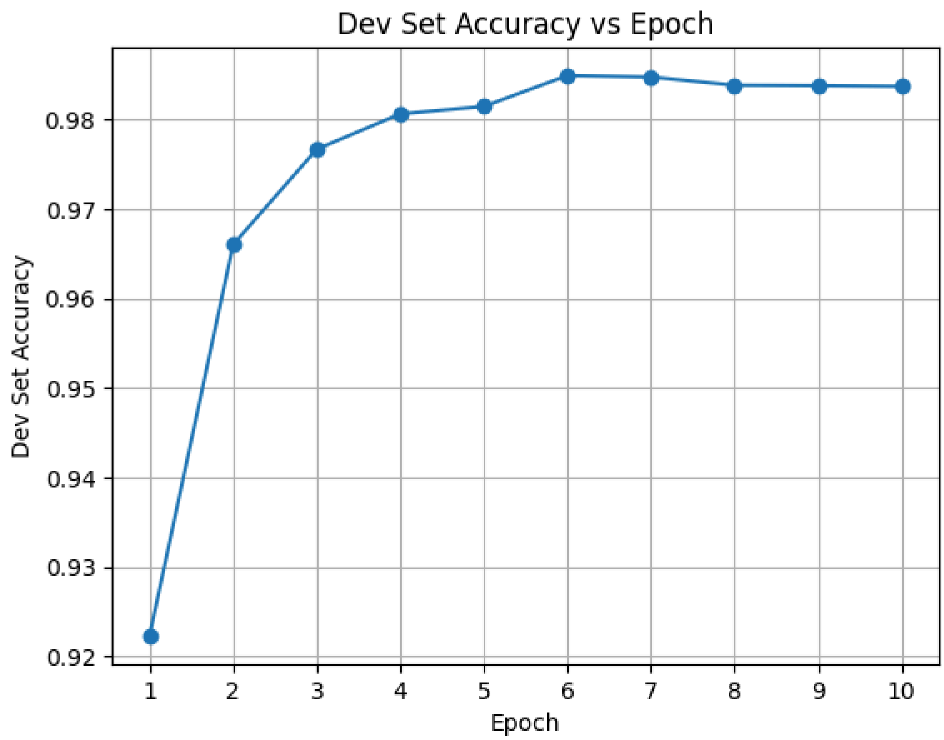
```python
train_dataset = POSDatatset_LSTM(train_filepath)
training_args = {'vocab': train_dataset.vocab, 'tags': train_dataset.tags, 'words_index': train_dataset.words_index,
                 'tags_index': train_dataset.tags_index, 'tags_one_hot': train_dataset.tags_one_hot}
dev_dataset = POSDatatset_LSTM(dev_filepath, training_args)

input_dim = len(train_dataset.vocab)
output_dim = len(train_dataset.tags)
embedding_dim = 64
hidden_dim = 128
stacks = 2
bidirectional = False

model1 = LSTM_Tagger(input_dim, embedding_dim, hidden_dim,
                     stacks, output_dim, bidirectional)
tagger = POS_Tagger('lstm')
tagger.train_graph(train_dataset, dev_dataset, model=model1)

print(tagger.model)
accuracy, recall, f1_micro, f1_macro, confusion_mat = tagger.evaluate(
    dev_dataset)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("F1 micro:", f1_micro)
print("F1 macro:", f1_macro)
print('Confusion matrix', confusion_mat)

table_lstm.add_row(
    [stacks, bidirectional, hidden_dim, embedding_dim, accuracy])
```

✓ 34.0s Python

Dev set evaluation metrics :

```
LSTM_Tagger(
  (embedding): Embedding(514, 64)
  (lstm): LSTM(64, 128, num_layers=2)
  (hidden_to_tag): Linear(in_features=128, out_features=14, bias=True)
)
Accuracy: 0.9828921078921079
Recall: 0.9828921078921079
F1 micro: 0.9828921078921079
F1 macro: 0.9015850177694916
Confusion matrix [[17382     0     0     0     0     0     0     0     0     0     0     0
      0     0]
 [    0   207     0     0     0     0     0     0    12     5     0     0
      3     0]
 [    0     0  1264     0     0     0     0     0     0     1   150     0
      0     0]
 [    0     7     0    44     0     0     0     0     1     7     0     0
      0     0]
 [    0     0     0     0   254     0     0     0     1     0     1     0
      0    10]
 [    0     0     0     0     0   107     0     0     0     0     0     0
      0     0]
 [    0     0    14     0     0     0   481     0     0     3     0    70
      0     0]
 [    0     0     0     0     0     0     0    34     0     0     0     0
      1     0]
 [    9     1     0     0     0     0     0     0  1089    13     0     0
     18    13]
 [    0     0     0     0     0     0     0     0     0   131     0     0
      0     0]
 [    0     0    20     0     0     0     0     0     0     0    53     0
      0     0]
 [    0     0     0     0     0     0     0     0     0     1     1   412
      0     0]
 [    0     0     0     0     0     0     0     0     6    18     0     0
   1527     0]
 [    0     3     0     0     4     0     0     0     9     9     0     0
      0   628]]
```

Epoch vs dev set accuracy graphs :



- Configuration 2

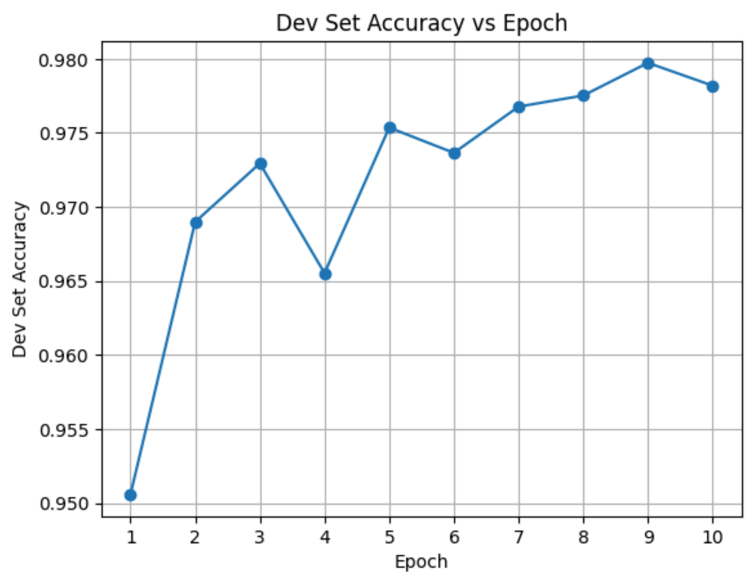
embedding_dim = 64
hidden_dim = 256
stacks = 1
bidirectional = True

```python
train_dataset = POSDatatset_LSTM(train_filepath)
training_args = {'vocab': train_dataset.vocab, 'tags': train_dataset.tags, 'words_index': train_dataset.words_index,
                 'tags_index': train_dataset.tags_index, 'tags_one_hot': train_dataset.tags_one_hot}
dev_dataset = POSDatatset_LSTM(dev_filepath, training_args)

input_dim = len(train_dataset.vocab)
output_dim = len(train_dataset.tags)
embedding_dim = 64
hidden_dim = 256
stacks = 1
bidirectional = True

model2 = LSTM_Tagger(input_dim, embedding_dim, hidden_dim,
                     stacks, output_dim, bidirectional)
tagger = POS_Tagger('lstm')
tagger.train_graph(train_dataset, dev_dataset, model=model2)

print(tagger.model)
accuracy, recall, f1_micro, f1_macro, confusion_mat = tagger.evaluate(
    dev_dataset)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("F1 micro:", f1_micro)
print("F1 macro:", f1_macro)
print('Confusion matrix', confusion_mat)

table_lstm.add_row(
    [stacks, bidirectional, hidden_dim, embedding_dim, accuracy])
```
✓ 1m 15.0s Python

Dev set evaluation metrics :

```
LSTM_Tagger(
  (embedding): Embedding(514, 64)
  (lstm): LSTM(64, 256, bidirectional=True)
  (hidden_to_tag): Linear(in_features=512, out_features=14, bias=True)
)
Accuracy: 0.9782301032301032
Recall: 0.9782301032301032
F1 micro: 0.9782301032301032
F1 macro: 0.8809799170264193
Confusion matrix [[17382     0     0     0     0     0     0     0     0     0     0     0
      0     0]
 [    0   216     0     0     0     0     0     0     3     7     0     0
      1     0]
 [    0     0  1198     0     0     0     0     0     0     1   216     0
      0     0]
 [    0     8     0    40     0     0     0     0     0    11     0     0
      0     0]
 [    0     0     0     0   252     0     0     0     1     2     8     0
      0     3]
 [    0     0     0     0     0   107     0     0     0     0     0     0
      0     0]
 [    0     0    12     0     0     0   483     0     0     1     0    72
      0     0]
 [    0     0     0     0     0     0     0    35     0     0     0     0
      0     0]
 [    0    10     0     0     0     0     0     0  1067    35     0     0
     21    10]
 [    0     0     0     0     0     0     0     0     0   131     0     0
      0     0]
 [    0     0    16     0     0     0     0     0     0     0    57     0
      0     0]
 [    0     0     0     0     0     0     0     0     0     1     1   412
      0     0]
 [    0     4     0     0     0     0     0     1     3    33     0     0
   1510     0]
 [    0     0     6     0     9     0     0     0     2    25     0     0
      0   611]]
```

Epoch vs dev set accuracy graphs :



- Configuration 3

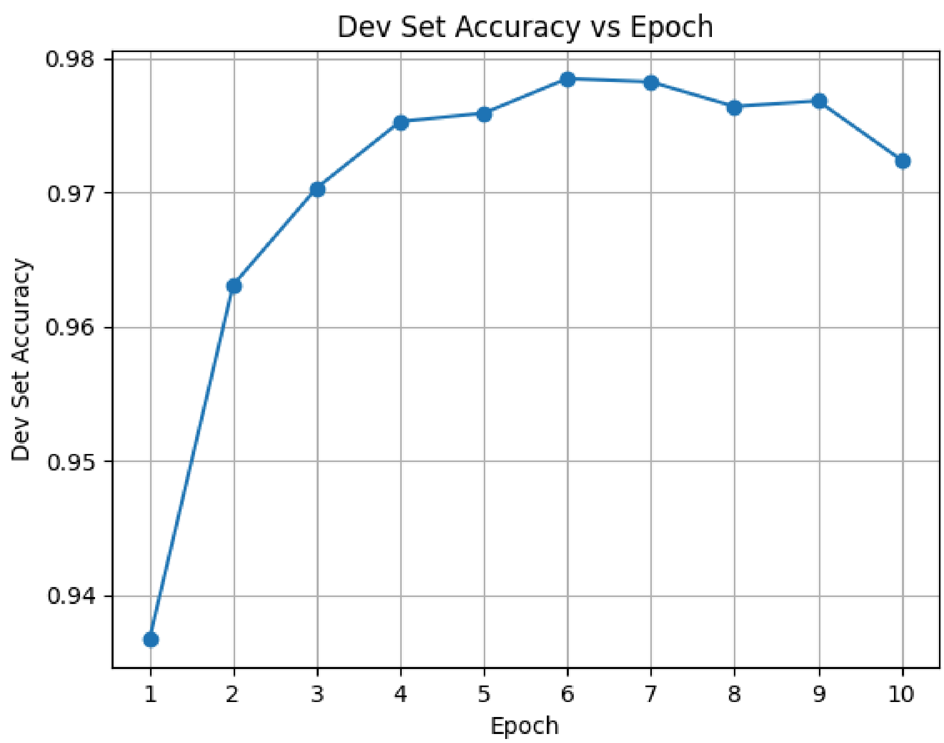
embedding_dim = 256
hidden_dim = 128
stacks = 1
bidirectional = True

```python
train_dataset = POSDatatset_LSTM(train_filepath)
training_args = {'vocab': train_dataset.vocab, 'tags': train_dataset.tags, 'words_index': train_dataset.words_index,
                 'tags_index': train_dataset.tags_index, 'tags_one_hot': train_dataset.tags_one_hot}
dev_dataset = POSDatatset_LSTM(dev_filepath, training_args)

input_dim = len(train_dataset.vocab)
output_dim = len(train_dataset.tags)
embedding_dim = 256
hidden_dim = 128
stacks = 1
bidirectional = True

model3 = LSTM_Tagger(input_dim, embedding_dim, hidden_dim,
                     stacks, output_dim, bidirectional)
tagger = POS_Tagger('lstm')
tagger.train_graph(train_dataset, dev_dataset, model=model3)

print(tagger.model)
accuracy, recall, f1_micro, f1_macro, confusion_mat = tagger.evaluate(
    dev_dataset)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("F1 micro:", f1_micro)
print("F1 macro:", f1_macro)
print('Confusion matrix', confusion_mat)

table_lstm.add_row(
    [stacks, bidirectional, hidden_dim, embedding_dim, accuracy])
```
✓ 45.8s Python

Dev set evaluation metrics :

```
LSTM_Tagger(
  (embedding): Embedding(514, 256)
  (lstm): LSTM(256, 128, bidirectional=True)
  (hidden_to_tag): Linear(in_features=256, out_features=14, bias=True)
)
Accuracy: 0.9720695970695971
Recall: 0.9720695970695971
F1 micro: 0.9720695970695971
F1 macro: 0.8852320788937293
Confusion matrix [[17382     0     0     0     0     0     0     0     0     0     0     0
      0     0]
 [    0   206     0     3     0     0     0     0    11     5     0     0
      2     0]
 [    0     0   967     0     0     0     0     0     0     1   447     0
      0     0]
 [    0     6     0    45     0     0     0     0     1     7     0     0
      0     0]
 [    0     0     0     0   258     0     0     0     0     0     8     0
      0     0]
 [    0     0     0     0     0   107     0     0     0     0     0     0
      0     0]
 [    0     0    13     0     0     0   497     0     0     1     0    57
      0     0]
 [    0     0     0     0     0     0     0    35     0     0     0     0
      0     0]
 [    0     2     0     0     0     0     0     0  1090    12     1     0
     20    18]
 [    0     0     0     0     0     0     0     0     0   131     0     0
      0     0]
 [    0     0     0     0     0     0     0     0     0     0    73     0
      0     0]
 [    0     0     0     0     0     0     4     0     0     1     1   408
      0     0]
 [    0     0     0     0     0     0     0     0     3    18     0     0
   1530     0]
 [    0     0     8     0    11     0     0     0     1     9     0     0
      0   624]]
```

Epoch vs dev set accuracy graphs :



Combining accuracies of all 3 configurations:

stacks	bidirectional	hidden dim	embedding size	accuracy
2	False	128	64	0.9828921078921079
1	True	256	64	0.9782301032301032
1	True	128	256	0.9720695970695971

We observe that the best configuration is Configuration 1 with hyperparameters:
embedding_dim = 64
hidden_dim = 128
stacks = 2
bidirectional = False
epochs = 10
learning rate = 0.001
batch size=32

Testing on Best Configuration

```python
train_dataset = POSDatatset_LSTM(train_filepath)
training_args = {'vocab': train_dataset.vocab, 'tags': train_dataset.tags, 'words_index': train_dataset.words_index,
                 'tags_index': train_dataset.tags_index, 'tags_one_hot': train_dataset.tags_one_hot}
dev_dataset = POSDatatset_LSTM(dev_filepath, training_args)

input_dim = len(train_dataset.vocab)
output_dim = len(train_dataset.tags)
embedding_dim = 64
hidden_dim = 128
stacks = 2
activation = nn.ReLU()
bidirectional = False

best_model = LSTM_Tagger(input_dim, embedding_dim, hidden_dim,
                         stacks, output_dim, bidirectional)
tagger = POS_Tagger('lstm')
tagger.train(train_dataset, dev_dataset, model=best_model)

print(tagger.model)
accuracy, recall, f1_micro, f1_macro, confusion_mat = tagger.evaluate(
    dev_dataset)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("F1 micro:", f1_micro)
print("F1 macro:", f1_macro)
print('Confusion matrix', confusion_mat)
```

✓ 30.7s Python

Test set evaluation metrics :

```
LSTM_Tagger(
  (embedding): Embedding(514, 64)
  (lstm): LSTM(64, 128, num_layers=2)
  (hidden_to_tag): Linear(in_features=128, out_features=14, bias=True)
)
Accuracy: 0.9816849816849816
Recall: 0.9816849816849816
F1 micro: 0.9816849816849816
F1 macro: 0.8828974862201834
Confusion matrix [[17382     0     0     0     0     0     0     0     0     0     0     0
      0     0]
 [    0   216     0     3     0     0     0     0     0     6     0     0
      2     0]
 [    0     0  1347     0     0     0     0     0     0     1    67     0
      0     0]
 [    0    13     0    38     0     0     0     0     0     8     0     0
      0     0]
 [    0     0     0     0   257     0     0     0     1     0     0     0
      0     8]
 [    0     0     0     0     0   107     0     0     0     0     0     0
      0     0]
 [    0     0    13     0     0     0   476     3     0     5     0    71
      0     0]
 [    0     0     0     1     0     0     0    34     0     0     0     0
      0     0]
 [    0    23     0     4     1     0     0     0  1047    15     0     0
     29    24]
 [    0     0     0     0     0     0     0     0     0   131     0     0
      0     0]
 [    0     0    23     0     1     0     0     0     0     0    49     0
      0     0]
 [    0     1     0     0     1     0     1     0     0     0     0   411
      0     0]
 [    0     5     0     0     0     0     0     0     0    26     0     0
   1520     0]
 [    0    65     3     0     6     0     0     0     1     9     0     0
      0   569]]
```

Analysis

In the context_window vs accuracy graph for FFNN, we observe that accuracy increases as size changes from 0 to 1 and then decreases from 1 to 4. This is probably because as context window size increases it becomes increasingly difficult to capture the long range dependencies and the model tends to overfit the data.

In the epoch vs dev set accuracy for LSTM, we observe that generally accuracy increases with epochs. This is because during training, the LSTM network learns to capture sequential patterns and dependencies within the input data. With each epoch, the network updates its parameters based on the optimization algorithm to better fit the training data. As a result, the network becomes more adept at recognizing and generalizing from the patterns present in the training data, leading to improved performance on the dev set. However, in some cases we observe that there is a decrease in accuracy This may be because as epochs increase the model becomes overly specialized to the training set, leading to a decrease in performance on the dev set.

We also observe that the best configuration of LSTM has higher accuracy than that of FFNN on the test set. This is because LSTMs have recurrent connections that allow them to maintain a memory of past inputs which enables them to capture long range dependencies between words that are crucial for accurate POS tagging.