Praktikum AI

RECURRENT NEURAL NETWORK (RNN)

Komang Niko Romano Prodi | 222011356 | 3SD1

Pendahuluan

Pada praktikum RNN ini, digunakan data sentiment yaitu Tweets.csv (Twitter US Airline Sentiment). Data diperoleh dari Kaggle.com. Data berisi label sentiment positif, negatif, dan netral.

Praktikum RNN

1. **Persiapan (Baca Data dan Import Library)**

Persiapan

```
[1] import pandas as pd
```

```
[2] from google.colab import drive
    drive.mount("/content/drive", force_remount=True)
```

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

```
[3] df = pd.read_csv('/content/drive/MyDrive/RNN/Tweets.csv')
    df.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gol
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	Na
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	Na
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	Na
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	Na
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	Na

2. Lakukan Preprocessing

Processing menggunakan library tweet-preprocessor dimana dilakukan penghilangan pada tanda baca dan ubah huruf menjadi huruf kecil.

```
[4] !pip install tweet-preprocessor

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Collecting tweet-preprocessor
      Downloading tweet_preprocessor-0.6.0-py3-none-any.whl (27 kB)
    Installing collected packages: tweet-preprocessor
    Successfully installed tweet-preprocessor-0.6.0
```

```
df['text_clean'] = ''
import preprocessor as p
import re

#forming a separate feature for cleaned tweets
for i,v in enumerate(df['text']):
  df.loc[i,'text_clean'] = p.clean(v)

# converting all text to lower case
df['text_clean'] = df['text_clean'].apply(str.lower)

# using regex to remove punctuation
df['text_clean'] = df['text_clean'].apply(lambda x: re.sub(r'[^\w\s]', '', x))
```

```
[6] df[['text', 'text_clean']]
```

```
tweets = df[['text_clean','airline_sentiment']]
tweets
```

	text_clean	airline_sentiment
0	what said	neutral
1	plus youve added commercials to the experience...	positive
2	i didnt today must mean i need to take another...	neutral
3	its really aggressive to blast obnoxious enter...	negative
4	and its a really big bad thing about it	negative
...	...	...
14635	thank you we got on a different flight to chicago	positive
14636	leaving over minutes late flight no warnings o...	negative
14637	please bring american airlines to	neutral
14638	you have my money you change my flight and don...	negative
14639	we have ppl so we need know how many seats are...	neutral

14640 rows × 2 columns

3. Lakukan Split Data

Split dataset menjadi 80% training set, 10% validation set, dan 10% test set. Jangan lupa mengatur seed = 43

```
[8] import numpy as np
    seed = 43
    df_train, df_val, df_test = np.split(tweets.sample(frac=1, random_state=seed), [int(.8*len(tweets)), int(.9*len(tweets))])
```

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11712 entries, 5747 to 10724
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   text_clean         11712 non-null  object
 1   airline_sentiment  11712 non-null  object
dtypes: object(2)
memory usage: 274.5+ KB
```

```
[10] df_val.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1464 entries, 2676 to 9289
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   text_clean         1464 non-null   object
 1   airline_sentiment  1464 non-null   object
dtypes: object(2)
memory usage: 34.3+ KB
```

4. Membuat Dataloader

```
# dataloaders
batch_size = 5

# make sure to SHUFFLE your data
pad_idx = 0

train_loader = DataLoader(train_data, shuffle=True, batch_size=batch_size,
collate_fn = MyCollate(pad_idx=pad_idx), worker_init_fn=seed_worker,
generator=g)

valid_loader = DataLoader(valid_data, shuffle=True, batch_size=batch_size,
collate_fn = MyCollate(pad_idx=pad_idx), worker_init_fn=seed_worker,
generator=g)

test_loader = DataLoader(test_data, shuffle=True, batch_size=batch_size,
collate_fn = MyCollate(pad_idx=pad_idx), worker_init_fn=seed_worker,
generator=g)
```

5. Cek apakah menggunakan GPU atau CPU

```
import torch

is_cuda = torch.cuda.is_available()

# If we have a GPU available, we'll set our device to GPU. We'll use this device variable later in our code.

if is_cuda:
  device = torch.device("cuda")
  print("GPU is available")
else:
  device = torch.device("cpu")
  print("GPU not available, CPU used")
```

```
GPU not available, CPU used
```

6. Mendefinisikan Model RNN

```
import torch.nn as nn
import torch.nn.functional as F

class SentimentRNN(nn.Module):
  def __init__(self,no_layers,vocab_size,hidden_dim,embedding_dim,output_dim,drop_prob=0.5):
    super(SentimentRNN,self).__init__()
    self.output_dim = output_dim
    self.hidden_dim = hidden_dim
    self.no_layers = no_layers
    self.vocab_size = vocab_size
    self.embedding = nn.Embedding(vocab_size, embedding_dim)
    self.rnn = nn.RNN(input_size=embedding_dim,hidden_size=self.hidden_dim, num_layers=no_layers, batch_first=True)
    self.dropout = nn.Dropout(0.3)
    self.fc = nn.Linear(self.hidden_dim, output_dim)
    self.softmax = nn.LogSoftmax(dim=1)

  def forward(self,x):
    batch_size = x.size(0)
    embeds = self.embedding(x)
    rnn_out, hidden = self.rnn(embeds)
    out = self.dropout(hidden.squeeze(0))
    out = self.fc(out)
    output = self.softmax(out)
    return output
```

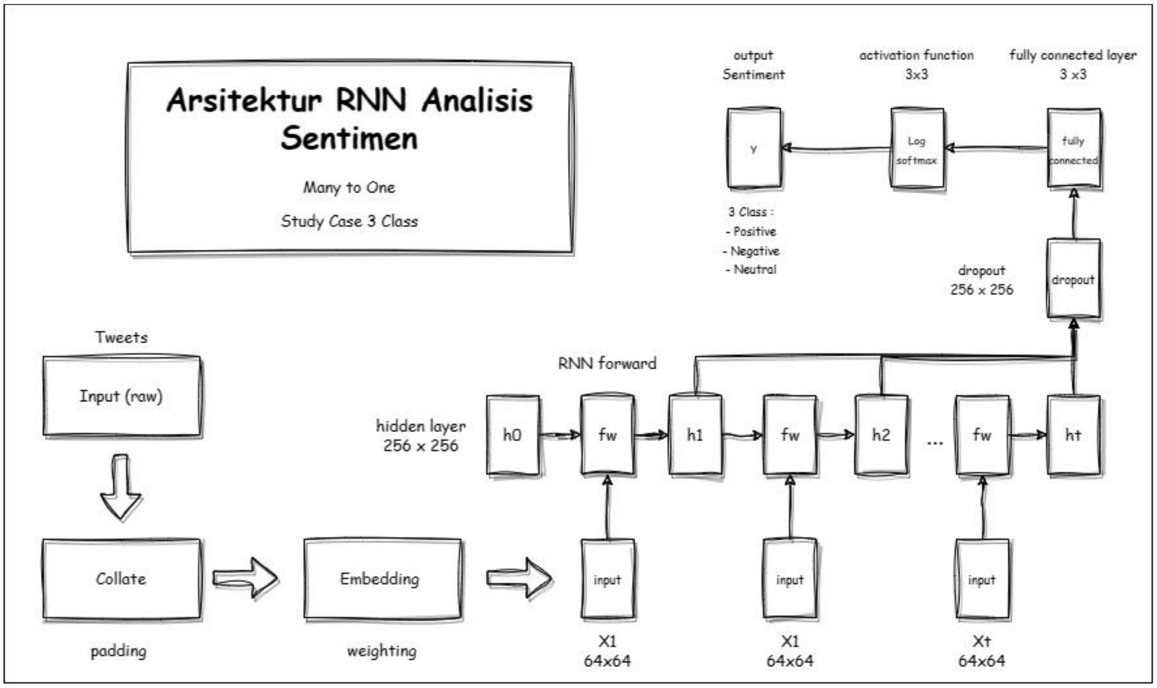
```
no_layers = 1
vocab_size = len(train_data.vocab) + 2 #extra 2 for padding and unknown
embedding_dim = 64
output_dim = 3
hidden_dim = 256

model = SentimentRNN(no_layers,vocab_size,hidden_dim,embedding_dim,output_dim,drop_prob=0.5)

#moving to gpu
model.to(device)
```

```
SentimentRNN(
  (embedding): Embedding(2956, 64)
  (rnn): RNN(64, 256, batch_first=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (fc): Linear(in_features=256, out_features=3, bias=True)
  (softmax): LogSoftmax(dim=1)
)
```

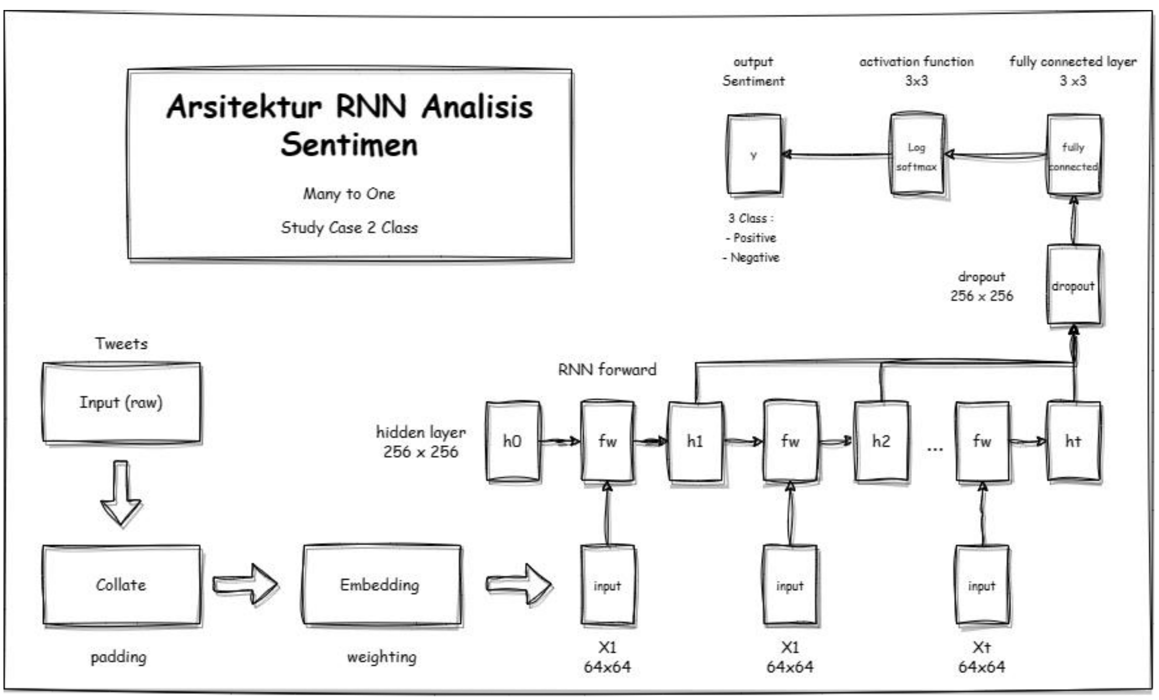
```
[23] import torch.optim as optim
     criterion = nn.CrossEntropyLoss()
     optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

7. Arsitektur Model

a. Model 3 label



b. Model 2 label



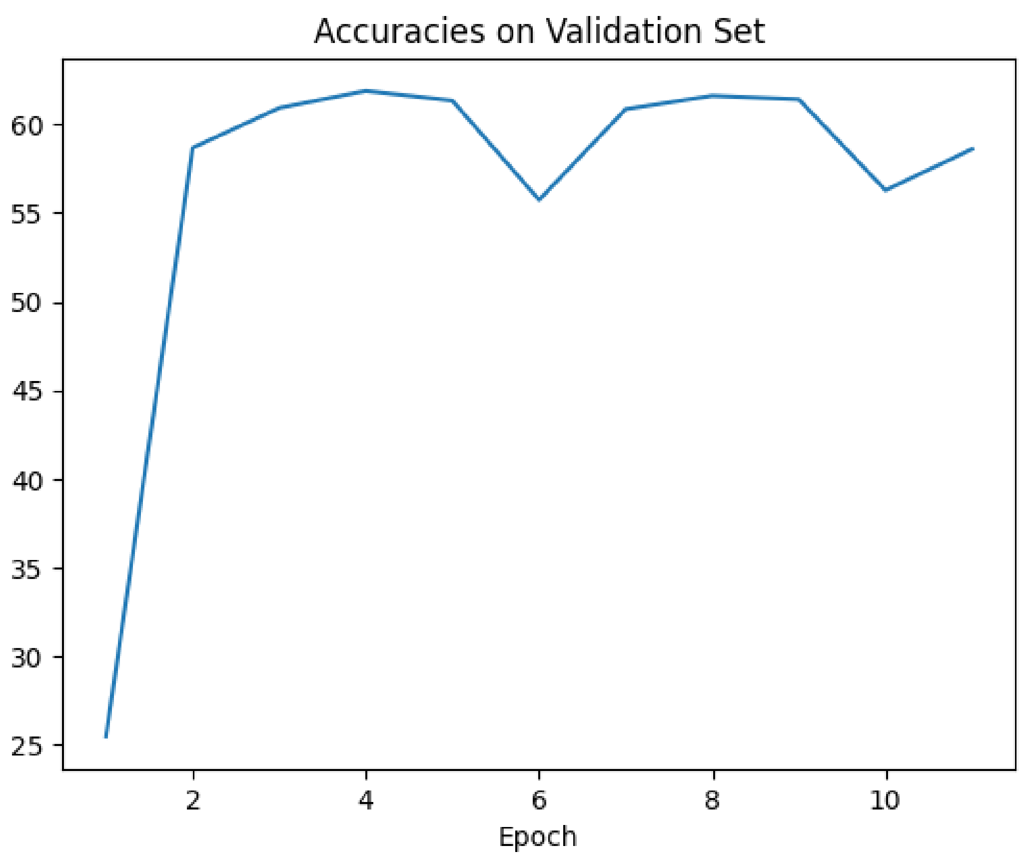
8. Hitung Akurasi Validasi

Proses training dilakukan dengan menggunakan epoch sebesar 10. Model diambil dari tingkat epoch yang memiliki akurasi pada set validasi tertinggi

a. Model 3 label

```
[1,   500] loss: 0.939
[1,  1000] loss: 0.879
[1,  1500] loss: 0.856
[1,  2000] loss: 0.861
val accuracy:58.67%
time elapsed: 0.64 min
[2,   500] loss: 0.841
[2,  1000] loss: 0.867
[2,  1500] loss: 0.867
[2,  2000] loss: 0.891
val accuracy:60.93%
time elapsed: 1.09 min
[3,   500] loss: 0.824
[3,  1000] loss: 0.893
[3,  1500] loss: 0.882
[3,  2000] loss: 0.846
val accuracy:61.89%
time elapsed: 1.53 min
[4,   500] loss: 0.878
[4,  1000] loss: 0.865
[4,  1500] loss: 0.894
[4,  2000] loss: 0.878
val accuracy:61.34%
time elapsed: 2.00 min
[5,   500] loss: 0.855
[5,  1000] loss: 0.845
[5,  1500] loss: 0.843
[5,  2000] loss: 0.849
val accuracy:55.74%
time elapsed: 2.42 min
[6,   500] loss: 0.833
[6,  1000] loss: 0.826
[6,  1500] loss: 0.851
[6,  2000] loss: 0.836
val accuracy:60.86%
time elapsed: 2.84 min
[7,   500] loss: 0.839
[7,  1000] loss: 0.842
[7,  1500] loss: 0.835
[7,  2000] loss: 0.833
val accuracy:61.61%
time elapsed: 3.29 min
```

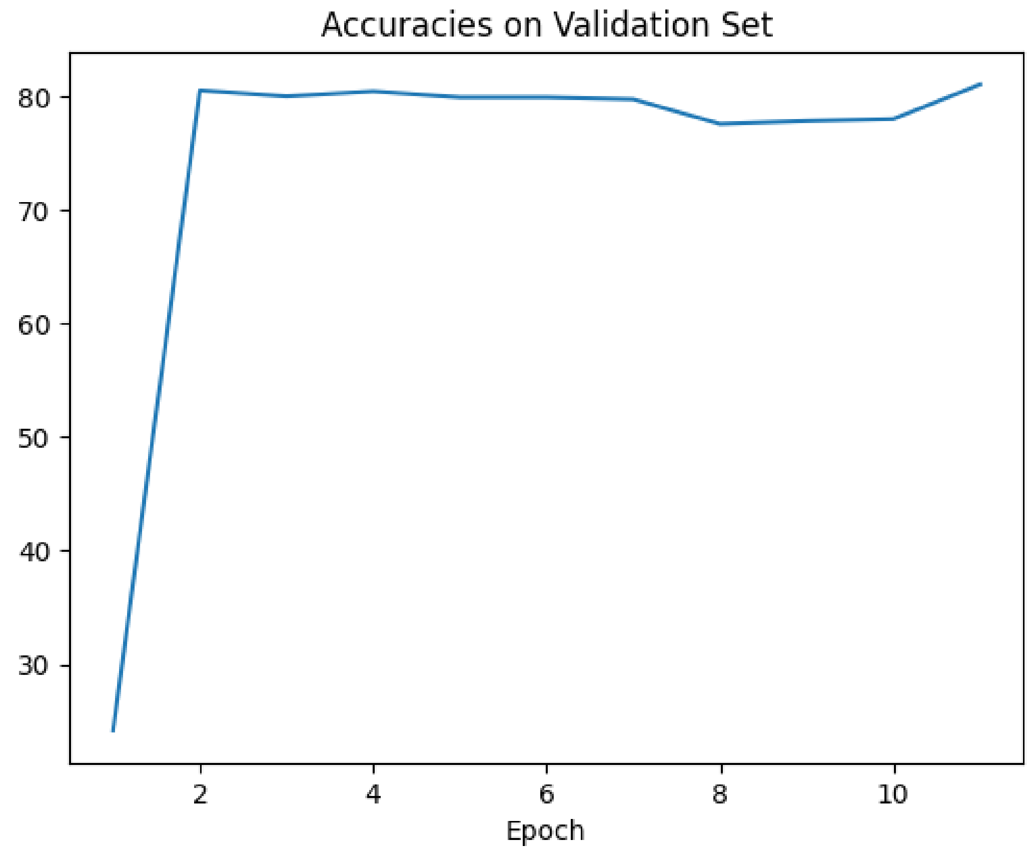
```
[8,   500] loss: 0.837
[8,  1000] loss: 0.834
[8,  1500] loss: 0.842
[8,  2000] loss: 0.832
val accuracy:61.41%
time elapsed: 3.75 min
[9,   500] loss: 0.841
[9,  1000] loss: 0.826
[9,  1500] loss: 0.814
[9,  2000] loss: 0.837
val accuracy:56.28%
time elapsed: 4.19 min
[10,   500] loss: 0.873
[10,  1000] loss: 0.853
[10,  1500] loss: 0.831
[10,  2000] loss: 0.855
val accuracy:58.61%
time elapsed: 4.63 min
```



b. Model 2 label

```
[1,   500] loss: 0.578
[1,  1000] loss: 0.495
```

```
[1,  1500] loss: 0.474
val accuracy:80.50%
time elapsed: 0.54 min
[2,   500] loss: 0.496
[2,  1000] loss: 0.451
[2,  1500] loss: 0.485
val accuracy:79.98%
time elapsed: 0.82 min
[3,   500] loss: 0.505
[3,  1000] loss: 0.465
[3,  1500] loss: 0.466
val accuracy:80.42%
time elapsed: 1.25 min
[4,   500] loss: 0.471
[4,  1000] loss: 0.467
[4,  1500] loss: 0.465
val accuracy:79.90%
time elapsed: 1.59 min
[5,   500] loss: 0.438
[5,  1000] loss: 0.491
[5,  1500] loss: 0.489
val accuracy:79.90%
time elapsed: 1.87 min
[6,   500] loss: 0.462
[6,  1000] loss: 0.529
[6,  1500] loss: 0.498
val accuracy:79.72%
time elapsed: 2.18 min
[7,   500] loss: 0.499
[7,  1000] loss: 0.495
[7,  1500] loss: 0.476
val accuracy:77.56%
time elapsed: 2.46 min
[8,   500] loss: 0.479
[8,  1000] loss: 0.439
[8,  1500] loss: 0.521
val accuracy:77.82%
time elapsed: 2.75 min
[9,   500] loss: 0.472
[9,  1000] loss: 0.477
[9,  1500] loss: 0.495
val accuracy:77.99%
time elapsed: 3.03 min
[10,   500] loss: 0.494
[10,  1000] loss: 0.478
[10,  1500] loss: 0.460
val accuracy:81.02%
time elapsed: 3.30 min
```



9. Hitung Akurasi Testing

a. Model 3 label

```
[ ] compute_accuracy(model_terbaik, test_loader, device)
```

```
tensor(61.7486)
```

b. Model 2 label

```
[ ] compute_accuracy(model_terbaik, test_loader, device)
```

```
tensor(79.7403)
```

10. Perbandingan

Model	Akurasi Testing (%)	Akurasi Validasi Terbaik (%)_
3 label	61.75	61.89 (epoch = 3)
2 label	79.74	81.02 (epoch = 10)

Dari output dan tabel di atas, dapat disimpulkan bahwa:

a. Model 3 label mencapai akurasi validasi terbaik pada epoch ke-3 yaitu sebesar 61.89%
b. Model 2 label mencapai akurasi validasi terbaik pada epoch ke-10 yaitu sebesar 81.02%
c. Model pada 2 label memiliki akurasi testing lebih besar dibandingkan model 2 label yaitu sebesar 79.74%. Sementara itu model 3 label memiliki akurasi testing sebesar 61.75%