

BAB III METODE KERJA

3.1 Waktu dan Tempat

Program ini berlangsung secara online setiap hari kerja (Senin sampai dengan Jumat) selama 8 jam perharinya, dengan rincian sebagai berikut:

Tabel 3.1 Agenda Kelas

Pukul (WIB)	Durasi (jam)	Aktivitas
08.00 s.d. 11.30	3.5	Kelas Sesi Pagi
13.00 s.d. 16.30	3.5	Kelas Sesi Siang
16.30 s.d. 17.30	1	<i>Self-Study</i>

Program ini berlangsung dari bulan Februari 2022 sampai dengan bulan Juli 2022.

3.2 Bahan dan alat

Hardware:

1. Leptop
2. Wifi

Software:

1. Python 3.9.1
2. Jupyter notebook
3. Google collaborator

3.3 Metode dan Proses Kerja

3.3.1 Metode

Metode yang digunakan dalam mengerjakan proyek akhir yaitu

1. Studi literature
Pada tahap ini dilakukan proses mencari referensi – referensi mengenai penelitian – penelitian yang terdahulu.
2. Data pre-processing
Setelah didapatkan beberapa sumber data, data dikumpulkan untuk nantinya akan dijadikan pola, kata kunci pada chatbot.
3. Data *exploration*
Mencari dan mengambil data untuk *chatbot*.
4. *Modelling*

NLP menggunakan QAS, *Neural Networks*, *Text Summarization*, *Text Classification*, dan *Speech Recognition*.

5. *Data processing*

Pada tahap ini, data mentah akan dilakukan *tokenize* dan kemudian dilakukan *training data*. Kemudian melakukan uji *chatbot* sebelum dilakukan *deployment*.

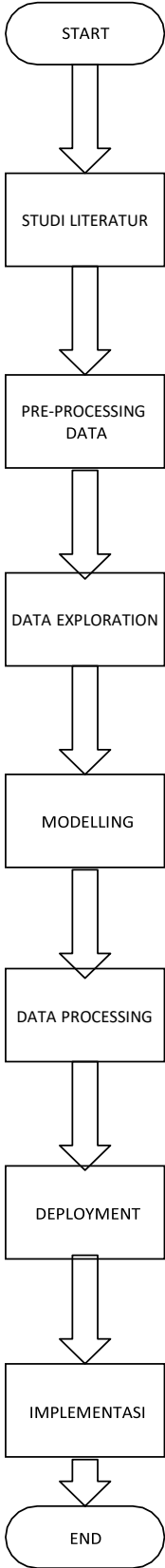
6. *Deployment*

Pada tahap ini, program di *deploy* pada *local* komputer dengan menggunakan *flask with jinja2*

7. *Evaluation*

Dari metode *training* model yang digunakan didapatkan akurasi sebanyak 0.9976, *avg loss* 0.0024 *rate*. Dengan model ini dikatakan metode terbaik untuk mentraining dataset pada *project* akhir. Kelemahan dari *Project Chatbot* Pelajaran Bahasa Indonesia ini masih terdapat data yang belum lengkap.

3.3.2 Proses Kerja



Gambar 3.1 Alur Tahap Penelitian

Dalam proses pelaksanaan proyek akhir ada beberapa proses yang dilewati untuk menghasilkan proyek akhir, antara lain :

1. Mencari sumber referensi yang akan menjadikan dasar dalam proyek akhir
2. Membuat model chatbot dengan neural network

```
model.py
1 import torch
2 import torch.nn as nn
3
4
5 class NeuralNet(nn.Module):
6     def __init__(self, input_size, hidden_size, num_classes):
7         super(NeuralNet, self).__init__()
8         self.l1 = nn.Linear(input_size, hidden_size)
9         self.l2 = nn.Linear(hidden_size, hidden_size)
10        self.l3 = nn.Linear(hidden_size, num_classes)
11        self.relu = nn.ReLU()
12
13    def forward(self, x):
14        out = self.l1(x)
15        out = self.relu(out)
16        out = self.l2(out)
17        out = self.relu(out)
18        out = self.l3(out)
19        # no activation and no softmax at the end
20        return out
21
```

Gambar 3.2 Model *neural network*

3. Membuat *dataset* dengan cara input manual

```
dataset.py
1
2 "intents": [
3
4     "tag": "Salon Perawatan",
5     "patterns": [
6         "Hi",
7         "Hey",
8         "Bagaimana kabar kalian Guys?"
9     ],
10    "responses": [
11        "Hey :-)",
12        "Sungguh luar biasa baik"
13    ],
14 ],
15
16 "tag": "Berita I",
17 "patterns": [
18     | "Apa yang dibacal dengan berita?"
19 ],
20 "responses": [
21     | "Berita adalah suatu teks yang menyampaikan kabar atau informasi kepada masyarakat tentang suatu peristiwa"
22 ],
23 ],
24
25 "tag": "Berita II",
26 "patterns": [
27     | "Apa saja struktur teks berita?"
28 ],
29 "responses": [
30     | "Perhatikan kembali teks - teks berita pada bagian sebelumnya apakah teks berita lain masih kamu sima"
31 ],
32 ],
33 ]
```

Gambar 3.3 *Dataset* untuk *chatbot* pelajaran bahasa indonesia

4. Men-*training* model dengan hyperparameter 1000 *epoch*, 8 *batch size*, *learning rate* 0,001, dan *hidden size* 8.

```

# chatbot.py
# y: Python CrossEntropyLoss needs only class labels, not one-hot
40 label = tags.index(tag)
41 y_train.append(label)
42
43
44 X_train = np.array(X_train)
45 y_train = np.array(y_train)
46
47 # Hyper-parameters
48 num_epochs = 1000
49 batch_size = 8
50 learning_rate = 0.001
51 input_size = len(X_train[0])
52 hidden_size = 8
53 output_size = len(tags)
54 print(input_size, output_size)
55
56 class ChatDataset(Dataset):
57
58     def __init__(self):
59         self.n_samples = len(X_train)
60         self.x_data = X_train
61         self.y_data = y_train
62
63     # support indexing such that dataset[i] can be used to get i-th sample
64     def __getitem__(self, index):
65         return self.x_data[index], self.y_data[index]
66
67
68 # we can call len(dataset) to return the size
69 def __len__(self):
70     return self.n_samples
71
72 dataset = ChatDataset()
73

```

Gambar 3.4 Source code training model

```

# chatbot.py
# y: Python CrossEntropyLoss needs only class labels, not one-hot
40 label = tags.index(tag)
41 y_train.append(label)
42
43
44 X_train = np.array(X_train)
45 y_train = np.array(y_train)
46
47 # Hyper-parameters
48 num_epochs = 1000
49 batch_size = 8
50 learning_rate = 0.001
51 input_size = len(X_train[0])
52 hidden_size = 8
53 output_size = len(tags)
54 print(input_size, output_size)
55
56 class ChatDataset(Dataset):
57
58     def __init__(self):
59         self.n_samples = len(X_train)
60         self.x_data = X_train
61         self.y_data = y_train
62
63     # support indexing such that dataset[i] can be used to get i-th sample
64     def __getitem__(self, index):
65         return self.x_data[index], self.y_data[index]
66
67
68 # we can call len(dataset) to return the size
69 def __len__(self):
70     return self.n_samples
71
72 dataset = ChatDataset()
73
74 # Create model
75 model = nn.Lstm(input_size, hidden_size, output_size, batch_first=True)
76
77 # Create optimizer
78 optimizer = optim.Adam(model.parameters())
79
80 # Training loop
81 for epoch in range(1, num_epochs + 1):
82     # Iterate over data
83     for i, (inputs, targets) in enumerate(dataset):
84         # Forward pass
85         outputs, _ = model(inputs)
86         # Loss function
87         loss = nn.CrossEntropyLoss()(outputs, targets)
88         # Backward pass
89         loss.backward()
90         # Update parameters
91         optimizer.step()
92         # Zero the parameter gradients
93         optimizer.zero_grad()
94     # Print the loss
95     print('Epoch: %d, Loss: %f' % (epoch, loss))
96
97 # Save the model
98 torch.save(model.state_dict(), 'chatbot_model.pth')
99

```

Gambar 3.5 Hasil training

5. Membuat program chatbot dari hasil training

```

# chatbot.py
# y: Python CrossEntropyLoss needs only class labels, not one-hot
40 label = tags.index(tag)
41 y_train.append(label)
42
43
44 X_train = np.array(X_train)
45 y_train = np.array(y_train)
46
47 # Hyper-parameters
48 num_epochs = 1000
49 batch_size = 8
50 learning_rate = 0.001
51 input_size = len(X_train[0])
52 hidden_size = 8
53 output_size = len(tags)
54 print(input_size, output_size)
55
56 class ChatDataset(Dataset):
57
58     def __init__(self):
59         self.n_samples = len(X_train)
60         self.x_data = X_train
61         self.y_data = y_train
62
63     # support indexing such that dataset[i] can be used to get i-th sample
64     def __getitem__(self, index):
65         return self.x_data[index], self.y_data[index]
66
67
68 # we can call len(dataset) to return the size
69 def __len__(self):
70     return self.n_samples
71
72 dataset = ChatDataset()
73
74 # Create model
75 model = nn.Lstm(input_size, hidden_size, output_size, batch_first=True)
76
77 # Create optimizer
78 optimizer = optim.Adam(model.parameters())
79
80 # Training loop
81 for epoch in range(1, num_epochs + 1):
82     # Iterate over data
83     for i, (inputs, targets) in enumerate(dataset):
84         # Forward pass
85         outputs, _ = model(inputs)
86         # Loss function
87         loss = nn.CrossEntropyLoss()(outputs, targets)
88         # Backward pass
89         loss.backward()
90         # Update parameters
91         optimizer.step()
92         # Zero the parameter gradients
93         optimizer.zero_grad()
94     # Print the loss
95     print('Epoch: %d, Loss: %f' % (epoch, loss))
96
97 # Save the model
98 torch.save(model.state_dict(), 'chatbot_model.pth')
99

```

Gambar 3.6 Program chatbot

6. Mendesain web chatbot dengan html, css dan javascript.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4
5 </head>
6 <meta charset="UTF-8">
7 <title>Chatbot</title>
8 </head>
9 <body>
10 <div class="container">
11 <div class="chatbox">
12 <div class="chatbox__support">
13 <div class="chatbox__header">
14 <div class="chatbox__image--header">
15 
16 </div>
17 <div class="chatbox__content--header">
18 <div class="chatbox__heading--header">Chat support</div>
19 <div class="chatbox__description--header">Hi, My name is Sam, how can I help you?</div>
20 </div>
21 </div>
22 <div class="chatbox__messages">
23 <div></div>
24 </div>
25 <div class="chatbox__input">
26 <input type="text" placeholder="write a message..." />
27 <button class="chatbox__send--footer send__button">Send</button>
28 </div>
29 </div>
30 <div class="chatbox__button">
31 <button type="button" class="chatbox__icon"></button>
32 </div>
```

Gambar 3.7 Source Code Html

```
1 {
2   box-sizing: border-box;
3   margin: 0;
4   padding: 0;
5 }
6
7 body {
8   font-family: "Helvetica", sans-serif;
9   font-weight: 400;
10  font-size: 16px;
11  background-color: #f2f2f2;
12 }
13
14 *, html {
15   --primary-gradient: linear-gradient(135deg, #e32020 8.52%, #e32020 8.52%, #e32020 38.8%);
16   --secondary-gradient: linear-gradient(368.8deg, #e32020 -2.14%, #e32020 39.68%);
17   --primary-box-shadow: 0px 1px 1px #e32020, 0px 0px 0px #e32020;
18   --secondary-box-shadow: 0px -1px 1px #e32020, 0px 0px 0px #e32020;
19   --primary: #e32020;
20 }
21
22 /* CHATBOX */
23
24 .chatbox {
25   position: absolute;
26   bottom: 50px;
27   right: 50px;
28 }
29
30 /* CHATBOX IS CLOSE */
31 .chatbox__support {
32   display: flex;
33   flex-direction: column;
34 }
```

Gambar 3.8 Source code css

```

1 class Chatbox {
2   constructor() {
3     this.args = {
4       chatbox: document.querySelector('#chatbox'),
5       chatbox: document.querySelector('#chatbox'),
6       sendbutton: document.querySelector('#sendbutton')
7     }
8
9     this.state = false;
10    this.messages = [];
11  }
12
13  display() {
14    const { chatbox, chatbox, sendbutton } = this.args;
15
16    chatbox.addEventListener('click', () => this.toggleState(chatbox));
17
18    sendbutton.addEventListener('click', () => this.sendMessage(chatbox));
19
20    const input = chatbox.querySelector('input');
21    input.addEventListener('keyup', (e) => {
22      if (e.key === 'Enter') {
23        this.sendMessage(chatbox);
24      }
25    });
26  }
27
28  toggleState(chatbox) {
29    this.state = !this.state;
30
31    // show or hide the box
32    if (this.state) {
33      chatbox.classList.add('show');
34    }
35  }
36
37  sendMessage(chatbox) {
38    const text = chatbox.value;
39    if (text.trim() === '') return;
40
41    this.messages.push({ text });
42    chatbox.value = '';
43
44    // ...
45  }
46 }

```

Gambar 3.9 Source code javascript

7. Membuat program deployment ke web

```

1 from flask import Flask, render_template, request, jsonify
2
3 from chat import get_response
4
5 app = Flask(__name__)
6
7 @app.route('/', methods=['GET', 'POST'])
8 def gv():
9     if request.method == 'GET':
10        return render_template("guru_virtual.html")
11    elif request.method == 'POST':
12        text = request.get_json().get("message")
13        # TODO: check if text is valid
14        response = get_response(text)
15        message = {"answer": response}
16        return jsonify(message)
17
18
19 if __name__ == "__main__":
20     app.run(port=5000, debug=True)

```

Gambar 3.10 Source Code program deployment

8. Menguji aplikasi untuk mengetahui apakah aplikasi berjalan atau tidak.



Gambar 3.11 Pengujian aplikasi