

AI VIET NAM – COURSE 2024

Streamlit – Project

Ngày 22 tháng 6 năm 2024

I. Lý thuyết

Xây dựng và phát triển các ứng dụng AI bao gồm các bước sau:

- Data Handling: bao gồm các bước về thu thập và xử lý dữ liệu
- Exploratory Data Analysis (EDA): phân tích các đặc trưng của dữ liệu
- Modeling: từ các đặc trưng, xây dựng các mô hình và đánh giá tính hiệu quả của mô hình
- Deployment: triển khai mô hình thành sản phẩm trên các nền tảng khác nhau như website, app,...

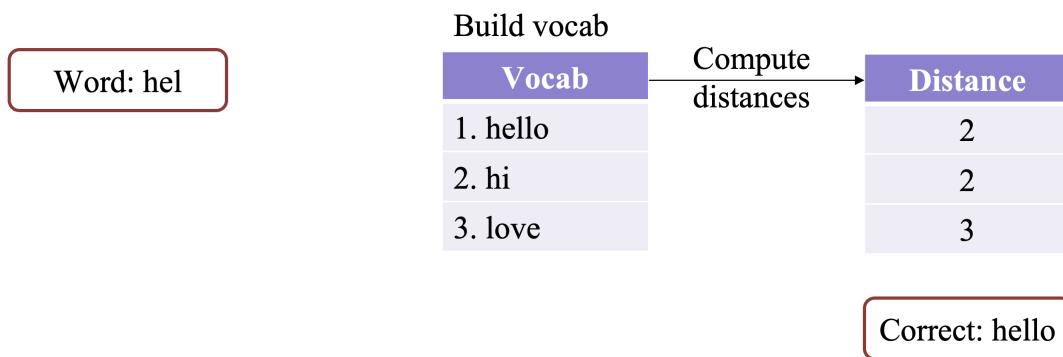
Trong phần này, chúng ta sẽ tập trung tìm hiểu thư viện **Streamlit**, một trong những thư viện mã nguồn mở được phát triển và ứng dụng rộng rãi để triển khai các ứng dụng AI nói chung. Streamlit cung cấp các công cụ hiệu quả để tương tác với người dùng để nhập thông tin hoặc hiển thị thông tin.

Trong các phần tiếp theo, chúng ta tập trung vào phát triển các ứng dụng cơ bản sau:

- Word Correction
- Object Detection
- Chatbot

1. Word Correction.

`levenshtein_distance(hel, hello) = 2`



Hình 1: Khoảng cách chỉnh sửa Levenshtein

Word Correction (Sửa lỗi chính tả) là một trong những ứng dụng cơ bản của xử lý ngôn ngữ tự nhiên, với mục đích xây dựng các ứng dụng nhận đầu vào là một từ, thông qua mô hình sửa lỗi nếu từ đó bị sai thành một từ đúng.

Ví dụ: người dùng nhập vào từ 'hel' là một từ sai, mô hình sẽ gợi ý sửa lỗi thành từ 'hello'. Để đơn giản trong phần này, chúng ta sẽ sử dụng **độ đo khoảng cách chỉnh sửa tối thiểu giữa hai từ**, là **độ đo levenshtein**. Được minh họa như hình trên.

Để giải quyết bài toán này, chúng ta thực hiện các bước sau để xây dựng thuật toán:

- Đầu tiên chúng ta sẽ xây dựng bộ từ điển (gọi là dictionary hoặc vocabulary). Một số từ vựng được chuẩn bị sẵn trong file vocab.txt có thể được tải về [tại đây](#)

```

1  def load_vocab(file_path):
2      with open(file_path, 'r') as f:
3          lines = f.readlines()
4          words = sorted(set([line.strip().lower() for line in lines]))
5      return words
6
7  vocabs = load_vocab(file_path='./vocab.txt')

```

- Tương ứng với mỗi từ trong từ điển, chúng ta tính khoảng cách levenshtein với từ đã có được nhập vào bởi người dùng.

```

1  def levenshtein_distance(token1, token2):
2      distances = [[0]*(len(token2)+1) for i in range(len(token1)+1)]
3
4      for t1 in range(len(token1) + 1):
5          distances[t1][0] = t1
6
7      for t2 in range(len(token2) + 1):
8          distances[0][t2] = t2
9
10     a = 0
11     b = 0
12     c = 0
13
14     for t1 in range(1, len(token1) + 1):
15         for t2 in range(1, len(token2) + 1):
16             if (token1[t1-1] == token2[t2-1]):
17                 distances[t1][t2] = distances[t1 - 1][t2 - 1]
18             else:
19                 a = distances[t1][t2 - 1]
20                 b = distances[t1 - 1][t2]
21                 c = distances[t1 - 1][t2 - 1]
22
23                 if (a <= b and a <= c):
24                     distances[t1][t2] = a + 1
25                 elif (b <= a and b <= c):
26                     distances[t1][t2] = b + 1
27                 else:
28                     distances[t1][t2] = c + 1
29
30     return distances[len(token1)][len(token2)]
31

```

- So sánh và chọn từ có khoảng cách chỉnh sửa bé nhất.

Tiếp theo, chúng ta sẽ xây dựng giao diện bằng thư viện streamlit. Giao diện của ứng dụng được mô tả như hình sau:

Word Correction using Levenshtein Distance

Word:

Compute

Hình 2: Giao diện ứng dụng chỉnh sửa chính tả

```
1 import streamlit as st
2
3 def main():
4     st.title("Word Correction using Levenshtein Distance")
5     word = st.text_input('Word:')
6
7     if st.button("Compute"):
8
9         # compute levenshtein distance
10        leven_distances = dict()
11        for vocab in vocabs:
12            leven_distances[vocab] = levenshtein_distance(word, vocab)
13
14        # sorted by distance
15        sorted_distences = dict(sorted(leven_distances.items(), key=lambda item: item[1]))
16        correct_word = list(sorted_distences.keys())[0]
17        st.write('Correct word: ', correct_word)
18
19        col1, col2 = st.columns(2)
20        col1.write('Vocabulary:')
21        col1.write(vocabs)
22
23        col2.write('Distances:')
24        col2.write(sorted_distences)
25
26 if __name__ == "__main__":
27     main()
```

Cuối cùng, tất cả code sẽ được để trong file levenshtein_distance.py. Sau đó, chúng ta chạy lệnh "streamlit run levelshtain_distance.py" và thực hiện được kết quả như hình sau:

Word:

bok

Compute

Correct word: book

Vocabulary:

```
▼ [  
  0 : "apple"  
  1 : "book"  
  2 : "dog"  
  3 : "hello"  
  4 : "never"  
  5 : "please"  
  6 : "random"  
  7 : "sleep"  
  8 : "start"  
  9 : "understand"  
]
```

Distances:

```
▼ {  
  "book" : 1  
  "dog" : 2  
  "apple" : 5  
  "hello" : 5  
  "never" : 5  
  "random" : 5  
  "sleep" : 5  
  "start" : 5  
  "please" : 6  
  "understand" : 10  
}
```

Hình 3: Kết quả thực nghiệm.

2. Object Detection

Object Detection là ứng dụng quan trọng điển hình của xử lý hình ảnh, với mục tiêu phát hiện các khung hình chứa các đối tượng trong ảnh. Ví dụ minh họa về ứng dụng như hình sau:



Hình 4: Object Detection.

Trong phần này chúng ta sẽ xây dựng ứng dụng cho người dùng tải lên ảnh đầu vào, sử dụng mô hình DNN từ thư viện opencv (Vì phần project này tập trung vào thư viện streamlit vì vậy nên chúng ta sẽ không đi sâu vào mô hình DNN của thư viện opencv). Chúng ta xây dựng 2 hàm: 1 hàm để đẩy ảnh đầu vào vào mô hình DNN để tìm được các bounding box phù hợp và 1 hàm để lọc những bounding box có độ tin cậy thấp và trả về kết quả là các vị trí toạ độ của bounding box đó. Các cài đặt cho mô hình được thiết lập trong 2 file model và prototxt có thể được tải về [tại đây](#). Mã nguồn ví dụ như sau:

```

1 import cv2
2 MODEL = "model/MobileNetSSD_deploy.caffemodel"
3 PROTOTXT = "model/MobileNetSSD_deploy.prototxt.txt"
4
5 def process_image(image):
6     blob = cv2.dnn.blobFromImage(
7         cv2.resize(image, (300, 300)), 0.007843, (300, 300), 127.5
8     )
9     net = cv2.dnn.readNetFromCaffe(PROTOTXT, MODEL)
10    net.setInput(blob)
11    detections = net.forward()
12    return detections
13
14 def annotate_image(
15     image, detections, confidence_threshold=0.5
16 ):
17     # loop over the detections
18     (h, w) = image.shape[:2]
19     for i in np.arange(0, detections.shape[2]):
20         confidence = detections[0, 0, i, 2]
21
22         if confidence > confidence_threshold:
23             # extract the index of the class label from the 'detections',
24             # then compute the (x, y)-coordinates of the bounding box for
25             # the object
26             idx = int(detections[0, 0, i, 1])
27             box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
28             (startX, startY, endX, endY) = box.astype("int")
29             cv2.rectangle(image, (startX, startY), (endX, endY), 70, 2)
30
31     return image

```

Giao diện ứng dụng được mô tả như hình sau:

Object Detection for Images

Upload Image



Drag and drop file here

Limit 200MB per file • JPG, PNG, JPEG

Browse files

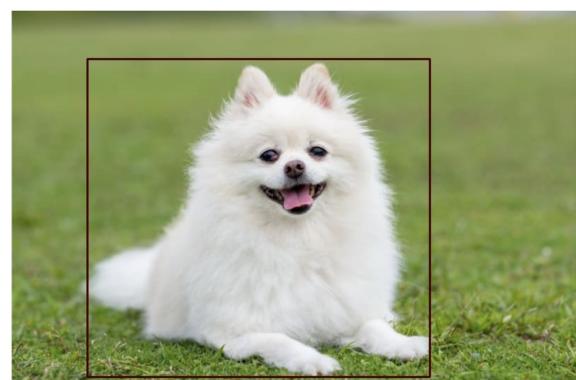
Hình 5: Giao diện ứng dụng object detection.

```
1 import numpy as np
2 from PIL import Image
3 import streamlit as st
4
5 def main():
6     st.title('Object Detection for Images')
7     file = st.file_uploader('Upload Image', type = ['jpg', 'png', 'jpeg'])
8     if file is not None:
9         st.image(file, caption = "Uploaded Image")
10
11    image = Image.open(file)
12    image = np.array(image)
13    detections = process_image(image)
14    processed_image = annotate_image(image, detections)
15    st.image(processed_image, caption = "Processed Image")
16
17 if __name__ == "__main__":
18     main()
```

Sau đó, tất cả code sẽ được để trong file: 'object_detection.py'. Cuối cùng chúng ta chạy lệnh 'streamlit run object_detection.py' để thử nghiệm và thu được kết quả như sau:



Uploaded Image

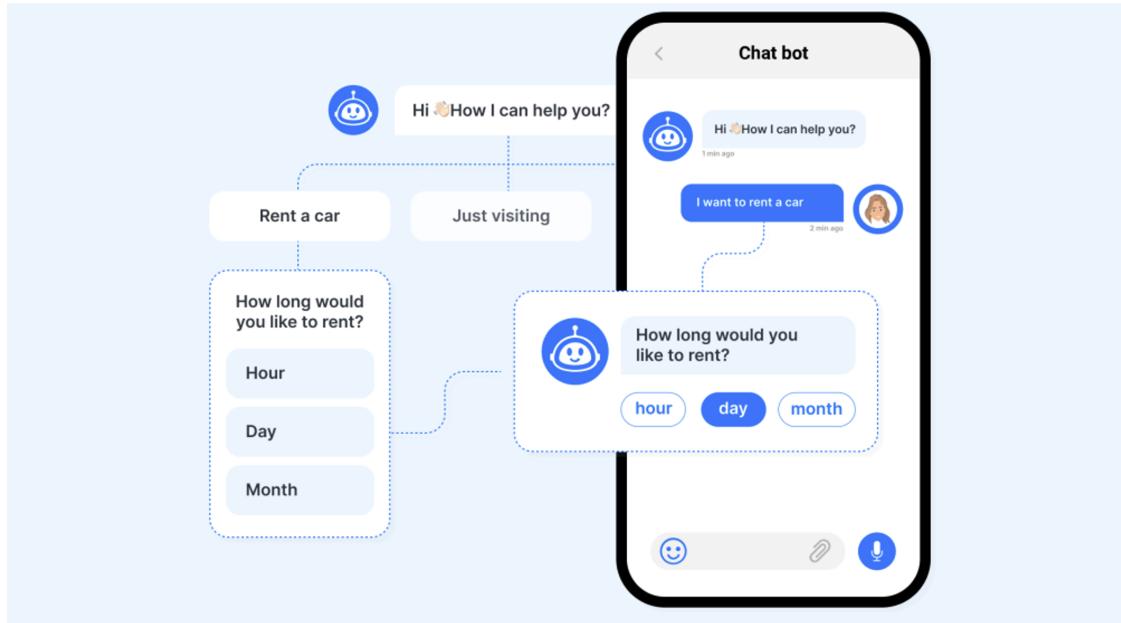


Processed Image

Hình 6: Kết quả thực nghiệm ứng dụng object detection.

3. Chatbot

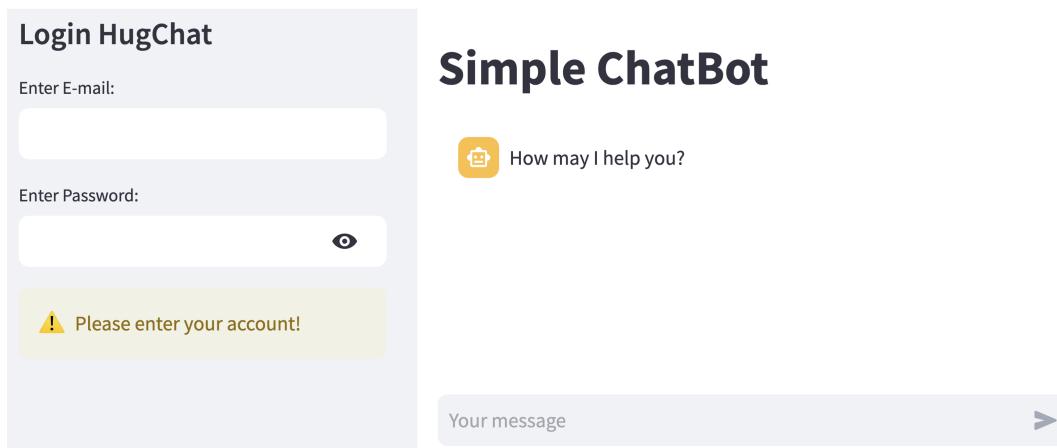
Chatbot là ứng dụng được chú ý phát triển mạnh những năm gần đây, các ứng dụng chatbot chủ yếu tập trung vào các mô hình ngôn ngữ lớn có thể tương tác tốt với các yêu cầu của người dùng. Ví dụ về chatbot được mô tả trong hình sau:



Hình 7: Ứng dụng chatbot.

Trong phần này, chúng ta xây dựng ứng dụng chatbot đơn giản dựa trên thư viện hugchat và streamlit. Đầu tiên chúng ta cần có tài khoản hugging face để có quyền truy cập hugchat. Sau đó truy cập vào [trang](#) sau đây để gửi yêu cầu truy cập hugchat.

Sau khi được chấp nhận quyền truy cập, chúng ta xây dựng giao diện ứng dụng được mô tả như sau:



Hình 8: Giao diện ứng dụng chatbot.

Giao diện có bố cục gồm 2 phần:

- Phần bên trái: Chứa thông tin nhập vào tài khoản và mật khẩu huggingface, được sử dụng để có quyền truy cập vào hugchat

```

1 import streamlit as st
2 from hugchat import hugchat
3 from hugchat.login import Login
4
5 # App title
6 st.title('Simple ChatBot')
7
8 # Hugging Face Credentials
9 with st.sidebar:
10     st.title('Login HugChat')
11     hf_email = st.text_input('Enter E-mail:')
12     hf_pass = st.text_input('Enter Password:', type='password')
13     if not (hf_email and hf_pass):
14         st.warning('Please enter your account!')
15     else:
16         st.success('Proceed to entering your prompt message!')
17

```

- Phần bên phải (phần trung tâm): Chứa khu vực nhập vào văn bản và trả về phản hồi từ chatbot.

```

1 # Store LLM generated responses
2 if "messages" not in st.session_state.keys():
3     st.session_state.messages = [{"role": "assistant", "content": "How
may I help you?"}]
4
5 # Display chat messages
6 for message in st.session_state.messages:
7     with st.chat_message(message["role"]):
8         st.write(message["content"])
9

```

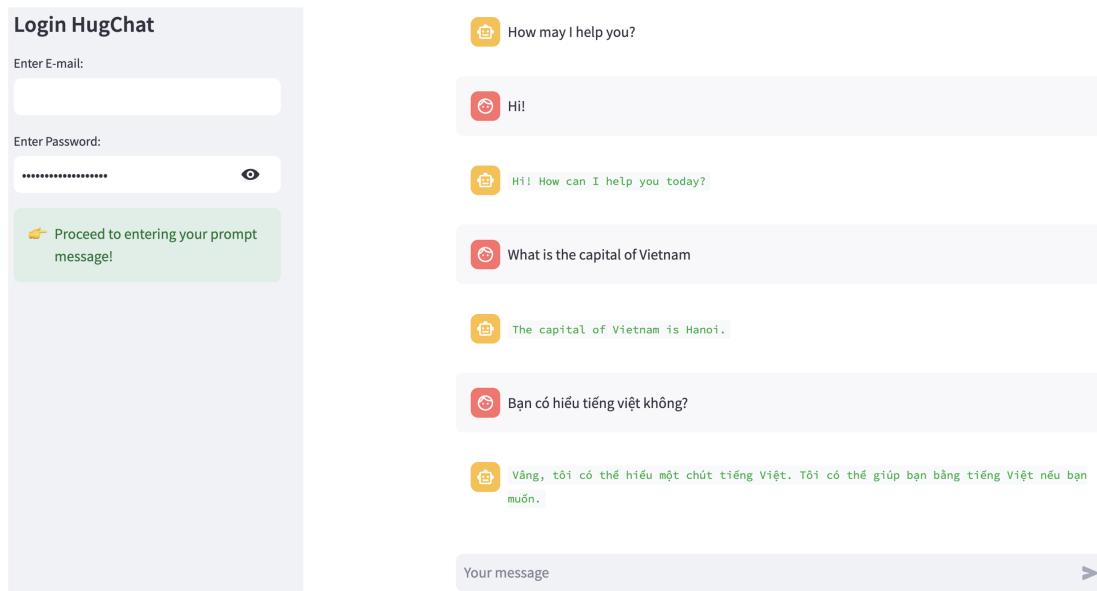
Tương ứng với mỗi request gửi đến chatbot, mô hình sẽ đăng nhập vào hugchat và đưa ra phản hồi tương ứng:

```

1 # Function for generating LLM response
2 def generate_response(prompt_input, email, passwd):
3     # Hugging Face Login
4     sign = Login(email, passwd)
5     cookies = sign.login()
6     # Create ChatBot
7     chatbot = hugchat.ChatBot(cookies=cookies.get_dict())
8     return chatbot.chat(prompt_input)
9
10 # User-provided prompt
11 if prompt := st.chat_input(disabled=not (hf_email and hf_pass)):
12     st.session_state.messages.append({"role": "user", "content": prompt})
13     with st.chat_message("user"):
14         st.write(prompt)
15
16 # Generate a new response if last message is not from assistant
17 if st.session_state.messages[-1]["role"] != "assistant":
18     with st.chat_message("assistant"):
19         with st.spinner("Thinking..."):
20             response = generate_response(prompt, hf_email, hf_pass)
21             st.write(response)
22     message = {"role": "assistant", "content": response}
23     st.session_state.messages.append(message)
24

```

Sau khi hoàn thiện code vào file 'chatbot.py', chúng ta chạy lệnh: 'streamlit run chatbot.py' và thử nghiệm thu được kết quả như sau:



Hình 9: Kết quả thực nghiệm ứng dụng chatbot.

II. Câu hỏi trắc nghiệm

Câu hỏi 1: Hàm nào sau đây được sử dụng để hiển thị chuỗi văn bản trong streamlit.

- a) st.text(...)
- b) st.image(...)
- c) st.selectbox(...)
- d) st.slider(...)

Câu hỏi 2: Đoạn code nào sau đây thể hiện đúng để hiển thị cho giao diện sau:

Your favorite colors:

Yellow × Blue ×

You selected:

```
▼ [
  0 : "Yellow"
  1 : "Blue"
]
```

```

1 a)
2 options = st.multiselect("Your favorite colors:")
3 st.write("You selected:", options)
4
5 b)
6 options = st.multiselect("Your favorite colors:", ["Green", "Yellow", "Red", "Blue"], ["Yellow", "Red"])
7
8 c)
9 options = st.multiselect("Your favorite colors:", ["Green", "Yellow", "Red", "Blue"], ["Yellow", "Red"])
10 st.write("You selected:", options)
11
12 d)
13 options = st.selectbox("Your favorite colors:", ["Green", "Yellow", "Red", "Blue"], ["Yellow", "Red"])
14 st.write("You selected:", options)

```

Câu hỏi 3: Hàm nào sau đây trong streamlit sử dụng để người dùng nhập văn bản trên giao diện.

- a) st.text(...)
- b) st.multibox(...)
- c) st.audio(...)
- d) st.text_input(...)

Câu hỏi 4: Khai báo nào sau đây là sai.

- a) st.image(image_path, caption='A cat', width=100, channels='RGB')
- b) st.image(image_path, caption='A cat', width=None, channels='BGR')
- c) st.image(image_path, caption='A cat', width='RGB', channels='BGR')

d) st.image(image_path, caption='A cat', width=None, channels='RGB')

Câu hỏi 5: Tính khoảng cách chỉnh sửa levenshtein của 2 từ sau: "elmets" và "elements".

- a) 2
- b) 3
- c) 4
- d) 5

Câu hỏi 6: Hàm `st.session_state` trong streamlit được sử dụng để làm gì.

- a) Hiển thị chuỗi văn bản
- b) Hiển thị hình ảnh
- c) Lưu hình ảnh
- d) Chia sẻ các biến tổng mõi phiên truy cập của người dùng

Câu hỏi 7: Đoạn code nào thể hiện đúng cho giao diện sau đây.

```

1 a)
2 col1, col2 = st.columns(2)
3 f_name = col1.text_input('First Name')
4 l_name = col2.text_input('Last Name')
5
6 b)
7 with st.form("my_form"):
8     col1, col2 = st.columns(2)
9     f_name = col1.text_input('First Name')
10    l_name = col2.text_input('Last Name')
11
12 c)
13 with st.form("my_form"):
14     col1, col2 = st.columns(2)
15     f_name = col1.text_input('First Name')
16     l_name = col2.text_input('Last Name')
17     submitted = st.form_submit_button("Submit")

```

```
18
19 d)
20 with st.form("my_form"):
21     col1, col2 = st.columns(2)
22     f_name = col1.text_input('First Name')
23     l_name = col2.text_input('Last Name')
24     submitted = st.form_submit_button("Submit")
25     if submitted:
26         st.write("First Name: ", f_name,
27                 " - Last Name:", l_name)
```

Câu hỏi 8 Hàm nào sau đây cho phép người dùng tải lên nhiều file.

- a) uploaded_files = st.file_uploader("Choose files", accept_multiple_files=True)
- b) uploaded_files = st.upload_file_uploader("Choose files", accept_multiple_files=True)
- c) uploaded_files = st.file_uploader("Choose files")
- d) uploaded_files = st.upload_file_uploader("Choose files")

Câu hỏi 9 Hàm nào sau đây không được sử dụng để hiển thị code trong streamlit?

- a) st.code(...)
- b) st.echo(...)
- c) st.markdown(...)
- d) st.slider(...)

Câu hỏi 10 Dung lượng mặc định tối đa mỗi file được tải lên trong streamlit là bao nhiêu.

- a) 100 MB
- b) 200 MB
- c) 300 MB
- d) 400 MB