Xây dựng ứng dụng chatbot trên nền tảng ứng dụng web

Chatbot không còn là xa lạ với thời đại ngày nay, sơ khai là một chương trình nhỏ lưu trữ các mẫu cụ thể và xuất thông tin lại với người dùng những mẫu cụ thể đó dựa trên những đầu vào và kết quả phân tích của lập trình viên để đưa ra một câu trả lời hợp lý.

Ngày nay với nhiều ứng dụng như hỗ trợ khách hàng, tự động trả lời những vấn đề trong việc học tập hay làm việc chúng ta có nhưng Chatbot hỗ trợ khách hàng, Chatbot hỗ trợ học tập, Chatbot hỗ trợ việc tìm kiếm thông tin...

Trong quy mô ứng dụng Chatbot này chúng ta sẽ xây dựng một Chatbot từ cơ bản đến nâng cao để hoàn thiện một Chatbot cơ bản phục vụ cho mục đích học tập và nghiên cứu về sau để tùy biến dữ liệu tương thích với yêu cầu đặt ra khi một yêu cầu nào đó muốn thỏa mãn điều người ta mong muốn

Yêu cầu

Để có thể được đơn giản hóa tránh bị thông tin bị ngộp trong quá trình học tập và nghiên cứu ứng dụng này chỉ cần nắm cơ bản về `python`, `HTML`, `Javacscript`, `CSS` (*có thể không cần thiết vì nó chỉ hỗ trợ cho việc giao diện dễ nhìn*)

Trong phần đầu tiên chúng ta sẽ sử dụng một Framework đơn giản và thịnh hành là `Flask` để viết các điều khiển, `HTML` dùng để hiển thị cho người dùng nhập dữ liệu vào cho Chatbot có dữ liệu để hoạt động.

Cài đặt cơ bản

Như chúng ta đã đề cập ở trên thì chúng ta phải có tối thiểu là `python` đã được cài đặt vì nó là một nền tảng để chạy những cái chúng ta sẽ làm. Ngoài ra cũng nên yêu cầu phiên bản của `python` là từ phiên bản `3x` trở lên. Tiếp đến là cài đặt Framework `Flask` dùng lệnh như sau

```
pip3 install Flask
```

Bắt đầu nền tảng

Sau khi chúng ta cài đặt xong những thứ cơ bản thì chúng ta sẽ bắt đầu để xây dựng hệ thống Chatbot giả sử chúng ta tạo một thư mục `Chatbot` nếu dùng hệ điều hành `Windows` thì chúng ta có thể làm trực tiếp với giao diện, còn với hệ điều hành mã nguồn mở thì chúng ta sử dụng lệnh `mkdir Chatbot` như vậy là chúng ta đã xong phần tạo thư mục chứa mã nguồn dự án `Chatbot` của chúng ta.

Chúng ta sẽ bắt đầu tạo một tập tin `chatbot.py` đây là tập tin sẽ lưu trữ mã nguồn chúng ta sẽ lập trình cho nó. Tập tin này được lưu trữ trong thư mục `Chatbot` mà ta đã tạo trước đó. đây là mã nguồn đầu tiên của tập tin `chatbot.py`

```
# -*- coding: utf-8 -*-

from flask import Flask
from flask import render_template

app = Flask("app", template_folder="./")

app.config["DEBUG"] = True

@app.route('/')
def homepage():
    return render_template('chatbot.html');

if __name__ == "__main__":
    app.run('0.0.0.0', port=9090)
```

Ở đoạn mã trên chúng ta sẽ lần lượt tìm hiểu các chức năng và hàm của nó

Dòng đầu tiên `# -*- coding: utf-8 -*-` thông báo cho trình biên dịch `python` hiểu là chúng ta sẽ viết mã nguồn có ký tự `utf-8` ví dụ như tiếng việt của chúng ta chẳng hạn, nếu sử dụng tiếng anh thì không cần thiết khai báo dòng này nhưng nên khai báo vì biết đâu trong quá trình chúng ta viết chúng ta sẽ sử dụng như ký tự của các biểu tượng.

Dòng `from flask import Flask` và `from flask import render_template` là hai dòng dùng để gọi Framework `Flask` và hàm để xuất giao diện cho chúng ta `Flask` là một đối tượng còn `render_template` là một hàm được xây dựng sẵn của Framework `Flask`

Dòng `app = Flask("app", template_folder="./")` là dòng khởi tạo đối tượng `app` từ đối tượng `Flask` trong đó có hai tham trị được truyền vào là `app` và `template_folder` là `./` với `app` là cái tên mà khi chúng ta xem tiến trình sẽ hiện trong `taskmanager` của hệ điều hành. còn `template_folder` là khai báo rằng Flask sẽ lấy các mẫu giao diện từ thư mục hiện tại chưa tập tin `chatbot.py`

Đoạn mã `app.config["DEBUG"] = True` là khai báo chúng ta đang viết và kiểm thử mã nguồn nếu có lỗi thì sẽ hiện ra cho chúng ta để chúng ta tiến hành chỉnh sửa và loại bỏ lỗi.

Đoạn mã tiếp theo

```
@app.route('/')
def homepage():
    return render_template('chatbot.html');
```

Đoạn này sẽ khai báo cho ứng dụng khi người dùng đi vào trang chủ thì sẽ tải mẫu giao diện `chatbot.html` để hiển thị cho người dùng thấy. Trong `python` có một chức năng gọi hàm lồng nhau cho nên `@app.route('/')` chính là một hàm được định nghĩa sẵn và sẽ gọi hàm `homepage` để chạy những mã lệnh chúng ta viết ở trong hàm này.

Đoạn cuối cùng

```
if __name__ == "__main__":
    app.run('0.0.0.0', port=9090)
```

Đây là đoạn mà ứng dụng sẽ bắt đầu chạy và chúng ta gọi hàm `run` của từ đối tượng `Flask` với hai tham trị đơn giản đầu tiên `'0.0.0.0'` là `hostname` ứng dụng sẽ chạy và `port=9090` là ứng dụng chạy trên cổng `9090` trong trường hợp `hostname` là `0.0.0.0` thì ứng dụng sẽ lấy địa chỉ IP của máy.

Đến đây thì mã nguồn cơ bản đã hoàn thành nhưng chúng ta cần phải tạo một mẫu giao diện để ứng dụng bắt đầu chạy và hiện thị, chúng ta sẽ bắt đầu xây dựng mẫu giao diện `chatbot.html` như sau

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Simple Chatbot</title>
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
<style>
* {
    margin:0;
    padding:0;
    box-sizing:border-box;
    outline:none;
}
```

```
</style>
</head>
<body>
    <h1>Simple Chatbot</h1>
</body>
</html>
```

Chúng ta lưu nó vào thư mục `Chatbot` đã tạo vừa rồi sau khi chúng ta lưu xong chúng ta dùng cửa sổ lệnh trong Windows là `cmd` còn trong hệ điều hành mã nguồn mở là `terminal` gõ lệnh `python3 chatbot.py`

Sau khi chạy chúng ta sẽ có kết quả như sau

```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production
deployment.
 * Running on http://192.168.0.144:9090/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 583-830-554
```

Trong kết quả trên chúng ta thấy `Running on http://192.168.0.144:9090/` thì có nghĩa là mã nguồn của chúng ta đã được chạy thành công tại địa chỉ IP `192.168.0.144` với cổng `9090` và `192.168.0.144` cũng là địa chỉ `127.0.0.1` là địa chỉ `localhost` chúng ta mở trình duyệt lên gõ vào `http://192.168.0.144:9090/` hoặc `http://127.0.0.1:9090/` và chúng ta sẽ thấy kết quả là

```
Simple Chatbot
```

Như vậy chúng ta đã hoàn thành bước cơ bản đầu tiên của công việc tạo một nền tảng Chatbot của chúng ta. Chúng ta sẽ đi tới bước xây dựng giao diện đơn giản cho Chatbot của chúng ta.

Xây dựng giao diện Chatbot

Như vậy chúng ta đã xây dựng xong được nền tảng bước này chúng ta sẽ xây dựng một giao diện cơ bản cho Chatbot nếu như chưa rành nhiều về `HTML` và `CSS` bạn cứ viết theo như bản mẫu là được còn hiểu sâu nó chúng ta sẽ tìm hiểu ở trang HTML Tutorial và CSS Tutorial để tìm hiểu thêm chi tiết

Chúng ta sẽ bắt đầu sửa tập tin `chatbot.html` như sau

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Simple Chatbot</title>
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
<style>
* {
    margin:0;
    padding:0;
    box-sizing:border-box;
    outline:none;
}
*:before,*:after{
    -webkit-box-sizing:border-box;
    -moz-box-sizing:border-box;
    box-sizing:border-box;
}
::-webkit-scrollbar{
    width:2px;
    height:2px;
}
::-webkit-scrollbar-track{
    -webkit-box-shadow:inset 0 0 6px rgba(0,0,0,0.3);
    border-radius:1px;
}
::-webkit-scrollbar-thumb{
    border-radius:1px;
    -webkit-box-shadow:inset 0 0 6px rgba(0,0,0,0.5);
}
body {
    background-color:#f1f1f1;
    font-family:arial,sans-serif;
}
h1, #wrapper {
    max-width:90%;
    width:500px;
    margin:0 auto;
}
```

```css
h1 {
    margin-top:20px;
    margin-bottom:20px;
}
#wrapper {
    background-color: #fff;
    box-shadow: 0 5px 10px 0 rgb(0 0 0 / 10%);
    -moz-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    -webkit-box-shadow: 0 5px 10px 0 rgb(0 0 0 / 10%);
    -o-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    -ms-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    border-radius:4px;
}
#conversation {
    max-height:250px;
    min-height:250px;
    padding:15px;
    overflow-x: hidden;
    overflow-y: auto;
}
.item {
    margin-bottom: 10px;
    display: flex;
}
.me {
    justify-content: end;
}
.item p {
    padding: 8px;
    background-color: #eee;
    border-radius: 5px;
    max-width: 80%;
    box-shadow: 0 1px 0 0 rgba(0,0,0,0.18);
    font-family:arial,sans-serif;
}
.me p {
    background-color:#e5efff;
    box-shadow: 0 1px 0 0 #c8deff
}
.flex {
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: nowrap;
}
.chat {
    background-color: #eee;
    border: none;
    border-radius: 0 0 4px 4px;
```

```
}
input, button {
    background-color: #eee;
    padding: 6px 12px;
    font-size: 14px;
    color: #555;
    border: none;
    border-radius: 4px 0 0 4px;
    outline: none;
    font-family:arial,sans-serif;
}
input {
    width: 80%;
}
button {
    border-radius: 0 0 4px 0;
    background-color: #337ab7;
    width: 20%;
    cursor: pointer;
    border:none;
    color:#fff;
}
</style>
</head>
<body>
    <h1>Simple Chatbot</h1>
    <div id="wrapper">
        <div id="conversation">
            <div class="item">
                <p>Chúng ta có nên chống lại với người ngu hay không ?</p>
            </div>
            <div class="item me">
                <p>Theo Albert Einstein thì Không thể chống lại những người
ngu vì chúng quá đông.</p>
            </div>
        </div>
        <form action="" method="post" onsubmit="return false;">
            <p class="flex chat">
                <input type="text" name="message" value="" placeholder="Aa"
/>
                <button type="submit">Gửi</button>
            </p>
        </form>
    </div>
</body>
</html>
```

Như vậy là chúng ta đã hoàn thành được giao diện Chatbot đơn giản bao gồm một ô nhập liệu cho người dùng, một nút nhấn để người dùng gửi thông tin lên cho Chatbot xử lý chúng ta sẽ tiếp tục xây dựng mã nguồn `Javascript` để lấy dữ liệu gửi lên bất đồng bộ cho Chatbot ở phần tiếp theo.

Xây dựng mã thực thi ở máy trạm lấy dữ liệu người dùng nhập vào

Sau khi chúng ta xây dựng xong giao diện cho Chatbot đơn gian ở bước trước giờ chúng ta sẽ bắt đầu xây dựng mã `Javascript` để lấy dữ liệu và gửi bất đồng bộ lên Chatbot xử lý. Nếu bạn chưa rành về `Javascript` bạn có thể tham khảo JavaScript Tutorial

Chúng ta sẽ tiếp tục chỉnh sửa tập tin `chatbot.html` như sau

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Simple Chatbot</title>
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
<style>
* {
    margin:0;
    padding:0;
    box-sizing:border-box;
    outline:none;
}
*:before,*:after{
    -webkit-box-sizing:border-box;
    -moz-box-sizing:border-box;
    box-sizing:border-box;
}
::-webkit-scrollbar{
    width:2px;
    height:2px;
}
::-webkit-scrollbar-track{
    -webkit-box-shadow:inset 0 0 6px rgba(0,0,0,0.3);
    border-radius:1px;
}
::-webkit-scrollbar-thumb{
    border-radius:1px;
    -webkit-box-shadow:inset 0 0 6px rgba(0,0,0,0.5);
```

```
}
body {
    background-color:#f1f1f1;
    font-family:arial,sans-serif;
}
h1, #wrapper {
    max-width:90%;
    width:500px;
    margin:0 auto;
}
h1 {
    margin-top:20px;
    margin-bottom:20px;
}
#wrapper {
    background-color: #fff;
    box-shadow: 0 5px 10px 0 rgb(0 0 0 / 10%);
    -moz-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    -webkit-box-shadow: 0 5px 10px 0 rgb(0 0 0 / 10%);
    -o-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    -ms-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    border-radius:4px;
}
#conversation {
    max-height:250px;
    min-height:250px;
    padding:15px;
    overflow-x: hidden;
    overflow-y: auto;
}
.item {
    margin-bottom: 10px;
    display: flex;
}
.me {
    justify-content: end;
}
.item p {
    padding: 8px;
    background-color: #eee;
    border-radius: 5px;
    max-width: 80%;
    box-shadow: 0 1px 0 0 rgba(0,0,0,0.18);
    font-family:arial,sans-serif;
}
.me p {
    background-color:#e5efff;
    box-shadow: 0 1px 0 0 #c8deff
}
```

```
.flex {
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: nowrap;
}
.chat {
    background-color: #eee;
    border: none;
    border-radius: 0 0 4px 4px;
}
input, button {
    background-color: #eee;
    padding: 6px 12px;
    font-size: 14px;
    color: #555;
    border: none;
    border-radius: 4px 0 0 4px;
    outline: none;
    font-family:arial,sans-serif;
}
input {
    width: 80%;
}
button {
    border-radius: 0 0 4px 0;
    background-color: #337ab7;
    width: 20%;
    cursor: pointer;
    border:none;
    color:#fff;
}
</style>
</head>
<body>
    <h1>Simple Chatbot</h1>
    <div id="wrapper">
        <div id="conversation">
            <div class="item">
                <p>Chúng ta có nên chống lại với người ngu hay không ?</p>
            </div>
            <div class="item me">
                <p>Theo Albert Einstein thì Không thể chống lại những người
ngu vì chúng quá đông.</p>
            </div>
        </div>
        <form action="" method="post" onsubmit="return false;">
            <p class="flex chat">
                <input type="text" name="message" value="" placeholder="Aa"
```

```
/>
                <button type="submit">Gửi</button>
            </p>
        </form>
    </div>
    <script>
    /* Gọi sự kiện khi trang đã tải xong */
    document.addEventListener("DOMContentLoaded", function(e) {
        /* Viết sự kiện nhấn nút */

document.querySelector('button[type="submit"]').addEventListener('click',
function(e) {
            e.preventDefault(); /* Dừng không cho submit lên */
            /* Lấy dữ liệu người dùng nhập vào */
            var message =
document.querySelector('input[name="message"]').value.trim();
            /* Kiểm tra xem có dữ liệu không */
            if (message.length == 0) {
                /* Nếu như không nhập gì thì không làm gì cả */
                return false;
            }
            /* Lấy khung hội thoại */
            var conversation = document.querySelector('#conversation');
            /* Xóa nội dung người dùng nhập */
            document.querySelector('input[name="message"]').value = '';
            /* Tạo một thẻ div lưu tin nhắn */
            var item = document.createElement('div');
            item.className = 'item me'; /* Thêm class để cho CSS nhận diện
*/
            item.innerHTML = '<p>'+message+'</p>'; /* Nhúng nội dung vào */
            /* Đưa vào khung hội thoại */
            conversation.appendChild(item);
            /* Cuộn xuống tới tin nhắn */
            conversation.scrollTop = conversation.scrollHeight;
            /* Tạo một cái tự trả lời lại */

            /* Tạo một thẻ div lưu tin nhắn */
            var item = document.createElement('div');
            item.className = 'item'; /* Thêm class để cho CSS nhận diện */
            item.innerHTML = '<p>Chatbot:'+message+'</p>'; /* Nhúng nội
dung vào */
            /* Đưa vào khung hội thoại */
            conversation.appendChild(item);
            /* Cuộn xuống tới tin nhắn */
            conversation.scrollTop = conversation.scrollHeight;
            return false; /* return false dừng không cho submit lên và
ngược lại */
        }, false);
    }, false);
```

```
    </script>
</body>
</html>
```

Như vậy là chúng ta đã làm xong thao tác lấy nội dung người dùng nhập vào và đưa vào hộp thoại Chatbot và chưa làm thao tác gửi lên Chatbot xử lý bằng `python` tiếp theo chúng ta sẽ xây dựng cổng tiếp nhận và gửi dữ liệu lên.

Ở trong phần này chúng ta chỉ đơn giản viết mã bằng `Javascript` không liên quan tới `python` hay `HTML` hoặc `CSS` để tiện cho việc hiểu nguyên lý trong đoạn mã có các đoạn chú thích để dễ hiểu hơn về công năng và nhiệm vụ của từng đoạn mã. Để hiểu rõ về cách lập trình mã này vui lòng xem chi tiết liên kết được đưa ra ở trên.

Xây dựng cổng nhận dữ liệu

Ở phần này chúng ta sẽ tiếp tục chỉnh sửa tập tin `chatbot.py` trả về dữ liệu `JSON` cho mã nguồn `Javascript` tiếp nhận và xử lý thông tin.

Cấu trúc `JSON` sẽ trả về có hai dạng như sau

Nếu có lỗi chúng ta sẽ trả về

```
{
    "error":"Nội dung lỗi"
}
```

Nếu như không có lỗi chúng ta sẽ trả về tin nhắn người dùng và trả lời của Chatbot như sau

```
{
    "message":"Tin nhắn người dùng",
    "reply":"Nội dung trả lời lại"
}
```

Nào chúng ta cùng bắt đầu xây dựng công này với địa chỉ là `/api` nội dung của tập tin `chatbot.py` sẽ như sau vì nội dung tin nhắn có thể sẽ dài vì thế chúng ta chỉ hạn chế cho người dùng gửi lên thông qua phương thức `HTTP POST` và dữ liệu gửi lên có tên là `text`

```
# -*- coding: utf-8 -*-

from flask import Flask
```

```
from flask import render_template, request, jsonify

app = Flask("app", template_folder="./")

app.config["DEBUG"] = True
app.config["JSON_AS_ASCII"] = False
app.config["JSONIFY_PRETTYPRINT_REGULAR"] = False

@app.route('/')
def homepage():
    return render_template('chatbot.html');

@app.route('/api',methods=['POST'])
def chatreply():
    # Nếu gửi lên bằng JSON thì chúng ta sẽ lấy JSON
    if request.content_type and "application/json" in request.content_type:
        data = request.get_json()
    else:
        # Chúng ta sẽ lây thông qua form data
        data = request.form
    # Lấy nội dung người dùng gửi lên
    text = data.get('text','').strip()
    # Nếu như không có gì thì thông báo lỗi và thoát
    if not text:
        return jsonify({
            "error":"Tôi không nghe bạn nói gì cả."
        })
    # Đơn giản ở phần này chúng ta gửi trả lại những gì người ta gửi lên
    return jsonify({
        "message":text,
        "reply":text
    })

if __name__ == "__main__":
    app.run('0.0.0.0', port=9090)
```

Như vậy là chúng ta đã hoàn thành việc tạo cổng nhận dữ liệu người dùng nhập vào cho Chatbot trong phần này chúng ta cần `import` thêm `request` để lấy dữ liệu được gửi lên và `jsonify` để trả về `JSON` cho người dùng.

Vì với những ký tự `utf-8` thì mặc định `Flask` sẽ trả `JSON` sẽ mã hóa `utf-8` để trả nguyên vẹn thì chúng ta sẽ thêm hai dòng cấu hình này

```
app.config["JSON_AS_ASCII"] = False
app.config["JSONIFY_PRETTYPRINT_REGULAR"] = False
```

Lúc này chúng ta gửi dữ liệu là `Chào chatbot` thì trả về cho chúng ta sẽ như sau

```
{
    "message":"Chào chatbot",
    "reply":"Chào chatbot"
}
```

Như vậy là chúng ta đã hoàn thành xong bước này tiếp đến chúng ta sẽ xây dựng mã nguồn `Javascript` để gửi dữ liệu lên cổng này để có dữ liệu trả về.

Xây dựng mã gửi dữ liệu lên và xử lý dữ liệu

Ở phần trên chúng ta đã xây dựng xong cổng tiếp nhận dữ liệu bước này chúng ta sẽ xây dựng các phương thức để tiến hành gửi dữ liệu lên Chatbot, trong `Javascript` có hàm `fetch` chúng ta sẽ sử dụng hàm này để tiến hành gửi dữ liệu lên Chatbot. Chúng ta có hai cách để đưa dữ liệu

1. Đưa vào theo biến `form-data` với header của nó là `'Content-Type': 'application/x-www-form-urlencoded'`
2. Đưa vào theo kiểu `JSON` với header của nó `'Content-Type': 'application/json'`

Để đơn giản chúng ta sẽ chọn cách đưa lên bằng `JSON` với bằng hàm `sendToChatBot` nội dung mã như sau:

```
function sendToChatBot(url, message, callback) {
    callback = callback || function(obj) { console.log(obj); };
    fetch(url, {
        method: "post",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
            text:message /* Dữ liệu sẽ được đưa vào JSON ở đây
JSON.stringify sẽ đưa về thành chuỗi */
        })
    }).then(function(resp) {
        if (resp.status === 200) {
            return resp.json();
        } else {
            console.log("Status: " + resp.status);
            return Promise.reject("server");
        }
    }).then(function(obj) {
        callback(obj);
    }).catch(function(err) {
        callback({
```

```
            "error": "Có lỗi trong quá trình gửi dữ liệu"
        })
    });
}
```

Chúng ta định nghĩa hàm `sendToChatBot(url, message, callback)` với 3 tham trị `url`, `message`, `callback` trong đó

- `url` là địa chỉ của cổng chúng ta muốn gửi dữ liệu tới là kiểu chuỗi (`string`)
- `message` là nội dung tin nhắn của người dùng gửi đi dữ liệu kiểu chuỗi (`string`)
- `callback` là một hàm sẽ được gọi sau khi có kết quả trả về của hàm kiểu là một hàm với tham số là `obj` là một đối tượng JSON chúng ta đã quy định ở phần xây dựng cổng tiếp nhận dữ liệu

Bây giờ chúng ta sẽ đưa nội dung của hàm định nghĩa này vào tập tin `chatbot.html` và viết thêm phần xử lý khi đó chúng ta sẽ có đoạn mã như sau

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Simple Chatbot</title>
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
<style>
* {
    margin:0;
    padding:0;
    box-sizing:border-box;
    outline:none;
}
*:before,*:after{
    -webkit-box-sizing:border-box;
    -moz-box-sizing:border-box;
    box-sizing:border-box;
}
::-webkit-scrollbar{
    width:2px;
    height:2px;
}
::-webkit-scrollbar-track{
    -webkit-box-shadow:inset 0 0 6px rgba(0,0,0,0.3);
    border-radius:1px;
}
::-webkit-scrollbar-thumb{
```

```
    border-radius:1px;
    -webkit-box-shadow:inset 0 0 6px rgba(0,0,0,0.5);
}
body {
    background-color:#f1f1f1;
    font-family:arial,sans-serif;
}
h1, #wrapper {
    max-width:90%;
    width:500px;
    margin:0 auto;
}
h1 {
    margin-top:20px;
    margin-bottom:20px;
}
#wrapper {
    background-color: #fff;
    box-shadow: 0 5px 10px 0 rgb(0 0 0 / 10%);
    -moz-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    -webkit-box-shadow: 0 5px 10px 0 rgb(0 0 0 / 10%);
    -o-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    -ms-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    border-radius:4px;
}
#conversation {
    max-height:250px;
    min-height:250px;
    padding:15px;
    overflow-x: hidden;
    overflow-y: auto;
}
.item {
    margin-bottom: 10px;
    display: flex;
}
.me {
    justify-content: end;
}
.item p {
    padding: 8px;
    background-color: #eee;
    border-radius: 5px;
    max-width: 80%;
    box-shadow: 0 1px 0 0 rgba(0,0,0,0.18);
    font-family:arial,sans-serif;
}
.me p {
    background-color:#e5efff;
```

```
    box-shadow: 0 1px 0 0 #c8deff
}
.flex {
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: nowrap;
}
.chat {
    background-color: #eee;
    border: none;
    border-radius: 0 0 4px 4px;
}
input, button {
    background-color: #eee;
    padding: 6px 12px;
    font-size: 14px;
    color: #555;
    border: none;
    border-radius: 4px 0 0 4px;
    outline: none;
    font-family:arial,sans-serif;
}
input {
    width: 80%;
}
button {
    border-radius: 0 0 4px 0;
    background-color: #337ab7;
    width: 20%;
    cursor: pointer;
    border:none;
    color:#fff;
}
</style>
</head>
<body>
    <h1>Simple Chatbot</h1>
    <div id="wrapper">
        <div id="conversation">
            <div class="item">
                <p>Chúng ta có nên chống lại với người ngu hay không ?</p>
            </div>
            <div class="item me">
                <p>Theo Albert Einstein thì Không thể chống lại những người
ngu vì chúng quá đông.</p>
            </div>
        </div>
        <form action="" method="post" onsubmit="return false;">
```

```
            <p class="flex chat">
                <input type="text" name="message" value="" placeholder="Aa"
/>
                <button type="submit">Gửi</button>
            </p>
        </form>
    </div>
    <script>
    /* Hàm gửi dữ liệu tin nhắn lên chatbot */
    function sendToChatBot(url, message, callback) {
        callback = callback || function(obj) { console.log(obj); };
        fetch(url, {
            method: "post",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({
                text:message /* Dữ liệu sẽ được đưa vào JSON ở đây
JSON.stringify sẽ đưa về thành chuỗi */
            })
        }).then(function(resp) {
            if (resp.status === 200) {
                return resp.json();
            } else {
                console.log("Status: " + resp.status);
                return Promise.reject("server");
            }
        }).then(function(obj) {
            callback(obj);
        }).catch(function(err) {
            callback({
                "error": "Có lỗi trong quá trình gửi dữ liệu"
            })
        });
    }
    /* Gọi sự kiện khi trang đã tải xong */
    document.addEventListener("DOMContentLoaded", function(e) {
        /* Viết sự kiện nhấn nút */

document.querySelector('button[type="submit"]').addEventListener('click',
function(e) {
            e.preventDefault(); /* Dừng không cho submit lên */
            /* Lấy dữ liệu người dùng nhập vào */
            var message =
document.querySelector('input[name="message"]').value.trim();
            /* Kiểm tra xem có dữ liệu không */
            if (message.length == 0) {
                /* Nếu như không nhập gì thì không làm gì cả */
                return false;
            }
            /* Lấy khung hội thoại */
```

```
            var conversation = document.querySelector('#conversation');
            /* Xóa nội dung người dùng nhập */
            document.querySelector('input[name="message"]').value = '';
            /* Tạo một thẻ div lưu tin nhắn */
            var item = document.createElement('div');
            item.className = 'item me'; /* Thêm class để cho CSS nhận diện
*/
            item.innerHTML = '<p>'+message+'</p>'; /* Nhúng nội dung vào */
            /* Đưa vào khung hội thoại */
            conversation.appendChild(item);
            /* Cuộn xuống tới tin nhắn */
            conversation.scrollTop = conversation.scrollHeight;
            /* Tiến hành gửi dữ liệu lên và xử lý dữ liệu trả về */
            sendToChatBot('{{ url_for(".chatreply") }}', message,
function(obj) {
                if (obj.error != undefined) {
                    /* Tạo một thẻ div lưu tin nhắn */
                    var item = document.createElement('div');
                    item.className = 'item'; /* Thêm class để cho CSS nhận
diện */
                    item.innerHTML = '<p>Chatbot:'+obj.error+'</p>'; /*
Nhúng nội dung lỗi vào */
                    /* Đưa vào khung hội thoại */
                    conversation.appendChild(item);
                    /* Cuộn xuống tới tin nhắn */
                    conversation.scrollTop = conversation.scrollHeight;
                } else {
                    /* Tạo một thẻ div lưu tin nhắn */
                    var item = document.createElement('div');
                    item.className = 'item'; /* Thêm class để cho CSS nhận
diện */
                    item.innerHTML = '<p>Chatbot:'+obj.reply+'</p>'; /*
Nhúng nội dung lỗi vào */
                    /* Đưa vào khung hội thoại */
                    conversation.appendChild(item);
                    /* Cuộn xuống tới tin nhắn */
                    conversation.scrollTop = conversation.scrollHeight;
                }
            });
            return false; /* return false dừng không cho submit lên và
ngược lại */
        }, false);
    }, false);
    </script>
</body>
</html>
```

Trong thay đổi này chúng ta có một lưu ý `'{{ url_for(".chatreply") }}'` đây là một hàm của `Template Engine` của `Flask` nó sẽ tự động tạo đường dẫn nếu như hàm trong tồn tại ở đây là hàm `chatreply` chúng ta đã định nghĩa ở trong tập tin `chatbot.py` để biết thêm chi tiết về `Temlate Engine` của `Flask` chúng ta tham khảo Templates

Và đoạn mã mới được thêm cùng với hàm `sendToChatBot` như sau

```
sendToChatBot('{{ url_for(".chatreply") }}', message, function(obj) {
    if (obj.error != undefined) {
        /* Tạo một thẻ div lưu tin nhắn */
        var item = document.createElement('div');
        item.className = 'item'; /* Thêm class để cho CSS nhận diện */
        item.innerHTML = '<p>Chatbot:'+obj.error+'</p>'; /* Nhúng nội dung
lỗi vào */
        /* Đưa vào khung hội thoại */
        conversation.appendChild(item);
        /* Cuộn xuống tới tin nhắn */
        conversation.scrollTop = conversation.scrollHeight;
    } else {
        /* Tạo một thẻ div lưu tin nhắn */
        var item = document.createElement('div');
        item.className = 'item'; /* Thêm class để cho CSS nhận diện */
        item.innerHTML = '<p>Chatbot:'+obj.reply+'</p>'; /* Nhúng nội dung
lỗi vào */
        /* Đưa vào khung hội thoại */
        conversation.appendChild(item);
        /* Cuộn xuống tới tin nhắn */
        conversation.scrollTop = conversation.scrollHeight;
    }
});
```

Cấu trúc xử lý như trước chỉ khác chỗ `..., function(obj) {...});` đoạn này chính là cái hàm `callback` với `obj` sẽ được truyền khi hàm `sendToChatBot` xử lý xong dữ liệu.

Như vậy ở đây chúng ta đã hoàn thành một Chatbot đơn giản để có thể đưa Chatbot này vào ứng dụng thực tế có trí tuệ nhân tạo thì chúng ta sẽ bắt đầu triển khai các đoạn mã vào phần thân của hàm `chatreply` trong tập tin `chatbot.py` tiếp theo chúng ta sẽ bắt đầu xử lý dữ liệu chuỗi (`string`) để chuyển đổi đưa vào máy học và đưa ra kết quả theo như mong muốn của người dùng.

Áp dụng mô hình máy học vào việc xử lý cho Chatbot

Để Chatbot trả lời chính xác hơn thì chúng ta phải xử lý tách từ, phân tích từ để tìm đúng chủ đề người dùng đang muốn nói... vì vậy việc này cần phải rất nhiều thời gian để tìm kiếm cũng như tham khảo từ các bài báo khoa học để đưa ra ý tưởng và thuật giải cho nó. Để tiện cho việc học tập tìm hiểu mức cơ bản hiểu được máy học như thế nào cho dễ dàng hơn trong con đường tự học hay tự nghiên cứu thì chúng ta tập tham khảo với bộ thư viện `sklearn` một thư viện có thể nói nó hoạt động khá tốt trên mọi máy tính cá nhân cũng như độ phổ biến các thuật toán của các cộng đồng lập trình viên đóng góp vào cho nó để cài đặt bộ thư viện này chúng ta dùng lệnh trên `cmd` của Windows hoặc `terminal` trên hệ điều hành mã nguồn mở như sau

```
pip3 install scikit-learn
```

Sau khi cài đặt thư viện này chúng ta chỉ sử dụng đối tượng `TfidfVectorizer` và hàm `cosine_similarity` để đơn giản cho việc học của chúng ta, trong đó `TfidfVectorizer` là TF-IDF vả `cosine_similarity` là dùng để đo độ tương tự cosin của chuỗi

Để đơn giản chúng ta tạm thời xây dựng một tập câu trả lời dưới dạng `text` lưu với tập tin `brain.txt` như sau:

```
Trong truy hồi thông tin, tf–idf, TF*IDF, hay TFIDF, viết tắt từ cụm từ
tiếng Anh: term frequency–inverse document frequency, là một thống kê số
học nhằm phản ánh tầm quan trọng của một từ đối với một văn bản trong một
tập hợp hay một ngữ liệu văn bản. tf–idf thường dùng dưới dạng là một trọng
số trong tìm kiếm truy xuất thông tin, khai thác văn bản, và mô hình hóa
người dùng.

Giá trị tf–idf tăng tỉ lệ thuận với số lần xuất hiện của một từ trong tài
liệu và được bù đắp bởi số lượng tài liệu trong kho ngữ liệu có chứa từ,
giúp điều chỉnh thực tế là một số từ xuất hiện nói chung thường xuyên hơn.
tf-idf là một trong những lược đồ (scheme) tính trọng số phổ biến nhất hiện
nay. Một cuộc khảo sát được thực hiện vào năm 2015 cho thấy 83% các hệ
thống
khuyến nghị dựa trên văn bản (text-based recommender systems) trong các thư
viện số sử dụng tf-idf.

IDF có ứng dụng trong máy tìm kiếm. Ví dụ, khi người dùng gửi một truy vấn
đến máy tìm kiếm, hệ thống cần biết từ nào là từ người dùng quan tâm nhất.
Chẳng hạn: truy vấn của người dùng là "làm thế nào để sửa máy ủi". Sau khi
tách từ, chúng ta sẽ có tập các từ: làm, thế nào, để, sửa, máy ủi. Trong
các từ này, "máy ủi" sẽ có IDF cao nhất. Hệ thống sẽ lấy ra tất cả các văn
bản có chứa từ máy ủi và sau đó mới thực hiện việc đánh giá và so sánh dựa
trên toàn bộ câu truy vấn.

Độ tương tự cosin là một cách đo độ tương tự (measure of similarity) giữa
```

```
hai vectơ khác không của một không gian tích vô hướng. Độ tương tự này được
định nghĩa bằng giá trị cosine của góc giữa hai vectơ, và cũng là tích vô
hướng của cùng các vectơ đơn vị để cả hai đều có chiều dài 1. Giá trị
cosine
của 0° là 1, và bé hơn 1 với bất kỳ góc nào trong khoảng các radian (0, π].

Độ tương tự cosin là một thẩm định có tính định hướng chứ không phải về độ
lớn (to nhỏ): hai vectơ cùng hướng có độ tương tự cosin là 1, hai vectơ
vuông
góc nhau (hay có hướng 90°) có độ tương tự là 0, và hai vectơ đối nhau theo
đường kính có độ tương tự (hay 180°) là -1. Độ tương tự cosin đặc biệt được
sử dụng trong không gian dương với kết quả được giới hạn chặt chẽ trong
biên
độ [0,1]. Cái tên "độ tương tự cosin" bắt nguồn từ thuật ngữ "cosin có
hướng":
trong trường hợp này, các vectơ đơn vị có độ "tương tự" tối đa nếu chúng
song song và "khác nhau" cực đại nếu chúng là trực giao (vuông góc). Điều
này tương tự với cosin, có giá trị lớn nhất khi các phân đoạn tạo thành một
góc bằng 0, và giá trị bằng 0 (không liên quan) khi các đoạn thẳng vuông
góc.

Độ tương tự có giá trị −1 có nghĩa là trái nghĩa hoàn toàn, với giá trị 1
nghĩa là giống nhau hoàn toàn, với 0 có nghĩa là trực giao hay tương quan
(decorrelation), trong khi các giá trị ở giữa biểu thị sự giống nhau hoặc
không giống nhau ở mức trung gian.
```

Chúng ta sẽ nạp nội dung của tập tin `brain.txt` vào Chatbot như sau

```
sentences = []
try:
    with open('brain.txt','r',encoding='utf-8') as fin:
        text = []
        for line in fin:
            line = line.strip()
            if not line:
                sentences.append(" ".join(text))
                text = []
                continue
            text.append(line)
except:
    pass
```

Ở phần này chúng ta đã nạp câu trả lời vào một biết là `sentences` để chúng ta sẽ sử dụng sau này.

Tiếp đến chúng ta sẽ viết vài hàm cơ bản để xử lý.

1. Chào hỏi

```
def chaohoi(text):
    if not text.strip():
        return None
    chao_hoi = ["chào","hi","halo","bonjour","hello"]
    dap_lai = ["Vâng xin chào","Rất vui khi được gặp bạn","Tôi có thể giúp
được gì không"]
    for word in text.split(' '):
        if word.lower() in chao_hoi:
            return random.choice(dap_lai)
    return None
```

Trong hàm này chỉ là vài mẫu đơn giản, chúng ta có thể định nghĩa thêm cho nó vì đây là ứng dụng cơ bản nên chúng ta phải phụ trợ vào cho nó. Ở hàm này nếu như có tồn tại những âm tiết chào hỏi thì sẽ trả về ngẫu nhiên một câu trả lời cho người dùng, nếu không có thì sẽ trả về `None` để chúng ta sẽ gọi hàm khác xử lý việc này.

2. Cảm ơn

```
def camon(text):
    if not text.strip():
        return None
    cam_on = ["cảm ơn","thank","thanks"]
    dap_lai = ["Vâng không có gì","Rất vui khi được giúp bạn","Tôi rất hân
hạnh khi có thể giúp được bạn"]
    for word in text.split(' '):
        if word.lower() in cam_on:
            return random.choice(dap_lai)
    return None
```

Hàm này cũng gần giống chức năng với `chaohoi` nếu coi `chào hỏi` là một chủ đề thì `cám ơn` cũng là một chủ để đây là một ví dụ có thể giúp chúng ta hiểu được cách phân tách chủ đề trong nói chuyện.

3. Tạm biệt

```
def tambiet(text):
    if not text.strip():
        return None
    tam_biet = ["chào nhé","bye","tạm biệt"]
    dap_lai = ["Vâng xin chào ạ","Vâng tạm biết hi vọng cuộc nói chuyện vui
vẻ"]
    for word in text.split(' '):
```

```
            if word.lower() in tam_biet:
                return random.choice(dap_lai)
    return None
```

Hàm này cũng gần giống chức năng với `chaohoi` và `camon` là một chủ đề nói chuyện

4. Tìm câu trả lời dùng độ tương đồng `cosin` và tần suất

```
# Tách các câu trả lời ra thành các câu nhỏ hơn.
sent_tokens = set()
for sent in sentences:
    sents = [s.strip() for s in
re.compile(r'[.?!]\s+').split(sent.strip()) if s.strip() ]
    for s in sents:
        s = s.lower()
        s = " ".join([w for w in s.split() if w.strip() and w not in
string.punctuation ])
        if not s.strip():
            continue
        if s not in sent_tokens:
            sent_tokens.add(s)

def tokenize(text):
    return re.compile(r'\s+').split(text.strip())

def chatbot_reply(text):
    tokens = list(sent_tokens)
    if len(tokens) == 0:
        return 'Xin lỗi tôi chưa hiểu ý bạn muốn nói, tôi sẽ học lại sau'
    tokens.append(text.lower())
    TfidfVec = TfidfVectorizer(tokenizer=tokenize,stop_words=None)
    tfidf = TfidfVec.fit_transform(tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx = vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if req_tfidf == 0:
        return 'Xin lỗi tôi chưa hiểu ý bạn muốn nói, tôi sẽ học lại sau'
    return tokens[idx]
```

Trong đoạn trên chúng ta sử dụng cắt các câu lớn thành các câu nhỏ hơn, tiến hành khai báo một hàm tạo `tokens` đơn giản tách các từ để tính toán và một hàm lấy câu trả lời từ câu người dùng đưa vào và dữ liệu từ các câu được cắt nhỏ dùng `TF-IDF` để tính tần suất và dùng hàm `cosin` để tiệm cận đáp án.

Như vậy là các ý tưởng đã hoàn thành chúng ta sẽ ráp mã nguồn vào tập tin `chatbot.py` như sau

```
# -*- coding: utf-8 -*-

from flask import Flask
from flask import render_template, request, jsonify
import re
import string
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

app = Flask("app", template_folder="./")

app.config["DEBUG"] = True
app.config["JSON_AS_ASCII"] = False
app.config["JSONIFY_PRETTYPRINT_REGULAR"] = False

sentences = []
try:
    with open('brain.txt','r',encoding='utf-8') as fin:
        text = []
        for line in fin:
            line = line.strip()
            if not line:
                sentences.append(" ".join(text))
                text = []
                continue
            text.append(line)
except:
    pass

# Tách các câu trả lời ra thành các câu nhỏ hơn.
sent_tokens = set()
for sent in sentences:
    sents = [s.strip() for s in
re.compile(r'[.?!]\s+').split(sent.strip()) if s.strip() ]
    for s in sents:
        s = s.lower()
        # Loại bỏ ký tự đặc biệt trong câu
        s = " ".join([w for w in s.split() if w.strip() and w not in
string.punctuation ])
        if not s.strip():
            continue
        if s not in sent_tokens:
            sent_tokens.add(s)
```

```
# Khai báo hàm tách từ
def tokenize(text):
    return re.compile(r'\s+').split(text.strip())

# Khai báo hàm lấy câu trả lời
def chatbot_reply(text):
    tokens = list(sent_tokens)
    if len(tokens) == 0:
        return 'Xin lỗi tôi chưa hiểu ý bạn muốn nói, tôi sẽ học lại sau'
    tokens.append(text.lower())
    TfidfVec = TfidfVectorizer(tokenizer=tokenize,stop_words=None)
    tfidf = TfidfVec.fit_transform(tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx = vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if req_tfidf == 0:
        return 'Xin lỗi tôi chưa hiểu ý bạn muốn nói, tôi sẽ học lại sau'
    return tokens[idx]

# Kiểm tra người dùng có chào hỏi hay không
def chaohoi(text):
    if not text.strip():
        return None
    chao_hoi = ["chào","hi","halo","bonjour","hello"]
    dap_lai = ["Vâng xin chào","Rất vui khi được gặp bạn","Tôi có thể giúp
được gì không"]
    for word in text.split(' '):
        if word.lower() in chao_hoi:
            return random.choice(dap_lai)
    return None

# Kiểm tra người dùng có cám ơn hay không
def camon(text):
    if not text.strip():
        return None
    cam_on = ["cảm ơn","thank","thanks"]
    dap_lai = ["Vâng không có gì","Rất vui khi được giúp bạn","Tôi rất hân
hạnh khi có thể giúp được bạn"]
    for word in text.split(' '):
        if word.lower() in cam_on:
            return random.choice(dap_lai)
    return None

# Kiểm tra xem người dùng có tạm biệt
def tambiet(text):
    if not text.strip():
        return None
```

```
    tam_biet = ["chào nhé","bye","tạm biệt"]
    dap_lai = ["Vâng xin chào ạ","Vâng tạm biết hi vọng cuộc nói chuyện vui
vẻ"]
    for word in text.split(' '):
        if word.lower() in tam_biet:
            return random.choice(dap_lai)
    return None

@app.route('/')
def homepage():
    return render_template('chatbot.html');

@app.route('/api',methods=['POST'])
def chatreply():
    # Nếu gửi lên bằng JSON thì chúng ta sẽ lấy JSON
    if request.content_type and "application/json" in request.content_type:
        data = request.get_json()
    else:
        # Chúng ta sẽ lây thông qua form data
        data = request.form
    # Lấy nội dung người dùng gửi lên
    text = data.get('text','').strip()
    # Nếu như không có gì thì thông báo lỗi và thoát
    if not text:
        return jsonify({
            "error":"Tôi không nghe bạn nói gì cả."
        })
    # Chúng ta sẽ xử lý câu trả lời lại cho chatbot
    reply = chaohoi(text) # Xem có chào không
    if reply is None:
        reply = tambiet(text) # Không chào xem có tạm biệt không

    if reply is None:
        reply = camon(text) # Xem thử có cám ơn không

    if reply is None:
        reply = chatbot_reply(text) # Cuối cùng là tìm câu trả lời

    return jsonify({
        "message":text,
        "reply":reply
    })

if __name__ == "__main__":
    app.run('0.0.0.0', port=9090)
```

Trong đoạn mã này có thêm vào như sau

```
import re
import string
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

- `re` dùng để cắt chuỗi thành các câu nhỏ hơn và cắt câu thành các từ riêng biết
- `string` để lấy danh sách các ký tự đặc biệt
- `from sklearn.feature_extraction.text import TfidfVectorizer` để thêm vào đối tượng `TF-IDF`
- `from sklearn.metrics.pairwise import cosine_similarity` dùng để thêm vào hàm tính độ tương đối `cosin`

Và đoạn

```
# Chúng ta sẽ xử lý câu trả lời lại cho chatbot
    reply = chaohoi(text) # Xem có chào không
    if reply is None:
        reply = tambiet(text) # Không chào xem có tạm biệt không

    if reply is None:
        reply = camon(text) # Xem thử có cám ơn không

    if reply is None:
        reply = chatbot_reply(text) # Cuối cùng là tìm câu trả lời
```

Đoạn này dùng để tìm câu trả lời để trả về cho người dùng.

Trong tập tin `chatbot.html` chúng ta thêm đoạn `Xin chào tôi có thể giúp được gì ạ?` cho mỗi lần tải trang sẽ có câu thông báo này.

Mã nguồn của tập tin `chatbot.html` như sau

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Simple Chatbot</title>
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">
<style>
* {
```

```
    margin:0;
    padding:0;
    box-sizing:border-box;
    outline:none;
}
*:before,*:after{
    -webkit-box-sizing:border-box;
    -moz-box-sizing:border-box;
    box-sizing:border-box;
}
::-webkit-scrollbar{
    width:2px;
    height:2px;
}
::-webkit-scrollbar-track{
    -webkit-box-shadow:inset 0 0 6px rgba(0,0,0,0.3);
    border-radius:1px;
}
::-webkit-scrollbar-thumb{
    border-radius:1px;
    -webkit-box-shadow:inset 0 0 6px rgba(0,0,0,0.5);
}
body {
    background-color:#f1f1f1;
    font-family:arial,sans-serif;
}
h1, #wrapper {
    max-width:90%;
    width:500px;
    margin:0 auto;
}
h1 {
    margin-top:20px;
    margin-bottom:20px;
}
#wrapper {
    background-color: #fff;
    box-shadow: 0 5px 10px 0 rgb(0 0 0 / 10%);
    -moz-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    -webkit-box-shadow: 0 5px 10px 0 rgb(0 0 0 / 10%);
    -o-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    -ms-box-shadow: 0 5px 10px 0 rgba(0,0,0,.1);
    border-radius:4px;
}
#conversation {
    max-height:250px;
    min-height:250px;
    padding:15px;
    overflow-x: hidden;
```

```
    overflow-y: auto;
}
.item {
    margin-bottom: 10px;
    display: flex;
}
.me {
    justify-content: end;
}
.item p {
    padding: 8px;
    background-color: #eee;
    border-radius: 5px;
    max-width: 80%;
    box-shadow: 0 1px 0 0 rgba(0,0,0,0.18);
    font-family:arial,sans-serif;
}
.me p {
    background-color:#e5efff;
    box-shadow: 0 1px 0 0 #c8deff
}
.flex {
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: nowrap;
}
.chat {
    background-color: #eee;
    border: none;
    border-radius: 0 0 4px 4px;
}
input, button {
    background-color: #eee;
    padding: 6px 12px;
    font-size: 14px;
    color: #555;
    border: none;
    border-radius: 4px 0 0 4px;
    outline: none;
    font-family:arial,sans-serif;
}
input {
    width: 80%;
}
button {
    border-radius: 0 0 4px 0;
    background-color: #337ab7;
    width: 20%;
```

```
    cursor: pointer;
    border:none;
    color:#fff;
}
</style>
</head>
<body>
    <h1>Simple Chatbot</h1>
    <div id="wrapper">
        <div id="conversation">
            <div class="item">
                <p>Xin chào tôi có thể giúp được gì ạ?</p>
            </div>
        </div>
        <form action="" method="post" onsubmit="return false;">
            <p class="flex chat">
                <input type="text" name="message" value="" placeholder="Aa"
/>
                <button type="submit">Gửi</button>
            </p>
        </form>
    </div>
    <script>
    /* Hàm gửi dữ liệu tin nhắn lên chatbot */
    function sendToChatBot(url, message, callback) {
        callback = callback || function(obj) { console.log(obj); };
        fetch(url, {
            method: "post",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({
                text:message /* Dữ liệu sẽ được đưa vào JSON ở đây
JSON.stringify sẽ đưa về thành chuỗi */
            })
        }).then(function(resp) {
            if (resp.status === 200) {
                return resp.json();
            } else {
                console.log("Status: " + resp.status);
                return Promise.reject("server");
            }
        }).then(function(obj) {
            callback(obj);
        }).catch(function(err) {
            callback({
                "error": "Có lỗi trong quá trình gửi dữ liệu"
            })
        });
    }
    /* Gọi sự kiện khi trang đã tải xong */
```

```
    document.addEventListener("DOMContentLoaded", function(e) {
        /* Viết sự kiện nhấn nút */

document.querySelector('button[type="submit"]').addEventListener('click',
function(e) {
            e.preventDefault(); /* Dừng không cho submit lên */
            /* Lấy dữ liệu người dùng nhập vào */
            var message =
document.querySelector('input[name="message"]').value.trim();
            /* Kiểm tra xem có dữ liệu không */
            if (message.length == 0) {
                /* Nếu như không nhập gì thì không làm gì cả */
                return false;
            }
            /* Lấy khung hội thoại */
            var conversation = document.querySelector('#conversation');
            /* Xóa nội dung người dùng nhập */
            document.querySelector('input[name="message"]').value = '';
            /* Tạo một thẻ div lưu tin nhắn */
            var item = document.createElement('div');
            item.className = 'item me'; /* Thêm class để cho CSS nhận diện
*/
            item.innerHTML = '<p>'+message+'</p>'; /* Nhúng nội dung vào */
            /* Đưa vào khung hội thoại */
            conversation.appendChild(item);
            /* Cuộn xuống tới tin nhắn */
            conversation.scrollTop = conversation.scrollHeight;
            /* Tiến hành gửi dữ liệu lên và xử lý dữ liệu trả về */
            sendToChatBot('{{ url_for(".chatreply") }}', message,
function(obj) {
                if (obj.error != undefined) {
                    /* Tạo một thẻ div lưu tin nhắn */
                    var item = document.createElement('div');
                    item.className = 'item'; /* Thêm class để cho CSS nhận
diện */
                    item.innerHTML = '<p>Chatbot:'+obj.error+'</p>'; /*
Nhúng nội dung lỗi vào */
                    /* Đưa vào khung hội thoại */
                    conversation.appendChild(item);
                    /* Cuộn xuống tới tin nhắn */
                    conversation.scrollTop = conversation.scrollHeight;
                } else {
                    /* Tạo một thẻ div lưu tin nhắn */
                    var item = document.createElement('div');
                    item.className = 'item'; /* Thêm class để cho CSS nhận
diện */
                    item.innerHTML = '<p>Chatbot:'+obj.reply+'</p>'; /*
Nhúng nội dung lỗi vào */
                    /* Đưa vào khung hội thoại */
```

```
                    conversation.appendChild(item);
                    /* Cuộn xuống tới tin nhắn */
                    conversation.scrollTop = conversation.scrollHeight;
                }
            });
            return false; /* return false dừng không cho submit lên và
ngược lại */
        }, false);
    }, false);
    </script>
</body>
</html>
```

Như vậy chúng ta đã hoàn thiện một Chatbot đơn giản có ứng dụng thuật toán máy học để tạo ra một ứng dụng cho riêng chúng ta. Để có thêm những sản phẩm tuyệt vời hi vọng bạn có thể tìm hiểu được ý tưởng cơ bản từ ứng dụng Chatbot cơ bản này

Chúc bạn thành công !