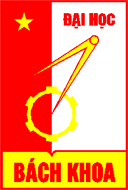
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ Thông tin và Truyền thông

****

**BÁO CÁO HỌC PHẦN PROJECT III**

**Đề tài:**

**Xây dựng hệ thống Chatbot tự động**

**với nền tảng Facbook Messenger hỗ trợ bán hàng**

|  |  |
| --- | --- |
| Giáo viên hướng dẫn: | T.SNguyễn Đình Thuận |
| Sinh viên thực hiện: | Trần Trung Đức  MSSV: 20173032  Mã lớp: 699564 |
| Bộ môn: | Kỹ thuật máy tính |

*Hà Nội, tháng 01 năm 2021*

**MỤC LỤC**

[CHƯƠNG 1. TÌM HIỂU BÀI TOÁN 2](#_Toc62754440)

[1.1. Messenger Platform 2](#_Toc62754441)

[1.1.1. Cách thức hoạt động 2](#_Toc62754442)

[1.1.2. Webhooks 2](#_Toc62754443)

[1.2. Ngrok 2](#_Toc62754444)

[1.3. MQTT 3](#_Toc62754445)

[1.4. NLP và bài toán phân loại văn bản 5](#_Toc62754446)

[CHƯƠNG 2. NỘI DUNG 6](#_Toc62754447)

[2.1. Các bước thực hiện bài toán 6](#_Toc62754448)

[2.2. Cấu trúc chương trình 6](#_Toc62754449)

[2.2.1. Biểu đồ ca sử dụng 6](#_Toc62754450)

[2.2.2. Biểu đồ trình tự 6](#_Toc62754451)

[2.3. Cài đặt chương trình 7](#_Toc62754452)

[2.3.1. Đăng ký facebook developer 7](#_Toc62754453)

[2.3.2. Cài đặt Webooks 10](#_Toc62754454)

[2.3.3. Xây dựng model phân loại câu hỏi bằng Machine Learnning 13](#_Toc62754455)

[2.3.4. Xây dựng giao thức truyền thông điệp MQTT 15](#_Toc62754456)

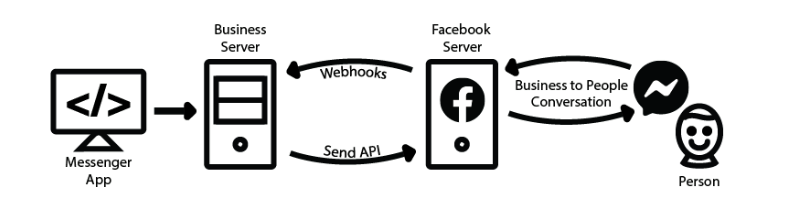
[CHƯƠNG 3. Đánh giá kết quả 18](#_Toc62754457)

[Tài liệu tham khảo 19](#_Toc62754458)

# TÌM HIỂU BÀI TOÁN

## Messenger Platform

### Cách thức hoạt động



Hình 1.1 Nền tảng Facebook Messenger

Trong mô hình nghiệp vụ bán hàng, khi tiến hành cài đặt phần mềm từ bên thứ ba cho những cuộc đối thoại tự động, luồng dự liệu sẽ hoạt động như trên hình 1.1.

Đầu tiên, Facebook sẽ tiến hành gửi thông điệp webhooks cho máy chủ của doanh nghiệp, nơi phần mềm hỗ trợ được cài đặt. Tại đây, phần mềm sẽ xử lý và đưa ra phản hồi tương ứng đến cho Facebook Server qua API được cung cấp. Khi đó, Facebook sẽ phản hồi tin nhắn lại cho người dùng mà ở đây là khách hàng của doanh nghiệp.

### Webhooks

Là một khái niệm ra đời năm 2007, mang đến một cách hữu ích và tương đối dễ dàng, gọn nhẹ trong việc triển khai các phản ứng sự kiện.

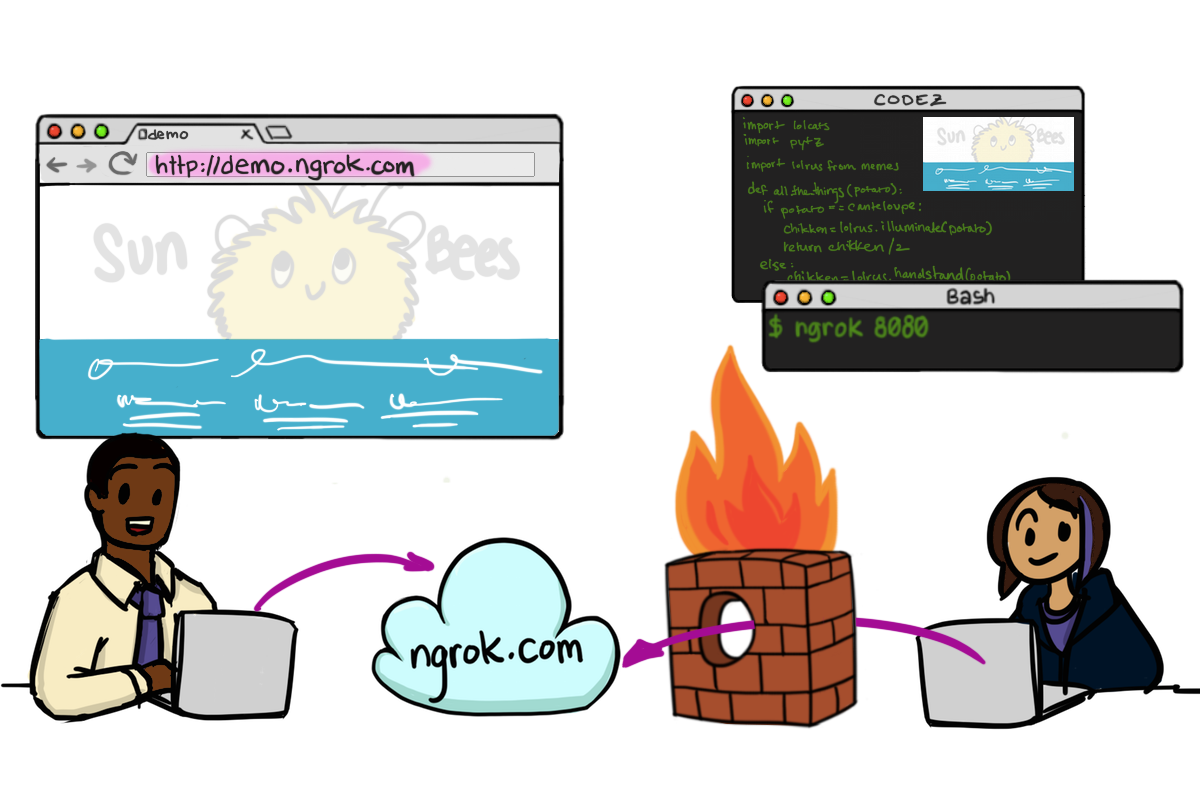
Webhooks đôi khi còn được gọi là "Reverse APIs". Trong các API, ứng dụng client-side sẽ gọi (tiêu thụ) ứng dụng server-side. Trong khi đó, khi có web hook, phía server-side sẽ gọi web hook (end-point URL được cung cấp bởi ứng dụng client-side), ví dụ: ứng dụng server-side gọi ứng dụng client-side [1].

Ưu điểm: Webhooks cho phép người dùng không phải thường xuyên thực hiện các yêu cầu API (API request) để kiểm tra trạng thái của server. Thay vào đó, khi có sự thay đổi hoặc một sự kiện xảy ra, server sẽ gọi đến người dùng qua API để thông báo sự thay đổi đó. => Giảm thiểu được sử dụng băng thông và tải cho server.

Nhược điểm: Khó khăn trong thiết lập ban đầu.

## Ngrok

Là công cụ tạo đường hầm (tunnel) giữa localhost và internet thông qua việc truy cập đến domain đặc biệt của ngrok.

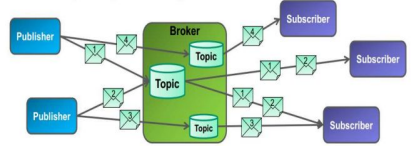


Cách thức hoạt động:

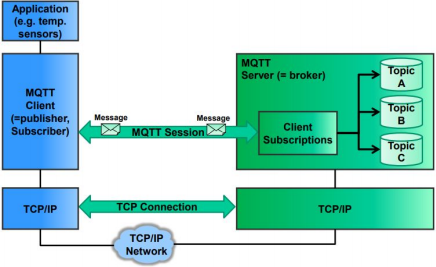
* ngrok tạo kết nối với localhost qua 1 cổng nhất định và tạo 1 url tương ứng với cổng đã được kết nối đó. Khi đó, để ta có thể kết nối với cổng này của localhost qua việc kết nối với trang web public được tạo của ngrok.

## MQTT

* MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp / thuê bao), được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Nó dựa trên một Broker (tạm dịch là “Máy chủ môi giới”) “nhẹ” (khá ít xử lý) và được thiết kế có tính mở (tức là không đặc trưng cho ứng dụng cụ thể nào), đơn giản và dễ cài đặt.NLP model
* Tính năng, đặc điểm nổi bật
  + Dạng truyền thông điệp theo mô hình Pub/Sub cung cấp việc truyền tin phân tán một chiều, tách biệt với phần ứng dụng.
  + Việc truyền thông điệp là ngay lập tức, không quan tâm đến nội dung được truyền.
  + Sử dụng TCP/IP là giao thức nền.
  + Tồn tại ba mức độ tin cậy cho việc truyền dữ liệu (QoS: Quality of service)
    - QoS 0: Broker/client sẽ gửi dữ liệu đúng một lần, quá trình gửi được xác nhận bởi chỉ giao thức TCP/IP.
    - QoS 1: Broker/client sẽ gửi dữ liệu với ít nhất một lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
    - QoS 2: Broker/client đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng một lần, quá trình này phải trải qua 4 bước bắt tay.
  + Phần bao bọc dữ liệu truyền nhỏ và được giảm đến mức tối thiểu để giảm tải cho đường truyền.
* Cơ chế hoạt động của MQTT theo mô hình Pub/Sub



* + MQTT hoạt động theo cơ chế client/server, nơi mà mỗi cảm biến là một khách hàng (client) và kết nối đến một máy chủ, có thể hiểu như một Máy chủ môi giới (broker), thông qua giao thức TCP (Transmission Control Protocol). Broker chịu trách nhiệm điều phối tất cả các thông điệp giữa phía gửi đến đúng phía nhận.
  + MQTT là giao thức định hướng bản tin. Mỗi bản tin là một đoạn rời rạc của tín hiệu và broker không thể nhìn thấy. Mỗi bản tin được publish một địa chỉ, có thể hiểu như một kênh (Topic). Client đăng kí vào một vài kênh để nhận/gửi dữ liệu, gọi là subscribe. Client có thể subscribe vào nhiều kênh. Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gửi dữ liệu vào kênh đã đăng ký. Khi một client gửi một bản tin đến một kênh nào đó gọi là publish.
* Kiến trúc thành phần của MQTT



* + Thành phần chính của MQTT là Client (Publisher/Subscriber), Server (Broker), Sessions (tạm dịch là Phiên làm việc), Subscriptions và Topics.
  + MQTT Client (Publisher/Subscriber): Clients sẽ subscribe một hoặc nhiều topics để gửi và nhận thông điệp từ những topic tương ứng.
  + MQTT Server (Broker): Broker nhận những thông tin subscribe (Subscriptions) từ client, nhận thông điệp, chuyển những thông điệp đến các Subscriber tương ứng dựa trên Subscriptions từ client.
  + Topic: Có thể coi Topic là một hàng đợi các thông điệp, và có sẵn khuôn mẫu dành cho Subscriber hoặc Publisher. Một cách logic thì các topic cho phép Client trao đổi thông tin với những ngữ nghĩa đã được định nghĩa sẵn. Ví dụ: Dữ liệu cảm biến nhiệt độ của một tòa nhà.
  + Session: Một session được định nghĩa là kết nối từ client đến server. Tất cả các giao tiếp giữa client và server đều là 1 phần của session.
  + Subscription: Không giống như session, subscription về mặt logic là kết nối từ client đến topic. Khi đã subscribe một topic, Client có thể nhận/gửi thông điệp (message) với topic đó.
* Trong đồ án này, MQTT sẽ là giao thức truyền thông điệp giữa Webhook và mô hình NLP đã tạo để phân loại câu hỏi.

## NLP và bài toán phân loại văn bản

* NLP (Natural Language Processing) là một trong những nhánh khó của trí tuệ nhân tạo. Bởi lẽ ngôn ngữ là một hệ thống phức tạp để giao tiếp giữa những động vật bậc cao hay có năng lực tư duy như con người. Nếu NLP được giải quyết thành công đồng nghĩa với việc máy tính có thể hiểu và sử dụng ngôn ngữ tự nhiên để giao tiếp như chúng ta.
* Đối với đề tài này, mô hình em sử dụng thuộc vào dạng bài toán phân loại trong xử lý ngôn ngữ tự nhiên. Trước mỗi câu hỏi của người dùng, model NLP sẽ dự đoán mục đích câu họi người dùng là gì (hỏi về giá sản phẩm, hỏi về trạng thái sản phẩm, …), làm tiền đề cho việc phát triển hệ thống trả lời tự động sau này.
  + Phân loại văn bản (text classification) là bài toán thuộc nhóm học có giám sát (Supervised learninng) trong học máy. Bài toán này yêu cầu dữ liệu có nhãn, mô hình sẽ học từ dữ liệu có nhãn đó, sau đó mô hình được dùng để dự đoán nhãn cho các dữ liệu mới mà mô hình chưa gặp.
  + Để xây dựng mô hình phân loại văn bản, có 2 hướng em tiếp cận, đó là sử dụng các thuật toán Machine Learning truyền thống và sử dụng mô hình Deeplearning.
  + Tuy nhiên, việc sử dụng DeepLearning yêu cầu bộ dữ liệu huấn luyện phải đủ lớn, đi kèm đó là có thể sẽ cần một hạ tầng phần cứng đủ mạnh để đáp ứng thời gian đủ nhanh. Vì vậy trong phạm vi đồ án, em sẽ áp dụng các thuật toán Machine Learning truyền thống như: Naive Bayes, Logistic Regression hay SVM.
  + Với các mô hình truyền thống, xác suất dự đoán đúng là không quá cao, ở mức chấp nhận được. Tuy nhiên việc dự đoán diễn ra nhanh, yêu cầu bộ dữ liệu huấn luyện không quá lớn.

# NỘI DUNG

## Các bước thực hiện bài toán

* Tìm hiểu về các công nghệ sử dụng.
* Thực hiện kết nối giữa ứng dụng với trang web trên facebook.
* Tích hợp NLP cho chatbot.
* Kết nối, giao tiếp giữa NLP và Webhook sử dụng maqiatto
* Cải thiện hiệu năng và khắc phục lỗi trong ứng dụng.

## Cấu trúc chương trình

### Biểu đồ ca sử dụng

Diagram

Description automatically generated

Hình 2.1 Biểu đồ Use Case chương trình

* Người dùng (khách hàng) tiến hành gửi tin nhắn lên hệ thống chat Messenger của facebook.
* Khi hệ thống nhận được tin nhắn, tin nhắn sẽ được lưu vào database.
* Mặt khác, lúc này, tin nhắn sẽ được kiểm tra bởi hệ thống, nếu có thể trả lời được, hệ thống sẽ tiến hành phản hồi tin nhắn, đồng thời lưu tin đã phản hồi vào cơ sở dữ liệu. Trong trường hợp không trả lời được, hệ thống sẽ chuyển tiếp tin nhắn sang ứng dụng mà chủ trang web đang truy cập.
* Chủ trang web phản hồi lại tin nhắn cho người dùng qua hệ thống, đồng thời hệ thống sẽ lưu lại tin nhắn phản hồi này để có thể học tập và trả lời tốt hơn sau đó.

### Biểu đồ trình tự

A picture containing diagram

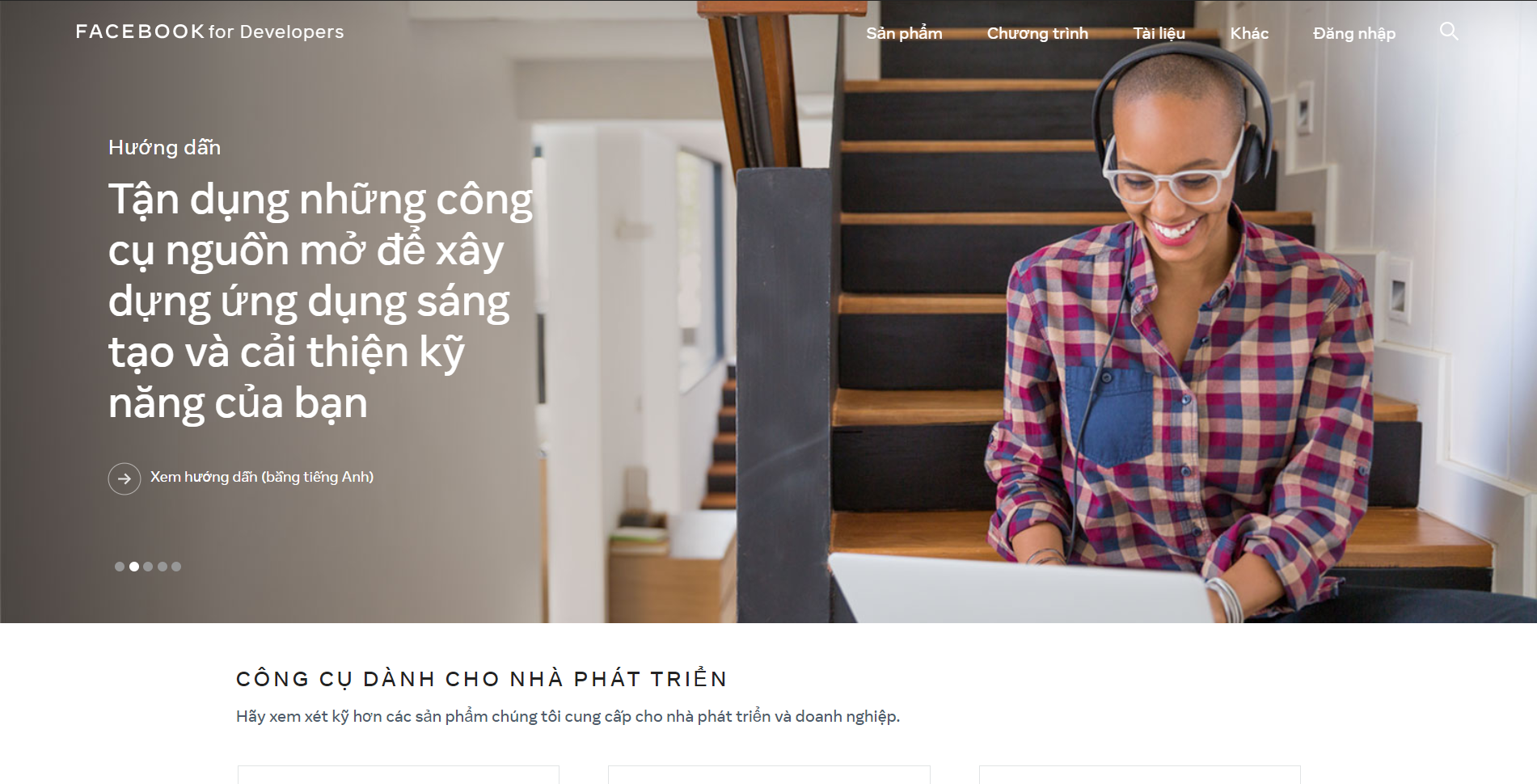
Description automatically generated

Hình 2.2 Biểu đồ trình tự

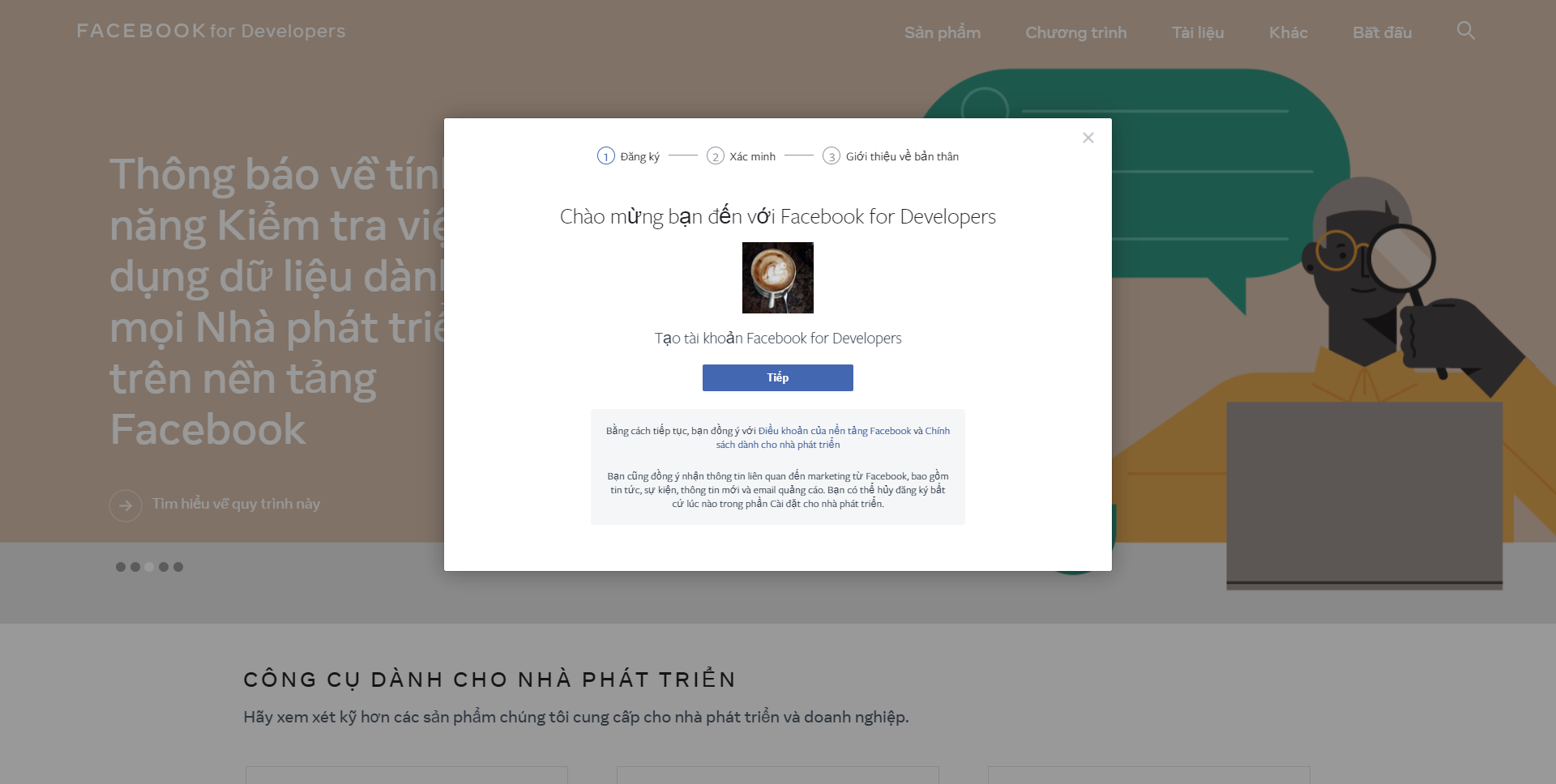
## Cài đặt chương trình

### Đăng ký facebook developer

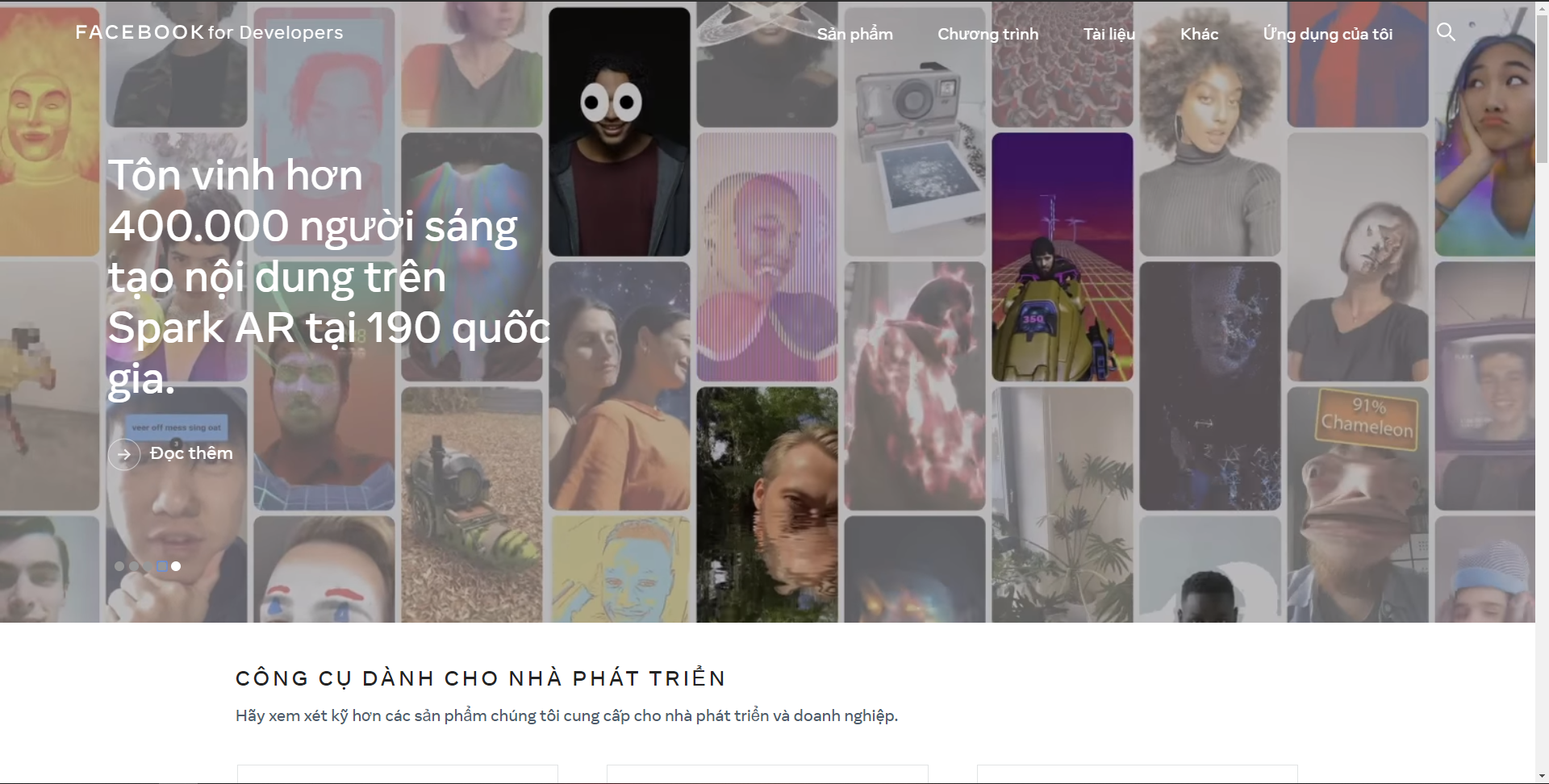
* Truy cập <https://developers.facebook.com>, chọn Đăng nhập



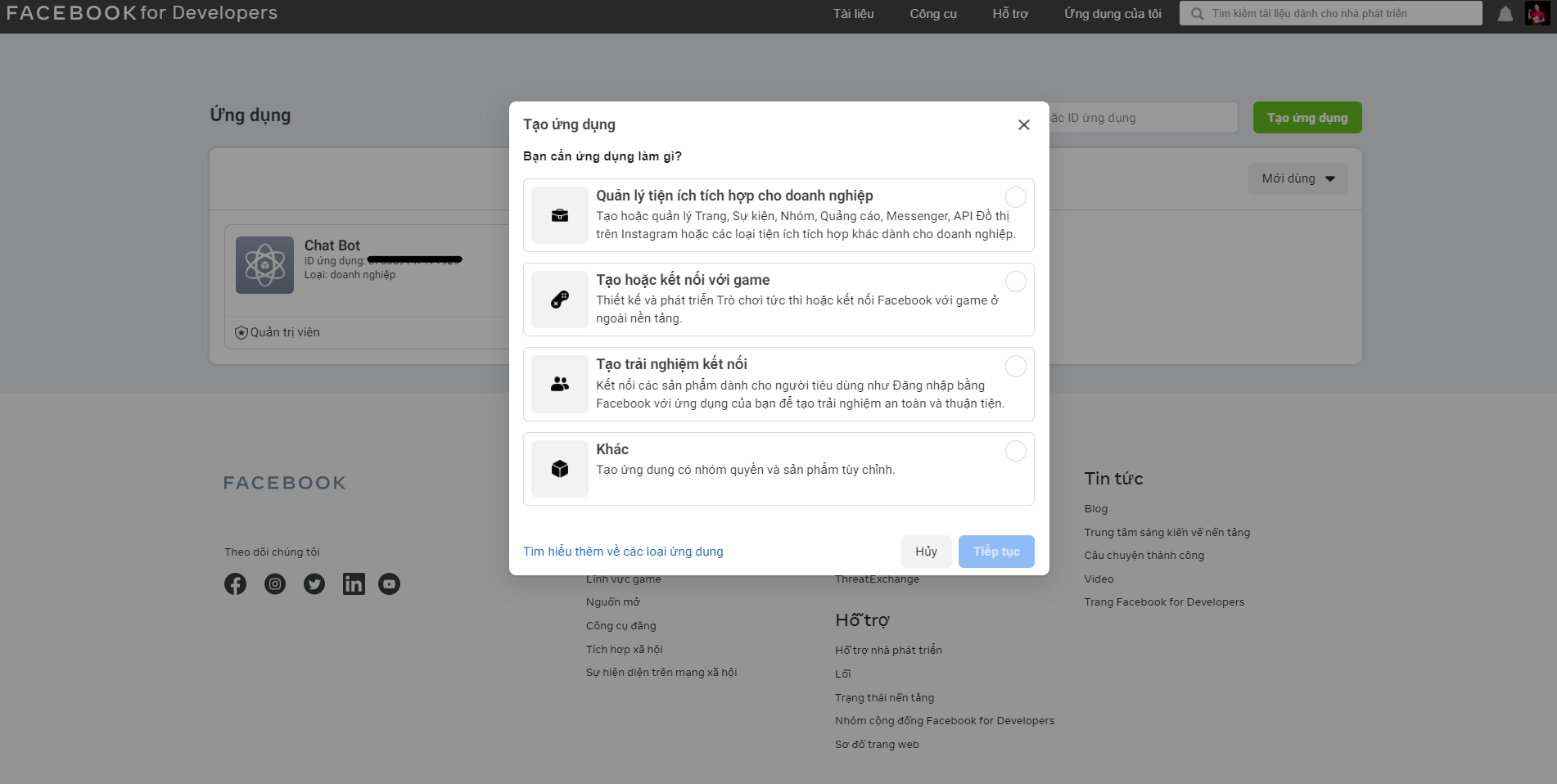
* Lựa chọn Bắt đầu => Tiếp và tiến hành xác minh tài khoản



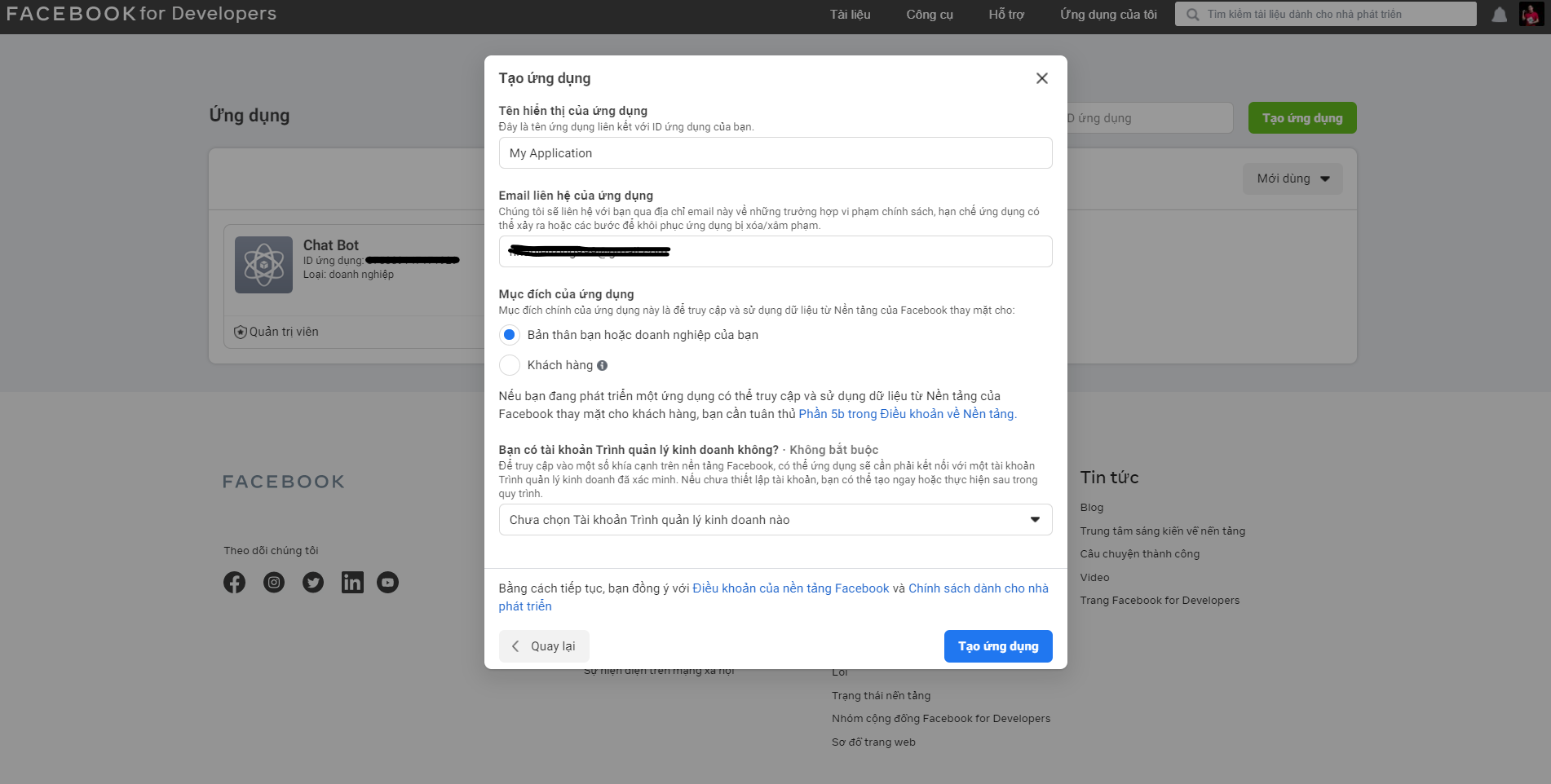
* Sau khi được xác nhận, vào lại, chọn Ứng dụng của tôi



* Chọn Tạo ứng dụng mới => Quản lý tiện ích tích hợp cho doanh nghiệp => Tiếp



* Điền các mục và chọn “Tạo ứng dụng”



* Vào Cài đặt, tạo mã xác thực ứng dụng (PAGE\_ACCESS\_TOKEN):





### Cài đặt Webooks

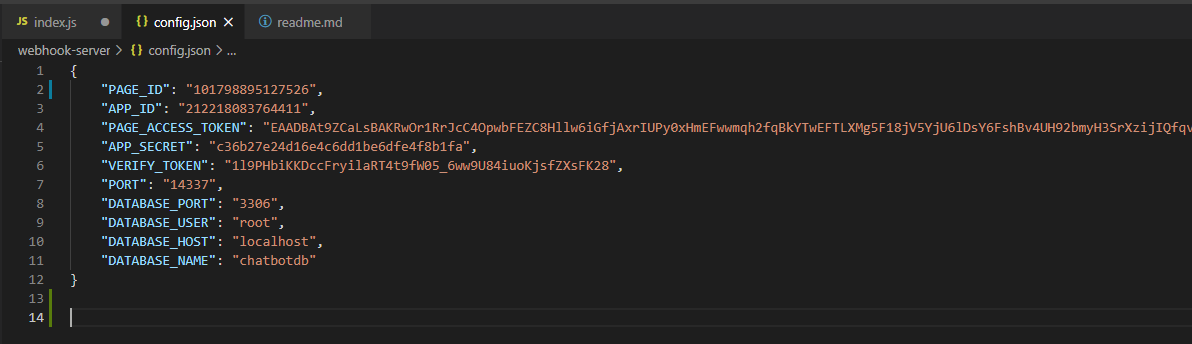
* Tạo một chương trình cho Webhook-server

|  |
| --- |
| 'use strict';  // Imports dependencies and set up http server  const  express = require('express'),  bodyParser = require('body-parser'),  app = express().use(bodyParser.json()); // creates express http server  // Sets server port and logs message on success  app.listen(process.env.PORT || 1337, () => console.log('webhook is listening'));  // Creates the endpoint for our webhook  app.post('/webhook', (req, res) => {    let body = req.body;  // Checks this is an event from a page subscription  if (body.object === 'page') {  // Iterates over each entry - there may be multiple if batched  body.entry.forEach(function(entry) {  // Gets the message. entry.messaging is an array, but  // will only ever contain one message, so we get index 0  let webhook\_event = entry.messaging[0];  console.log(webhook\_event);  });  // Returns a '200 OK' response to all requests  res.status(200).send('EVENT\_RECEIVED');  } else {  // Returns a '404 Not Found' if event is not from a page subscription  res.sendStatus(404);  }  });  // Adds support for GET requests to our webhook  app.get('/webhook', (req, res) => {  // Your verify token. Should be a random string.  let VERIFY\_TOKEN = "<YOUR\_VERIFY\_TOKEN>"    // Parse the query params  let mode = req.query['hub.mode'];  let token = req.query['hub.verify\_token'];  let challenge = req.query['hub.challenge'];    // Checks if a token and mode is in the query string of the request  if (mode && token) {    // Checks the mode and token sent is correct  if (mode === 'subscribe' && token === VERIFY\_TOKEN) {    // Responds with the challenge token from the request  console.log('WEBHOOK\_VERIFIED');  res.status(200).send(challenge);    } else {  // Responds with '403 Forbidden' if verify tokens do not match  res.sendStatus(403);  }  }  }); |

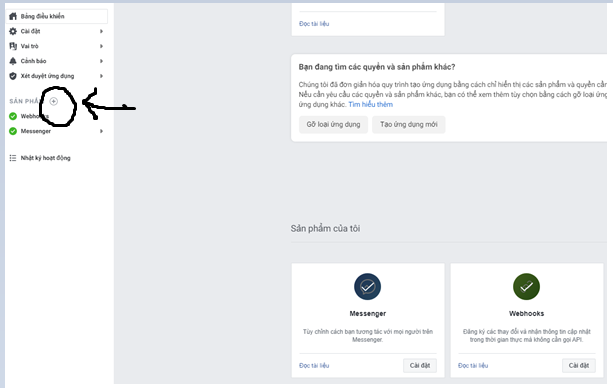
* Tạo mã xác minh riêng (một mã tuỳ ý)

Truy cập freeprivacypolicy.com để tạo 1 mã xác minh riêng

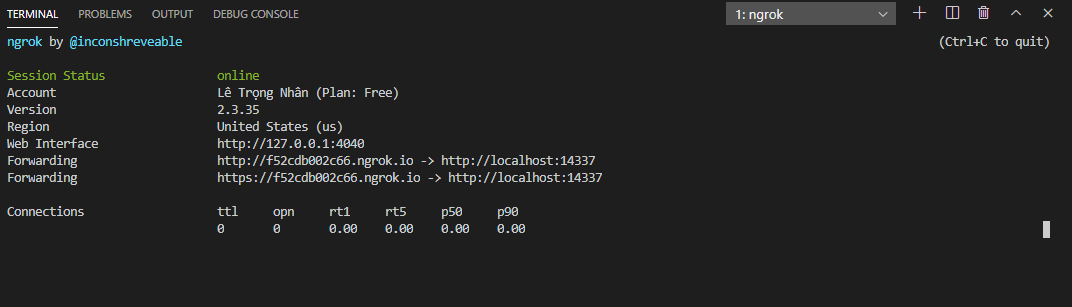
* Cấu hình các thông tin cần thiết trong file webhook-server/config.json



* + Trong đó:
    - PAGE\_ID: ID của trang facebook (Xem ở cài đặt của sản phẩm Messeger)
    - APP\_ID: ID của ứng dụng (xem ở cài đặt của app)
    - PAGE\_ACCESS\_TOKEN: Mã xác thực tạo trong ứng dụng
    - APP\_SECRET: mã của ứng dụng
    - VERIFY\_TOKEN: mã xác minh riêng.
    - PORT: Cổng (theo cấu hình ngrok)
    - DATABASE\_PORT: Cổng đối với database
    - DATABASE\_USER: Tên người dùng trong database
    - DATABASE\_HOST: Tên máy chủ trong database
    - DATABASE\_NAME: Tên database
* Cài đặt thêm sản phẩm cho trang web là messeger và webhooks

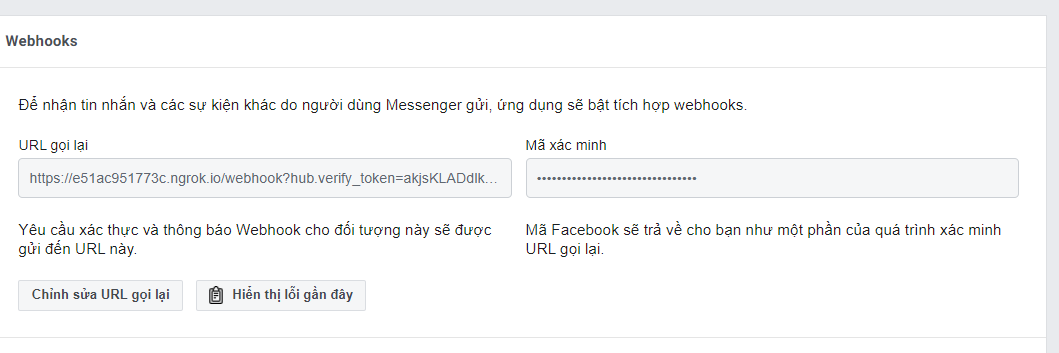


* Sử dụng ngrok để tạo server https trỏ đến server webhook vừa tạo



* Đưa đường dẫn https vừa tạo được đến mục Chỉnh sửa URL gọi lại (Edit Callback URL):

<ngrok\_https\_server>/webhook?hub.verify\_token=<VERIFY\_TOKEN>&hub.challenge=CHALLENGE\_ACCEPTED&hub.mode=subscribe (mã xác minh và VERIFY\_TOKEN là giống nhau, là mã vừa tạo bên trên, không phải mã xác thực PAGE\_ACCESS\_TOKEN)



### Xây dựng model phân loại câu hỏi bằng Machine Learnning

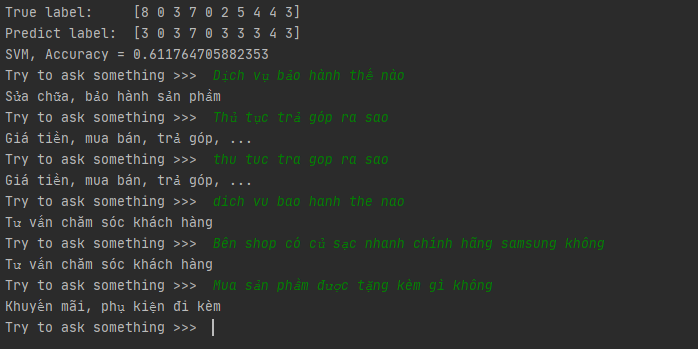
* Tiền xử lý dữ liệu text là tiếng việt, xử dụng thư viện pyvi-thư viện hỗ trợ xử lý tiếng việt như tách từ, tái tạo từ, loại bỏ dấu, …
  + Bên cạnh đó cần xử lý dữ liệu như loại bỏ stopword trong tiếng việt, loại bỏ các kí tự, số, …
* # split words  
  middle = ViTokenizer.tokenize(middle)  
    
  # remove punctuation  
  middle = re.sub(r'\W', ' ', middle) # remove puntuaction  
  middle = re.sub(r'\d', ' ', middle) # remove digits  
    
  # delete ư, ô,....  
  middle = ViUtils.remove\_accents(middle)  
  middle = middle.lower()  
    
  # delete redundant spaces  
  middle = str(middle)  
  middle = re.sub(r'\s\s+', ' ', middle.strip())  
    
  # remove stopwords  
  line\_clean = ''  
    
  for word in middle.split(' '):  
   word = re.sub(r"b'", '', word)  
   word = re.sub(r'\W', ' ', word)  
   word = re.sub(r'\s\s+', ' ', word.strip())  
    
   if len(word) > 1:  
   if word not in stopwords and pos\_tagging\_text(word):  
   line\_clean += word + ' '  
    
  result.append(line\_clean.strip())
* Phân chia dữ liệu làm 2 phần Training và Testing, sau đó huấn luyện mô hình

# split data to training data and test data # = 0.1  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(text\_data, label\_data, test\_size=test\_percent, random\_state=42)  
  
# encode label  
label\_encoder = LabelEncoder()  
label\_encoder.fit(y\_train)  
  
y\_train = label\_encoder.transform(y\_train)  
y\_test = label\_encoder.transform(y\_test)  
  
# build model using svm  
classify\_model.build\_and\_run\_svm(X\_train, y\_train)

* Sử dụng mô hình SVM với thư viện hỗ trợ sklearn của Python
* def build\_and\_run\_svm(X\_train, y\_train):  
   start\_time = time.time()  
   text\_clf = Pipeline([('vect', CountVectorizer(ngram\_range=(1, 1),  
   max\_df=0.8,  
   max\_features=None)),  
   ('tfidf', TfidfTransformer()),  
   ('clf', SVC(gamma='scale'))  
   ])  
   text\_clf = text\_clf.fit(X\_train, y\_train)  
    
   # train\_time = time.time() - start\_time  
   # print('Done training SVM in', train\_time, 'seconds.')  
    
   # Save model  
   pickle.dump(text\_clf, open(os.path.join(MODEL\_PATH, "svm.pkl"), 'wb'))
* Bên cạnh đó sử dụng thêm mô hình logistic để xem xác xuất của mô hình nào tốt hơn

def build\_and\_run\_logistic(X\_train, y\_train):  
 text\_clf = Pipeline([('vect', CountVectorizer(ngram\_range=(1, 1),  
 max\_df=0.8,  
 max\_features=None)),  
 ('tfidf', TfidfTransformer()),  
 ('clf', LogisticRegression(solver='lbfgs',  
 multi\_class='auto',  
 max\_iter=10000))  
 ])  
 text\_clf = text\_clf.fit(X\_train, y\_train)  
  
 # Save model  
 pickle.dump(text\_clf, open(os.path.join(MODEL\_PATH, "linear\_classifier.pkl"), 'wb'))

* Mỗi khi 1 câu hỏi mới đưa vào để dự đoán, nó sẽ được tiền xử lý sau đó mới đưa vào mô hình để dự đoán. Một số hình ảnh test mô hình:



### Xây dựng giao thức truyền thông điệp MQTT

* Sử dụng MaQiaTTo:
  + - * MaQiaTTo là một MQTT broker miễn phí, trực tuyến và sẵn dùng dành cho các nhà phát triển IoT hoặc các kiểm thử viên. Điều này có nghĩa chúng ta sẽ không cần mua 1 nền tảng đám mây nào đó để sử dụng MQTT broker cũng như không cần phải sử dụng bất kỳ thư viện hoặc công cụ nào cho hệ thống con tin nhắn. Nó được phát triển hoàn toàn cho các nhà phát triển IoT bởi các nhà phát triển IoT, những người đã giải quyết các vấn đề truyền thông dựa trên đám mây trên các nền tảng khác nhau.
      * MaQiaTTo dựa trên thư viện Mosquitto MQTT trong lõi của nó và có một số hoạt động cho việc quản lý truy cập (Access Control Lists) chuyên biệt cho từng người dùng
* Xây dựng 2 topic để truyền-nhận dữ liệu giữa webhook và NLP model
  + Topic Question: truyền thông tin câu hỏi (Câu hỏi của người dùng), phía webhook sẽ publish lên đây, còn phía NLP model sẽ subcribe để lấy được dữ liệu về xử lý.
  + Topic Reply: truyền thông tin câu trả lời (Câu trả lời mà model NLP dự đoán), phía NLP model sẽ publish lên đây, còn phía webhook sẽ subcribe để lấy được câu trả lời từ NLP model.
* Phía NLP model:
* BROKER = 'www.maqiatto.com'  
  PORT = 1883  
  TOPIC\_QUES = 'ductrantrung.hust@gmail.com/question'  
  TOPIC\_REPLY = 'ductrantrung.hust@gmail.com/reply'  
  USERNAME = 'ductrantrung.hust@gmail.com'  
  PASSWORD = '03021999'
  + Publish
* data = {"payload": message}  
  data\_json = json.dumps(data)  
  client = paho.Client()  
    
  client.username\_pw\_set(USERNAME, PASSWORD)  
  client.connect(BROKER, PORT, 60)  
  client.publish(TOPIC\_REPLY, data\_json)  
  client.disconnect()
  + Subcribe
* def on\_connect(client, userdata, flags, rc):  
   if rc == 0:  
   print('client is connected')  
   client.subscribe(TOPIC\_QUES)  
    
   else:  
   print('Connected failed')  
    
    
  def on\_message(client, userdata, message):  
   data = json.loads(message.payload)  
   print('message received: '+ data['payload'])  
   process\_message(data['payload'])  
    
  client = paho.Client()  
    
  client.username\_pw\_set(USERNAME, PASSWORD)  
  client.connect(BROKER, PORT, 60)  
  client.on\_connect = on\_connect  
  client.on\_message = on\_message  
  client.loop\_forever()
* Phía Webhook:

// mqtt config and connection

var url = 'mqtt://www.maqiatto.com:1883'

var options = {

    username: 'ductrantrung.hust@gmail.com',

    password: '03021999',

};

var topic\_ques = 'ductrantrung.hust@gmail.com/question'

var topic\_reply = 'ductrantrung.hust@gmail.com/reply'

// Create a client connection

var client = mqtt.connect(url, options);

* + Subcribe

function subcribe() {

    let result = '';

    console.log('subcribing...');

    // subscribe to a topic

    client.subscribe(topic\_reply, function () {

        // when a message arrives, do something with it

        client.on('message', function (topic, msg) {

            console.log("Received '" + msg + "' on '" + topic + "'");

            console.log("Received '%s'", JSON.parse(msg.content));

            result = JSON.parse(msg.content).payload;

        });

    });

}

* + Publish
* function publish\_message(msg) {
* let msgJson = {
* payload: msg
* };
* console.log('Publishing...');
* console.log(msgJson);
* // publish a message to a topic
* client.publish(topic\_ques, JSON.stringify(msgJson), function () {
* console.log("Message is published");
* });
* }

# Đánh giá kết quả

* Đối với hệ thống Chatbot tự động trên nền tảng bán hàng Facebook messeger mặc dù kết quả là tương đối, xác suất đúng ~65-70%, tuy nhiên vẫn còn một số hạn chế sau:
  + Sử dụng giao thức MQTT để giao tiếp giữa webhook và mô hình phân loại văn bản nhưng chưa xử lý các trường hợp như quá nhiều yêu cầu gửi đến mà chưa kịp phản hồi. Bên cạnh đó MQTT broker sử dụng là Maquitto – một MQTT message broker trực tuyến miễn phí có thể xảy ra trục trặc kĩ thuật mà không kiểm soát được
  + Dữ liệu huấn luyện là thu thập từ trên mạng, gán nhãn bằng tay, vì vậy sẽ không hoàn toàn chính xác đúng đắn, dữ liệu có thể bị mất cân bằng, dẫn đến việc mô hình sau khi huấn luyện dự đoán sai
  + Mô hình sử dụng là các thuật toán Machine learning cơ bản như VSM, logistic regression đem lại kết quả ổn với tập dữ liệu vừa đủ. Tuy nhiên các thuật toán này mới chỉ dừng lại ở việc chuyển đổi từ sang vector nhờ các thuật toán cổ điển (count-vector, tf-idf). Nếu có một bộ dữ liệu đủ lớn, việc áp dụng deeplearning cùng với việc xử lý dữ liệu bằng các mô hình như word2vec sẽ đem lại kết quả tốt hơn

# Tài liệu tham khảo

[1] Cài đặt và các hàm trong thư viện Pivi

<https://pypi.org/project/pyvi/>

[2] Phân loại văn bản tiếng việt

<https://nguyenvanhieu.vn/phan-loai-van-ban-tieng-viet/>

[3] Cấu hình Facebook Messeger Bot

<https://www.youtube.com/watch?v=5wPrfMxvrgo&t=241s>

[4] Kết nối Maqiatto

<https://www.maqiatto.com/connect>

[5] Phân loại văn bản bằng Navie Bayes

<https://viblo.asia/p/phan-loai-bang-naive-bayes-phan-1-naQZRJAjZvx>