**8장. 봇과 대화**

와, 봇? 정말이니? 즐거움을 위해 봇을 만드는 것을 배울 것인가 아니면 비즈니스 사용 사례를 위해 봇을 만드는 것을 배울 것인가? 그래, 물론 8장에서는 파이썬을 사용한 새로운 봇의 세계로 안내한다.

8장은 다음을 살펴볼 것이다.

-감정 텔레그램 봇 개발

-다양한 유형의 봇 : stateless, stateful, and smart

-인공 지능을 갖춘 스마트 봇

-봇으로 비즈니스 프로세스 자동화

**소개**

지난 수십 년 동안 디지털 변환 및 자동화 시대가 왔다. 오늘날 대부분의 기업은 전통적인 제품 판매 방식보다는 온라인 판매 모델을 선호한다.

웹사이트는 기업들의 영역을 확대하도록 도움을 줄 뿐만 아니라, 제품을 판매하는 것을 더 저렴하게(렌탈 같은 고정 비용이 없음) 만들었다. 반응형 그래픽 사용자 인터페이스(GUI, graphical user interface)는 실시간 기술의 힘과 결합되어 보다 쉽게 판매 프로세스를 만들었다. 현재의 경영진들은 단지 잠재 고객들과 대화하고 제품 구매를 유도하여 변화를 증가시킬 수 있다.

인공지능(AI, artificial intelligence) 및 언어 처리 기술의 발달에 따라 비즈니스는 천천히 그러나 꾸준히 프로세스를 자동화하기 위해 대화식 인터페이스를 채택했다. 대화식 사용자 인터페이스는 자연어에 대한 자유 형태 텍스트가 있는 인터페이스를 나타낸다. 대화식 인터페이스 및 자연어 처리 기술을 사용하여 비즈니스는 컨텍스트(context)를 분석하여 특정 고객 질의에 응답할 수 있다고 생각한다. 오늘날의 세계에서 이러한 머신을 chatbots이라고 한다.

8장에서는 다양한 유형의 봇에 대해 배우고, 간단한 챗봇을 개발하는 방법을 살펴보고, 봇을 사용하여 비즈니스 프로세스를 자동화하는 방법에 대해 살펴본다. 또한 8장의 봇을 참조할 때는 대화 챗봇이나 텍스트 기반 봇에 대해 설명한다.

**봇은 무엇인가?**

좋아, 간단한 예제를 살펴보자. 이번 주말에 피자 헛(Pizza Hut)에서 친구들과 저녁에 피자를 주문하고 싶다고하자. 일반적으로 피자 헛 웹사이트로 이동하여 특정 유형의 피자 혹은 원하는 특정 토핑을 찾는 데 시간을 할애하고 주문할 수 있다. 대개, 당신은 이미 당신이 주문하고 싶은 것을 안다. 그렇다면 진짜 질문은 피자 헛 웹사이트에서 그것을 찾는 데 고통을 감수해야 할까?

더 이상 걱정하지마! 페이스북(Facebook)에 로그인하고 페이스북 메신저(Facebook Messenger) 챗봇을 사용하여 피자 헛에서 필요한 것을 구입한다. 뿐만 아니라 챗봇은 피자 헛에서 최신 정보 를 수정하고 게시한다. 그래서 챗봇은 좋아하는 소셜 네트워킹 플랫폼에서 웹사이트를 방문하는 것과 같은 경험을 줄 수 있다. http://blog.pizzahut.com/press-center/pizza-hut-announces-new-social-ordering-platform/에서 페이스북 메신저와 공동 작업에 대한 피자 헛의 발표를 살펴본다.

" 그래, 사용 사례를 이해하지만, 챗봇은 정확히 무엇일까?"라고 말할 수 있다.

챗봇은 고객으로서 챗(텍스트) 인터페이스를 통해 상호 작용하는 규칙 및 AI로 구동되는 서비스이다. 봇은 세미-인텔리전트 혹은 평범한 작업을 수행하고 소프트웨어 애플리케이션으로 실행한다. 챗봇은 여러 서비스를 제공 할 수 있으며 페이스북(Facebook), 텔레그램(Telegram), 슬랙(Slack) 등과 같은 소셜 플랫폼에서 실행할 수 있다. 챗봇은 여전히 활발한 연구 활동을 진행중이고 떠오르는 컴퓨터 과학 분야이다.

**봇의 작동방법**

지금까지 이야기한 것에 기초해보면, 여러분은 아마 이렇게 생각할 것이다. “이 봇은 어떻게 기능할까? 어떻게 인간의 단어 혹은 감정을 이해할까?” 따라서 여기에 대답이 있다. 일반적으로 두 종류의 챗봇이 있다.

규칙 엔진에서 작동하는 봇: 이 유형의 봇은 특정 단어 혹은 커맨드(말하자면)를 이해하므로 매우 제한적으로 행동한다. 그것은 꽤 간단하다. x가 입력이면 y가 출력이어야 한다. 고정된 일련의 질문이 있거나 질문이 질의로 작동하는 경우에 매우 유용하다. 예를 들어, CNN 챗봇은 그 순간에 가장 중요한 이야기를 할 수 있도록 해 주고, 게다가, 여러분은 정치 혹은 비즈니스와 같은 특정 주제에 대한 TOP 순위 스토리에 대해 TOP 순위의 스토리를 봇에게 물어본다(좋아! 그런데 왜 CNN 웹사이트에 방문해야 할까?). 페이스북 메신저 앱에서 CNN 챗봇과의 상호 작용으로 가져온 몇 개의 스크린샷을 살펴본다. 첫 번째 화면에서는 GET STARTED 클릭을 요청하고, 이렇게 하면 다음 화면으로 이동하여 TOP 스토리를 볼 수 있는 옵션을 제공한다.

<그림>

TOP STORIES를 클릭하면 Yahoo! 스토리를 보여주고 예를 들어 정치 같은 특정 주제에 관심이 있는지 묻는다.

<그림>

기계 학습에서 작동하는 스마트 봇: 스마트 봇은 AI 및 감정 분석을 사용하여 대화의 컨텍스트를 이해하고 언어 의미에 응답한다. 따라서 제품 구매 혹은 고객 지원 문의 응답과 같은 정교한 사용 사례에 적용할 수 있다. 더 나아가, 이 봇은 과거의 상호 작용에서 배울 수 있다. 놀랍다, 그렇지 않아?

<<중요표시시작>>

감정 분석은 오피니언 마이닝(opinion mining)이라고도 하며, 이용 가능한 텍스트에서 주관적인 정보를 확인하고 주관적인 정보를 추출하여, 텍스트의 문맥별 특성을 파악하는 것을 목적으로 한다.

<<중요표시종료>>

**왜 지금 봇이 필요한가?**

“세계는 지금 한동안 기계 학습을 이야기해 왔고, 챗 기능은 지금 오랜 시간에 걸쳐 있는데, 왜 봇이 지금 그렇게 중요해 지고 있는 것일까?”라고 물어볼 수 있다. 그 이유는 다음과 같다.

-사용 패턴(Usage patterns): 기업은 사용자가 소셜 미디어 플랫폼 혹은 웹사이트보다 챗에 더 많은 시간을 소비하는 경향을 파악했다. 따라서 기업은 챗 플랫폼으로 보다 나은 방법으로 사용자와 소통할 수 있다.

-비용 효과적(Cost-effective): 인간이 필요없다 - 전혀 비용이 들지 않는다! 기업은 인적 자원 투자 없이 고객 서비스와 같은 프로세스를 자동화하기 위해 봇을 이용한다.

-규모(Scale): 봇의 배포 경로로 작용하는 페이스북 혹은 텔레그램을 통해 수백만 명의 사용자에게 손쉽게 접근할 수 있다. 이러한 방식으로, 기업은 관련된 인건비를 확인하지 않고 가능한 많은 잠재 고객을 대상으로 삼을 수 있다.

-효율적인 기술(Efficient technology): AI 혹은 자연어 처리(NLP, Natural Language Processing)의 성장으로 알고리즘을 이러한 봇에 쉽게 연결할 수 있다. 알고리즘은 시간이 지남에 따라 발전할 수 있고, 앞으로도 발전할 것이며 고객에게 더 나은 서비스를 제공할 것이다.

그래, 좋아! 이제 봇과 그 유용성을 훨씬 더 잘 이해할 수 있게 되었으니, 손을 더럽히고 나만의 봇을 개발하자.

**감정 텔레그램 봇 개발**

봇 개발을 시작하기 전에, 목표를 분명히 해야한다. 봇은 무엇을 할 것인가? 사용자의 기분에 기초하여 이모지(emoji)로 반응하는 봇을 만드는 간단한 예제를 살펴본다. 그것은 사용자의 기분을 나타내는 간단한 이유 때문에 감정 봇이다. 흥미로운 사용 사례처럼 생각하는가? 시작하자.

이 레시피는 텔레그램 봇을 개발하기 위해 python-telegram-bot(https://github.com/python-telegram-bot/) 라이브러리를 사용할 것이다. 먼저 가장 인기있는 유틸리티인 파이썬 pip로 python-telegram-bot 모듈을 설치하자.

(bots)chetans-MacBookPro:ch09 Chetan$ pip install python-telegram-bot --

upgrade

Collecting python-telegram-bot

Downloading python\_telegram\_bot-5.1.0-py2.py3-none-any.whl (134kB)

100% |████████████████████████████████| 135kB

681kB/s

Collecting certifi (from python-telegram-bot)

Downloading certifi-2016.8.31-py2.py3-none-any.whl (379kB)

100% |████████████████████████████████| 380kB

612kB/s

Collecting future>=0.15.2 (from python-telegram-bot)

Downloading future-0.15.2.tar.gz (1.6MB)

100% |████████████████████████████████| 1.6MB

251kB/s

Collecting urllib3>=1.10 (from python-telegram-bot)

Downloading urllib3-1.17-py2.py3-none-any.whl (101kB)

100% |████████████████████████████████| 102kB

1.2MB/s

Building wheels for collected packages: future

Running setup.py bdist\_wheel for future

Stored in directory:

/Users/chetan/Library/Caches/pip/wheels/11/c5/d2/ad287de27d0f0d646f119dcffb

921f4e63df128f28ab0a1bda

Successfully built future

Installing collected packages: certifi, future, urllib3, python-telegram-

bot

Successfully installed certifi-2016.8.31 future-0.15.2 python-telegram-

bot-5.1.0 urllib3-1.17

또한 이모지 아이콘을 위한 emoji(https://github.com/carpedm20/emoji) 라이브러리를 설치하여 기분에 따라 적절한 표현을 사용자에게 반환할 수 있다.

(bots)chetans-MacBookPro:ch09 Chetan$ pip install emoji --upgrade

Collecting emoji

Downloading emoji-0.3.9.tar.gz

Building wheels for collected packages: emoji

Running setup.py bdist\_wheel for emoji

Stored in directory:

/Users/chetan/Library/Caches/pip/wheels/94/fc/67/441fb0ca2ed262d6db44d9ac2d

fc953e421f57730004dff44d

Successfully built emoji

Installing collected packages: emoji

Successfully installed emoji-0.3.9

이미 모듈을 설치했는가? 좋아! 계속 살펴보자.

**실행방법**

1. 나만의 봇을 개발하려면 먼저 휴대폰에서 텔레그램 앱을 다운로드 한다. 계정을 등록하고 번호를 확인한다. 이것을 진행했으면, 축하한다! 텔레그램 봇을 생성하는 데 한 걸음 더 가까워졌다.

2. 자, 이제 여러분의 봇을 등록하기 위해 해야할 다음 일은 BotFather이라는 다른 봇에게 연락하는 것이다. 텔레그램 앱에서 BotFather를 검색하고 클릭하여 대화를 시작한다. 이것은 다음과 같이 보인다.

<그림>

3. BotFather과 대화를 시작한 후 다음 단계를 수행하고 /newbot 혹은 /enable과 같은 커맨드를 사용하여 봇을 구성한다. 단계를 주의깊게 따라하면 새 봇을 생성할 것이다. 다음 스크린샷은 새 봇을 생성하는 프로세스를 안내해 줄 것이다.

<그림>

4. 새 봇을 생성하면, 봇에 특정한 토큰을 받을 것이다. 이것을 가지고 다니면서 안전하게 보관한다. 누구와도 공유하면 안 된다. 다음 스크린샷은 BotFather가 작동하는 방식과 토큰이 보이는방법을 보여준다.

<그림>

5. 좋아! 따라서 나만의 봇을 생성했다. 하지만 봇은 기능적이지도 않고 아직 놀라운 일을 하지 않았다. 레시피의 시작 부분에서 계획했던 것처럼 멋진 것을 생성하자. bot.py라는 파일을 생성하고 다음 코드를 복사한다. 또한, 토큰(token)을 봇의 토큰 ID로 변경해야 한다.

import logging

from telegram import InlineKeyboardButton, InlineKeyboardMarkup

from telegram.ext import Updater, CommandHandler, CallbackQueryHandler

import emoji

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',

level=logging.INFO)

def start(bot, update):

keyboard = [

[InlineKeyboardButton("Happy", callback\_data='1'),

InlineKeyboardButton("Whatever", callback\_data='2')],

[InlineKeyboardButton("Sad", callback\_data='3')]]

reply\_markup = InlineKeyboardMarkup(keyboard)

update.message.reply\_text('Hey there! How do you feel today?', reply\_markup=reply\_markup)

def button(bot, update):

query = update.callback\_query

if query.data == "1":

em = emoji.emojize(':smile:', use\_aliases=True)

bot.editMessageText(text="Oh wow! %s " % em,

chat\_id=query.message.chat\_id,

message\_id=query.message.message\_id)

if query.data == "2":

em = emoji.emojize(':expressionless:', use\_aliases=True)

bot.editMessageText(text="Does it matter? %s " % em,

chat\_id=query.message.chat\_id,

message\_id=query.message.message\_id)

if query.data == "3":

em = emoji.emojize(':disappointed:', use\_aliases=True)

bot.editMessageText(text="Oh man! %s " % em,

chat\_id=query.message.chat\_id,

message\_id=query.message.message\_id)

def help(bot, update):

update.message.reply\_text("Use /start to test this bot.")

def error(bot, update, error):

logging.warning('Update "%s" caused error "%s"' % (update, error))

# Updater를 생성하고 bot의 토큰을 전달한다.

updater = Updater('Token')

updater.dispatcher.add\_handler(CommandHandler('start', start))

updater.dispatcher.add\_handler(CallbackQueryHandler(button))

updater.dispatcher.add\_handler(CommandHandler('help', help))

updater.dispatcher.add\_error\_handler(error)

# 봇을 시작한다.

updater.start\_polling()

# 사용자가 Ctrl-C를 누르거나 프로세스가 SIGINT 혹은 SIGTERM, SIGABRT를 수신할 때까지

# bot을 실행한다.

updater.idle()

6. 그래 좋아! 이제 필요한 기능을 봇에 추가했고, 제대로 작동할 것으로 기대한다. 하지만 봇을 어떻게 테스트할까? 먼저 다음 커맨드로 파이썬 파일을 실행한다.

python bot.py

7. 그런 다음 봇을 검색하고 대화를 시작한다. 내 경우, 봇은 Chetbot라고 불리며 표준 /start 커맨드를 사용하여 대화를 시작했다.

<그림>

8. 앞의 스크린샷에서 봇과 대화를 시작했을 때, 그 날의 내 기분을 물어봤고 세 가지 옵션을 제시했다. 세 가지 옵션은 Happy, Whatever, Sad다.

9. 멋지다! 하지만 이 옵션 중 하나를 클릭하면 무슨 일이 생길까? 봐봐! 이모티콘으로 하루의 내 기분을 반환한다. 굉장하다!

<그림>

10. 대화를 다시 시작해야 한다면, 봇과 대화하기 위해 /start 커맨드를 다시 입력해야 한다. 다음 스크린샷에서 봇은 시작 커맨드를 인식하고 내 기분을 다시 묻는다. 멋지다, 그렇지?

<그림>

**작동원리**

python-telegram-bot 모듈은 표준 이벤트 기반 철학에서 작동한다. 봇은 이벤트에 대한 폴링을 유지하는 단일 스레드 이벤트 루프로 간주될 수 있다. 이벤트 루프는 또한 dispatcher라는 커맨드 핸들러와 함께 등록된다. 이벤트가 유발되는 즉시, 콜백은 이벤트를 처리하고 사용자에게 원하는 응답을 반환한다.

앞의 코드에서는 start()와 help()의 두 가지 커맨드 핸들러(command handlers)를 등록했다. start() 메소드는 사용자가 봇과 대화를 시작(/start 커맨드)하거나 도움말(/help 커맨드)을 요청할 때 호출된다.

또한 button()이 있는 콜백 쿼리 핸들러를 콜백 메소드로 추가했다. 이것은 사용자가 봇의 옵션에 응답할 때 호출된다.

2018.04.02

따라서 처음에는 입력을 기다리는 봇이 실행 중이다. 사용자가 /start라고 하면, 요청은 start() 메소드에 전송되고, 이 후 사용자에게 Hey there! How do you feel today?를 제시하고 Happy, Whatever 혹은 Sad 세 가지 옵션으로 인라인 키보드를 보여준다.

사용자가 옵션 중 하나를 선택하면 콜백 메소드 button()에 의해 처리되는 이벤트가 생성된다. 콜백은 선택한 옵션에 따라 작동하는 사전 로드된 데이터가 있다. 사용자의 선택에 따라 봇은 올바른 감정을 사용자에게 재 전송한다. 이모티콘은 모든 표현식이 구현된 emoji 라이브러리의 도움으로 사용자에게 다시 반환된다.

**더 알아보기**

좋아! 이미 나만의 봇을 생성했을까? 텔레그램 봇이 유용한 다른 간단한 예제를 생각할 수 있을까? telepot(https://github.com/nickoala/telepot) 혹은 twx.botapi (https://github.com/datamachine/twx.botapi) 같은 텔레그램으로 봇을 개발하여 사용할 수 있는 파이썬 모듈이 많이 있다. 모두 좋다. 둘 중 하나를 사용하여 봇을 실행할 수 있다. 무엇을 제공하는지 시험해 보는건 어떨까?

**다양한 종류의 봇**

스스로 봇을 만드는 것에 자신감을 갖고, 한 걸음 나아가 봇이 어떻게 분류될 수 있는지 살펴보자.

마지막 레시피에서 개발한 봇에는 지능이 낮은 태그를 붙일 수 있다. 지능적이지 못하기 때문에 사용자에게 질문하고 옵션에 따라 이모지(emoji)로 응답했다. 하지만 사용자가 다시 /start를 말하면 봇은 같은 질문을 던진다. 도움이 되지 않는다, 그렇지?

봇이 이전의 선택을 기억하고 여러분이 도시에서 갈 수 있는 몇가지 멋진 기사나 장소로 동기를 부여하는 시나리오는 어떤가? 기분을 바꾸고 싶은가? 실제로 행복 지수를 높이는 것은 어떨까?

위의 논의를 관점에서 설명하기 위해, 봇은 구현을 기반으로 세 가지 범주로 분류할 수 있다.

상태없는 봇(stateless bots): 이것은 봇 (bot)이 아닙니다.(확인) 그들은 정보를 유지하지 않는다. 다시 말하면, 모든 상호 작용은 새로운 세션이며 모든 질문을 개별적으로 처리한다. 예를 들어, 뉴스 봇은 최신 기사에 대한 최신 정보를 계속 제공하거나 정치 분야의 주요 기사를 항상 반환할 수 있다. 그러나 대화 상태를 기억하지 못한다면 상태가 없는 것으로 간주되어 유용하다고 간주되지 않는다. 오늘날 구축된 대부분의 봇은 이 범주에 속하는데, 그 이유는 이 봇이 제공하는 가치가 매우 제한적이기 때문이다.

상태있는 봇(Stateful bots): 앞의 포인트에서 뉴스 봇에 대해 살펴봤다. 뉴스 봇이 사용자의 관심에 대한 뉴스 카테고리를 기억하고 이에 따라 과거의 이야기를 더 많이 추천한다면 사용자가 흥미로운 기사를 읽을 수 있을까?그래, 이제 우린 사업을 말한다. 이렇게하면 사용자는 더 오랜 기간 동안 봇과 계속 대화할 수 있다.

이러한 봇은 사용자의 본질을 추적하고 현재 및 이전 세션의 정보를 유지한다. 예를 들어, 이러한 봇은 오늘 및 과거로부터 검색된 뉴스 카테고리를 저장하고 검색된 카테고리와 일치하는 사용자 뉴스 피드를 추천할 수 있다.

이러한 봇은 유용하지만 똑똑하지는 않다. 봇은 문맥과 언어의 의미를 이해하지 못한다.

스마트 봇: 스마트 봇은 많은 배터리가 연결된다. 그들은 기계 학습을 사용하고, 언어의 의미를 이해하고, 가지고 있는 데이터를 기반으로 예측 알고리즘을 구축할 수 있다.

기저귀와 맥주의 유명한 예제를 살펴보자. 구매 패턴을 분석하면 맥주 구입과 기저귀 구매 사이에는 높은 상관 관계가 있는데 기저귀를 구입하는 사람이 더 많이 혹은 적게 맥주를 구입하는 것을 의미한다. 스마트 봇은 데이터를 유지하고 대화에 의미있는 통찰력을 줄 수 있는 패턴을 제시할 수 있다. 언어 의미론의 또 다른 예제를 살펴본다. 문구 "filthy awesome"에 대해 생각한다. 지금 더러운, 더럽고 두려운 것이 아주 긍정적인 단어라는 것을 의미한다. 스마트 봇은 이 문구를 이해하고 사용자의 컨텍스트를 훨씬 잘 이해할 수 있다.

이전의 분류를 기반으로 특정 유스 케이스에 대해 어떤 종류의 봇을 개발해야하는지 결정해야 한다. 스마트 봇은 고객 지원의 경우처럼 상호 작용이 훨씬 인간적이고 관여가 되는 경우에 종종 필요하지만 스마트한 봇을 사용하여 생산성을 높일 수 있는 비즈니스를 상상해본다.

**인공 지능을 갖춘 스마트 봇**

이전 절에서 여러 유형의 봇에 대한 지식을 가지고 파이썬에서 인공 지능과 감정 분석(sentiment analysis)을 사용하는 봇을 작성해보자, 그러나 그 전에 이 두 필드를 간략하게 이해하자.

인공 지능(AI, Artificial intelligence)은 인간과 마찬가지로 반응할 수 있는 머신을 생성하는 데 중점을 둔 컴퓨터 과학 영역이다. 본질적으로 인공 지능은 컨텍스트를 인식하고 성공 가능성을 극대화하기 위해 컨텐트와 관련된 조치를 취하는 머신과 관련된다.

한편, 감정 분석은 텍스트를 식별하고 분류하여 관련자의 의견이나 태도가 제품이나 이벤트에 대해 긍정적인지, 중립인지 혹은 부정인지 여부를 결정한다. 텍스트 분석을 수행하고 컨텐츠의 주관적인 정보 또는 감점을 추출하기 위해 자연어 처리 알고리즘을 사용하는 것을 의미한다.

제 생각에 AI와 감정 분석이 우리의 봇에서 다양한 요구에 어떻게 사용될 수 있는지 생각해봐야한다. 이 레시피에서는 이러한 기술을 사용한 스마트 봇을 만들어보자.

<<중요표시시작>>

스마트 봇은 예측 인텔리전스, AI, NLP 등과 같은 여러 기술을 기반으로 구축될 수 있다. 그러나 목표를 달성하는 데 사용할 기술을 결정하는 것은 전적으로 당신에게 달려있다. 또한 봇은 웹이나 앱에 있을 필요가 없다. 간단한 CLI 기반 봇이 될 수 있다. 웹 UI, CLI 또는 모바일 앱은 봇의 배포자로 사용될 수 있지만 봇을 제작할 필요는 없다.

<<중요표시종료>>

**준비하기**

봇에 AI를 포함시키기 위해 aiml이라는 잘 알려진 파이썬 모듈을 사용할 것이다. AIML은 Artificial Intelligence Markup Language의 약자이지만 본질적으로 XML 파일이다. AIML은 패턴 매칭 및 응답 결정을 위한 규칙을 정의하는 XML 형식이다. 이제, aiml 모듈을 설치하여 시작하자.

chetans-MacBookPro:ch09 Chetan$ source bots/bin/activate

(bots)chetans-MacBookPro:ch09 Chetan$

(bots)chetans-MacBookPro:ch09 Chetan$ pip install aiml

Collecting aiml

Installing collected packages: aiml

Successfully installed aiml-0.8.6

**실행방법**

1. 1단계로서 AIML 파일을 생성한다. 좋아하는 편집기로 이동하여 일반 XML 파일과 마찬가지로 AIML 파일을 다음 컨텐츠와 함께 생성한다.

<aiml version="1.0.1" encoding="UTF-8">

<!-chat.aiml à

<category>

<pattern>HELLO</pattern>

<template>

Hi, hello!

</template>

</category>

<category>

<pattern>WHO ARE \*</pattern>

<template>

<random>

<li>I'm a bot!</li>

<li>Bad guy!</li>

<li>My name is superman!</li>

</random>

</template>

</category>

<category>

<pattern>AWESOME \*</pattern>

<template>

You're nice too! J

</template>

</category>

</aiml>

2. 다음으로, AIML 파일을 로드하는 시작 XML 파일을 생성한다. 앞의 AIML 파일에 추가한 인공 지능도 로드된다. 이 파일을 init.xml이라고 하자.

<aiml version="1.0.1" encoding="UTF-8">

<!-- init.xml -->

<!-- Category is an atomic AIML unit -->

<category>

<!-- Pattern to match in user input -->

<!-- If user enters "LOAD AIML B" -->

<pattern>LOAD AIML B</pattern>

<!-- Template is the response to the pattern -->

<!-- This learn an aiml file -->

<template>

<learn>chat.aiml</learn>

<!-- You can add more aiml files here -->

<!--<learn>more\_aiml.aiml</learn>-->

</template>

</category>

</aiml>

3. 이제 챗봇을 실행하기 위해 파이썬 코드를 개발해보자. 다음 코드는 필요로 하는 것을 정확하게 수행한다. 이 파일을 aibot.py라고 부른다.

import aiml

# Create the kernel and learn AIML files

kernel = aiml.Kernel()

kernel.learn("init.xml")

kernel.respond("load aiml b")

# Press CTRL-C to break this loop

while True:

print kernel.respond(raw\_input("Enter your message >>"))

4. 이 봇은 파이썬 aibot.py 커맨드로 실행하면 사용자 입력을 기다리는 입력 화면이 나타난다. 어떻게 작동하는지 보려면 다음 스크린샷을 확인한다.

<그림>

**작동원리**

위의 파이썬 코드는 AI에 기반을 둔 전형적인 봇을 모방한 것이다. 파이썬 코드를 실행하면 amil.Kernel()이 AI 커널을 로드한다.

커널이 로드되면 kernel.learn()은 시작 XML 파일을 호출한다. AIML 룰 엔진은 load aiml b 커맨드가 커널에 전송될 때 로드된다.

엔진이 커널에 로드되면 봇과 자유롭게 채팅할 수 있다.

앞의 스크린샷에서 hello라고 하면 챗(chat.aiml 파일)을 인식하고 chat.aiml에 구성된 Hi, hello!와 함께 응답한다.

두 번째 경우, AI 봇은 사용자가 who are you?를 요청하는 경우 패턴 WHO ARE\*과 일치한다. 패턴은 chat.aiml에서 다시 정의된다.

만약 관찰한다면, WHO ARE \* 패턴은 chat.aiml 파일에서 다중 응답을 위해 구성되고 봇은 런타임에 무작위 응답을 선택하고 My name is superman!을 반환한다.

**봇을 통한 비즈니스 프로세스 자동화**

지금까지 8장에서 봇이 무엇인지, 어떻게 만들어 졌는지, 그리고 봇이 사용될 수 있는 몇 가지 간단한 사용 사례를 다뤘다. 지금까지 개발한 지식으로 제이의 문제를 해결하고 어쩌면 봇을 만드는 법에 대해 더 많이 배울 수 있는 방법을 살펴보자.

제이는 유명한 서적 출판사인 MyBooks의 마케팅 매니저이다. 제이의 업무는 책 홍보 이메일을 찾는 것이다. 그는 발신하는 홍보 이메일이 너무 일반적이어서 독자에게 효과적으로 타겟팅되지 않는다고 생각한다. 예를 들어, 파이썬 학습 경로의 이메일은 자바 개발자가 비용을 지출하도록 권장하지 않을 수 있다. 그는 청중의 관심사를 이해하고 상호 작용을 보다 관련되게 만들면 훨씬 더 잘할 수 있다고 생각한다. 독자는 훨씬 더 책을 이런 방식으로 구입하기를 원할 것이다. 그는 또한 많은 독자(잠재 구매자)가 페이스북에 있다고 느끼지만 현재 출판사가 연락하지는 않는다. 제이를 도울 수 있을까?

**준비하기**

예, 제이에게 환상적인 봇을 개발하도록 도와주자. 제이의 문제를 살펴보면 제이는 청중(이 경우 책을 구입하는 데 관심이 있는 독자)을 이해하고 관심사에 따라 책을 제안해야한다. 따라서 봇은 독자들로부터 관련 정보를 얻을만큼 똑똑해야 한다.

또한 독자는 이미 페이스북에 있기 때문에 MyBooks Facebook 페이지를 생성하고 페이스북 메신져 봇을 만들어 독자와 연락할 수 있다. 이 작업을 수행하는 방법을 살펴보자.

봇을 만들기 전에 이 연습에 필요한 파이썬 모듈을 몇 개 설치하자. 파이썬 pip를 사용하여 플라스크와 요청 모듈을 설치한다.

(bots)chetans-MacBookPro:ch09 Chetan$ pip install flask

Collecting flask

Using cached Flask-0.11.1-py2.py3-none-any.whl

Collecting click>=2.0 (from flask)

Collecting itsdangerous>=0.21 (from flask)

Collecting Werkzeug>=0.7 (from flask)

Downloading Werkzeug-0.11.11-py2.py3-none-any.whl (306kB)

100% |████████████████████████████████| 307kB

1.4MB/s

Collecting Jinja2>=2.4 (from flask)

Using cached Jinja2-2.8-py2.py3-none-any.whl

Collecting MarkupSafe (from Jinja2>=2.4->flask)

Installing collected packages: click, itsdangerous, Werkzeug, MarkupSafe,

Jinja2, flask

Successfully installed Jinja2-2.8 MarkupSafe-0.23 Werkzeug-0.11.11

click-6.6 flask-0.11.1 itsdangerous-0.24

(bots)chetans-MacBookPro:ch09 Chetan$ pip install requests

Collecting requests

Using cached requests-2.11.1-py2.py3-none-any.whl

Installing collected packages: requests

Successfully installed requests-2.11.1

**실행방법**

1. 페이스북 메신져 봇을 개발하려면, 먼저 페이스북 계정(Facebook 계정이 없는 사용자)을 생성한다. 여러분의 계정에 로그인하여 https://www.facebook.com/pages/create/로 이동하여 새 페이지를 생성한다.

2. 우리의 경우에는 MyBook의 회사를 위한 페이지를 구축하고 있으므로, MyBooks 페이지를 호출하고 적절한 조직 유형인 Media/news company을 선택할 수 있다. 이것은 다음과 같다.

<그림>

**Page 259.**

3. 페이스북 페이지를 생성하는 두 번째 단계는 다음 스크린샷과 같이 페이스북에서 요청한 다른 세부 정보를 작성하는 것이다. 우리 페이지에 멋진 설명을 주었다(Get updates on our latest books).

<그림>

4. 제이의 모든 세부 사항을 채웠으며 MyBooks 페이스북 페이지는 준비가 되어있고 부서지기 쉬운 모습을 보인다.

<그림>

이제 좋은 시작이다. 독자는 이 페이지를 따라하기 시작할 것이지만 독자가 페이스북 페이지를 통해 대화할 수 있는 기능을 추가해야 한다. 페이스북 메신저 봇으로 이 작업을 수행한다. 이제 솔루션의 이러한 측면에 대해 연구하자.

5. 페이스북 메신져 봇을 만들려면 페이스북 앱이 필요하다. 다음 스크린샷과 같이 <https://developers.facebook.com/quickstarts/?platform>=web을 탐색하고 Skip and Create App ID를 클릭하여 앱을 생성한다.

<그림>

6. 이제 필요한 세부 정보를 입력하고 Create App ID 버튼을 눌러 앱을 생성한다. 다음 스크린샷은 앱을 생성하기 위해 추가한 세부 정보를 보여준다.

<그림>

7. 세부 정보를 입력하고 Create App ID를 클릭하자마자 새 앱이 생성된다. 이 페이스북 앱은 봇을 위한 것이다. 페이지의 오른쪽 상단에 앱 ID가 표시되지만 봇과 연결하려면 아래로 스크롤하여 메신저 섹션에서 시작하기를 클릭한다.

<그림>

8. 봇이 메신져에 액세스 할 수 있도록 다음 스크린샷과 같이 Page Access Token을 생성한다.

<<TIP시작>>

이 토큰을 안전하게 보관하고 누구와도 공유하면 안된다.

<<TIP종료>>

9. 이 토큰은 MyBooks 페이스북 페이지에서 봇과 대화를 시작한 독자에게 응답하는 데 사용된다.

<그림>

10. 좋아, 마지막으로 보류중인 것이 하나있다. 독자들로부터 메시지를 받아야한다. 그 후 독자들에게 응답할 수 있다. 이를 위해 Webhooks 절로 이동하여 몇 가지 설정을 추가한다.

-콜백 URL(Callback URL): 페이스북 페이지를 통해 독자의 메시지를 수신하는 서버에 대한 링크다.

-확인 토큰(Verification Token): 토큰을 말하면 어떤 문자 집합이라도 사용할 수 있다.

-구독 필드(Subscription Fields): 메시지를 봇의 입력 필드(subscription fields)로 선택한다(나중에 변경할 수 있음).

보이는듯이 콜백 URL이 필요하다. 콜백 URL이 제대로 설정되었는지 여부를 확인하기 위해 페이스북에서 사용된다. 이를 위해 플라스크 서버를 생성하고 콜백 URL에 사용할 경로를 구성한다. 다음 코드는 확인을 위해 콜백 URL로 사용되는 /bot이라는 경로를 생성한다.

from flask import Flask

from flask import request

import sys, requests, json, os

app = Flask(\_\_name\_\_)

@app.route("/bot/", methods=['GET', 'POST'])

def hello():

if request.method == 'GET':

return request.args.get('hub.challenge')

포트 5000에서 서버를 실행하고 ngrok를 사용하여 동일한 포트에서 실행하면 Webhook 설정으로 위치할 수 있는 콜백 URL을 얻는다. 콜백 URL의 모습은 다음과 같다.

<그림>

설정은 다음 스크린샷과 같이 버튼을 클릭하여 확인하고 저장할 수 있다.

<그림>

설정을 확인하고 저장하면 hub.challenge 코드로 플라스크 서버에 GET 요청이 전송된다. 플라스크 경로에서 이 코드를 페이스북에 반환하고 Webhook 설정을 확인한다.

/Users/chetan/book/ch09/bots/bin/python

/Users/chetan/book/ch09/bookbot.py

\* Running on <http://127.0.0.1:5000/>

(Press CTRL+C to quit)

127.0.0.1 - - [01/Oct/2016 10:17:43] "GET

/bot/?hub.mode=subscribe&hub

.challenge=1742124657&hub.verify\_token=

token HTTP/1.1" 200 –

봇이 잘 작동하려면 페이스북 페이지가 읽기 메시지 혹은 에코 메시지 같은 특정 이벤트를 허용하는지 확인해야한다. Webhooks 절에서 이러한 설정을 활성화한다.

<그림>

11. 좋아! 이제 독자들로부터 메시지를 수신하기 위해 Webhook을 사용할 준비가 되었고 또한 사용자에게 응답할 수 있는 액세스 토큰을 가진다. 아시다시피, Webhook은 봇 서버가 될 것이다! 우리의 봇이 똑똑한 것을 하도록 만들자. 다음 코드는 봇이 제이가 필요로 하는 모든 위대한 일을 할 수 있도록한다.

from flask import Flask

from flask import request

import requests, json

app = Flask(\_\_name\_\_)

def send\_weburl(payload, recipient\_id):

headers = {

"Content-Type": "application/json"

}

token = {

"access\_token":

"TOKEN"

}

if payload == 'Python':

data = json.dumps({

"recipient": {

"id": recipient\_id

},

"message":{

"attachment":{

"type":"template",

"payload":{

"template\_type":"generic",

"elements":[

"title":"Learn Python Design Patterns: Chetan

Giridhar",

"item\_url":"https://www.amazon.com/Learning-Python-

Design-Patterns-Second/dp/178588803X",

"image\_url":"https://images-na.ssl-imagesamazon.

com/images/I/51bNOsKpItL.\_SX404\_BO1,

204,203,200\_.jpg",

"subtitle":"Python Book for software architects and

developers",

"buttons":[

{

"type":"web\_url",

"url":"https://www.amazon.com/Learning-Python-

Design-Patterns-Second/dp/178588803X",

"title":"Buy",

"webview\_height\_ratio":"full"

}

]

}

]

}

}

}

})

if payload == 'Java':

data = json.dumps({

"recipient": {

"id": recipient\_id

},

"message":{

"attachment":{

"type":"template",

"payload":{

"template\_type":"generic",

"elements":[

{

"title":"RESTful Java Patterns and Best

Practices: Bhakti Mehta",

"item\_url":"https://www.amazon.com/RESTful-Java-

Patterns-Best-Practices/dp/1783287969",

"image\_url":"https://images-na.ssl-imagesamazon.

com/images/I/51YnSP6uqeL.\_SX403\_BO1,

204,203,200\_.jpg",

"subtitle":"Python Book for software architects and

developers",

"buttons":[

{

"type":"web\_url",

"url":"https://www.amazon.com/RESTful-Java-

Patterns-Best-Practices/dp/1783287969",

"title":"Buy",

"webview\_height\_ratio":"full"

}

]

}

]

}

}

}

})

r = requests.post("https://graph.facebook.com/v2.6/me/messages",

params=token, headers=headers, data=data)

def send\_postback(recipient\_id):

headers = {

"Content-Type": "application/json"

}

token = {

"access\_token":

"TOKEN"

}

data = json.dumps({

"recipient": {

"id": recipient\_id

},

"message": {

"attachment": {

"type": "template",

"payload": {

"template\_type": "button",

"text": "Hey there, Welcome to MyBooks.

What are you interested in?",

"buttons": [

{

"type":"postback",

"title":"Java",

"payload":"Java"

},

{

"type":"postback",

"title":"Python",

"payload":"Python"

}

]

}

}

}

})

r = requests.post("https://graph.facebook.com/v2.6/me/messages",

params=token, headers=headers, data=data)

@app.route("/bot/", methods=['GET', 'POST'])

def hello():

print request.data

if request.method == 'GET':

return request.args.get('hub.challenge')

data = request.get\_json()

if data["object"] == "page":

for entry in data["entry"]:

for messaging\_event in entry["messaging"]:

if messaging\_event.get("postback"):

sender\_id = messaging\_event["sender"]["id"]

payload = messaging\_event["postback"]["payload"]

send\_weburl(payload, sender\_id)

if messaging\_event.get("message"): # readers send us a

message

sender\_id = messaging\_event["sender"]["id"]

send\_postback(sender\_id)

return "ok", 200

if \_\_name\_\_ == "\_\_main\_\_":

app.run()

12. 앞의 플라스크 서버를 실행하여 봇을 활성화한다. 이제 페이스북 페이지로 이동하여 봇의 작동 방식을 살펴보자. 페이스북 페이지에서 Message를 클릭하면 MyBooks 페이지의 봇과 채팅을 시작할 수 있다.

<그림>

13. 간단한 Hi 메시지를 사용하여 봇과 대화를 시작해보자. 봇은 파이썬 책 혹은 자바 책에 대한 정보를 원하고 있는지에 대한 질문으로 응답한다. 좋다!

<그림>

**Page 270.**

14. 이제 Python을 클릭하면, 봇은 파이썬으로 작성된 건축 도서를 추천하고 독자들이 그것을 구매하도록 장려한다. 독자가 Java를 클릭할 때도 발생한다. 다음 스크린샷을 살펴본다.

<그림>

다음 스크린샷은 사용자가 Java를 선택할 때 RESTful Java Patterns and Best Practices 책을 권장하는 Java 예제를 보여준다.

<그림>

15. 멋지다, 그렇지? 이것이 제이가 필요로 하는 것이다. 따라서 독자가 MyBooks 페이지에 도착하면 봇과 대화를 하고 봇은 관심사를 기반으로 책을 추천한다. 봇의 제안은 일반적인 홍보용 전자 메일과 비교할 때 독자와 훨씬 관련이 있기 때문에 책을 구입하는 독자의 기회가 더 많다. 굉장하다!

**작동원리**

우선 제이의 출판사인 MyBooks의 페이스북 페이지를 생성했다. 이 후 페이스북의 메신저 봇과 이 페이지를 연결하고 액세스 토큰을 사용하여 봇과 채팅하는 독자에게 메시지를 전송했다. 또한 Webhook을 설정하여 봇이 독자들로부터 메시지를 받고 액세스 토큰을 사용하여 메시지를 다시 게시한다. 여기 Webhook은 봇 뒤에 숨어있는 두뇌이다.

독자가 MyBooks 페이지에 도달하면 Messenger를 클릭하여 봇과 대화를 시작한다. 그 혹은 그녀가 Hi라고 말하면 HTTP의 POST 요청은 Webhook https://2d7d823f.ngrok.io/bot/로 메시지와 함께 전송된다.

봇은 리더에서 메시지를 읽고 postback 옵션을 사용하여 generic template 메시지를 독자에게 전송한다. 봇은 페이스북의 그래프 API를 사용하여 이 메시지를 전송한다.

<<중요표시시작>>

페이스북에는 postback 메시지, 버튼, 이미지, URL 및 오디오/비디오 미디어 파일을 전송하기 위한 템플릿 메시지가 있다.

<<중요표시종료>>

독자가 Python을 선택하면 봇은 이 메시지를 수신하고 페이로드를 기반으로 도서의 이미지를 URL과 함께 반환하여 사용자가 구입할 수 있도록한다. 사용자는 Buy를 클릭하여 책의 URL로 이동하여 책을 구입할 수 있다. 정확하게 제이가 원하는 것이다.

**더 알아보기**

8장에서는 CLI, 웹 UI 및 모바일 앱을 기반으로 봇을 작성한다. 봇은 멋진 API 세트를 가진 Slack과 같은 다른 채팅 시스템에 상주할 수 있다. 하나를 쓰고 싶을 수도 있다. 당신이 하나를 쓰는다면, 나에게 포인터를 전송한다. 나는 그들을 밖으로 시도하고 싶다.

<<중요표시시작>>

트위터에서 나에게 연락하거나 직접 메시지를 전송하면 다시 연락할 것이다.

<<중요표시종료>>