란코처럼 말하는 트위터 봇 만들기

파이썬으로 트위터 봇을 만들고, 자동으로 문장을 생성하기

Otter

이 글은 독자가 파이썬의 문법을 간략하게 알고 있고, 트위터 API wrapper인 Tweepy에 대한 지식이 거의 없으며, 문장 자동 생성에 대한 기반 지식이 거의 없다는 가정하에 작성되었습니다.

이 글의 목표는 크게 두 단계로,

- 먼저 **트위터 API의 wrapper인 Tweepy를 이용하여 트윗을 작성하는 것**과,
- **작성할 간단한 문장을 생성해보는 것**입니다.

설명은 기본적으로 파이썬 3버전대와 터미널 환경을 사용하는것으로 가정하고 진행되지만, 2버전이나 다른 환경에서도 큰 문제는 없을것으로 생각됩니다.

I. Tweepy로 트윗을 하는 봇을 만들기

우리는 봇을 만들기 위해서 **Tweepy** http://www.tweepy.org/ 라는 파이썬 라이브러리를 사용할 것입니다. 먼저 라이브러리를 설치 할 필요가 있습니다.

pip를 사용하신다면

```
pip install tweepy
```

를 터미널에 입력하면 설치가 가능합니다. virtualenv와 같은 가상 개발환경에 대한 설명이나 모듈의 다른 설치 방법에 대한 설명은 여기서는 생략합니다.

라이브러리가 설치 된 후에 트위터 봇의 뼈대가 될 코드를 작성합시다. 만약 파이썬 프로그래밍에 별로 익숙하지 않다면, 처음부터 코드파일을 작성하려고 하기보다는 python IDLE을 사용하시거나, 터미널에서 python 또는 python3 명령어로 REPL에 진입한 후 다음의 과정을 따라오는것이 편할 수 있습니다.

우리가 가장 먼저 해야 할 일은, 앞에서 설치한 라이브러리를 불러 오는 것일 겁니다. 이걸 하기 위한 첫 줄의 코드는

```
import tweepy
```

정도가 될 것 입니다.

다음 단계로, 트위터 앱을 등록해야 합니다. **Twitter Developers** https://dev.twitter.com/ 에서 Manage your apps 를 찾은 후 Create New App 버튼을 눌러서 나오는 항목을 작성하면 됩니다. 여기서 앱을 등록하는 개발자 트위터 계정과 봇을 운영하는 계정은 일치 할 필요가 없습니다.

앱을 등록하고 나면 나오는 리스트에서 사용할 앱을 누르면 다양한 설정이 가능합니다. 그중에서 일단 우리는 트윗을 작성해서 내보내는 봇을 만들고자 하기 때문에 Access Level을 설정 해 줄 필요가 있습니다. 이 옵션을 최소 Read and Write로 설정 해 줍시다.

그리고 Keys and Access Tokens 탭으로 들어가면 Consumer Key와 Consumer Secret이라는 것이 나옵니다. 이는 앱의 인증과정에 사용됩니다. 우리의 코드로 돌아가서, 다음과 같이 문자열 변수로 선언해 둡시다.

```
CK = 'App consumer key'
CS = 'App secret key'
```

같은 식으로 작성하면 되겠습니다. 다만, 실제 코드로 작성할 때에는 조금 더 명시적인 변수명을(예를 들면, consumer_key라던가) 이용해 주세요. 이 글에서는 분량의 제약상 저런 표현을 사용했습니다. 또한, Consumer Secret의 값은 공개되어서는 안 되니 주의할 필요가 있습니다.

이 키들은 트위터의 인증방식인 OAuth에 사용됩니다. OAuth가 무엇인지에 대한 자세한 설명은 이 글에서 다루지는 않겠습니다. 하여튼, 이 인증 방식을 위해서 tweepy에서는 OAuthHandler를 제공합니다.

```
auth = tweepy.OAuthHandler(CK, CS)
```

인증은 위와 같이 제공되는 핸들러로 처리를 하면 됩니다. 하지만 인증이 여기서 끝나진 않습니다. 다음 단계로

```
auth.get_authorization_url()
```

의 url을 열어서 로그인을 하고, 앱 사용 승인을 하고, PIN 코드를 발급받은 후에 이를 입력해줘야 합니다. url은 print 함수로 출력해서 직접 브라우저에서 열어도 되고, 파이선에 내장된 webbrowser 모듈을 사용하여 자동으로 해당 url을 기본 브라우저로 열게 할 수도 있습니다.

그 다음으로 이 PIN 코드를

```
verifier = input()
auth.get_access_token(verifier)
```

같은 식으로 터미널에서 입력 받게 하면 됩니다. input() 함수의 경우 파이선 3버전대에서 사용 하면 되고, 2버전대에서는 raw_input()함수를 사용하면 됩니다.

마지막 단계로 얻은 액세스 토큰을 사용하여 인증을 끝마치면 됩니다.

```
AT = auth.access_token
ATS = auth.access_token_secret
auth.set_access_token(AT, ATS)
```

여기까지 하면 인증 절차는 끝이나고,

```
api = tweepy.API(auth)
```

와 같은 방식으로 이제 필요에 따라서 tweepy의 API를 호출 할 수 있습니다. 이런 인증 과정이 성가시게 느껴진다면, 이를 파일 등으로 저장해두는것도 방법이 될 수 있습니다. 그리고 앱을 등록한 계정의 액세스 토큰 값 두 가지는 Keys and Access Tokens 탭에서도 확인이 가능합니다.

마지막으로 트윗을 해봅시다.

```
api.update_status('어둠에 삼켜져라!')
```

트윗 작성외의 다른 API에 대해서는
Tweepy Documentation http://tweepy.readthedocs.io/en/ 을 참고하시길 바랍니다.

II. 자동으로 문장 생성하기

사실 자동으로 제대로 된 문장을 생성한다는 것은 매우 어려운 일입니다. 특히, 한국어의 경우 다른 언어보다 연구가 덜 이루어진 분야이기도 합니다. 또한 비교적 초심자를 위하는 취지로 작성하는 이 글에서 깊은 배경지식을 요구하는 것은 적절하지 못합니다. 따라서 아주 간단한 두가지 방법론을 소개 하고자 합니다.

하지만 그보다 앞서서 먼저, 문법에 대한 이야기를 간단하게 하고자 합니다. 우리의 평소 발화나 작문이 문법을 철저하게 지키지는 않지만, 최소한의 문법도 지키지 않으면 문장이 부자연스러워 집니다. 간단한 예시를 들자면 이렇습니다.

란코의 말은 쿠마모토 사투리려나

말은 쿠마모토 란코의 사투리려나

분명 같은 단어들을 사용하더라도 순서의 차이만으로도 의미가 달라지거나, 또는 제대로 된 의미를 갖춘 문장에서 벗어납니다.

그렇다면 어떻게 적절한 문장을 만들 수 있을까요? 크게 보자면 두가지의 방법론이 있습니다.

- **문법적으로 옳은 구조를 모델링 하고 그 모델을 사용해서 문장을 생성하기**
- **예시 문장들을 확률론적으로 해석하여 임의의 문장을 생성하기**

위의 두 방법의 가장 간단한 예시를 가지고 설명하겠습니다.

먼저 구조 모델링은 다음과 같이 이루어질 수 있습니다.

나는 학교에 간다

이 문장은 주어, 목적어, 동사의 어순을 가지고 있습니다. 우리는 이 규칙을 지키면서, 어휘를 바꿔넣는 방식으로 새로운 문장도 생성 할 수 있습니다. **나**가 아닌 **그**, **학교**가 아닌 **수렁**, **가+ㄴ다** 대신 **빠지+ㄴ다**로 치환해 보면

그는 수렁에 빠진다

라는 문장이 됩니다. 원본 문장의 문법적 형태는 유지하지만, 내용은 그와 무관한 문장을 새로 생성 할 수 있습니다. 단, 이렇게 생성된 문장은 문법적으로만 틀린 요소가 없을 뿐이지, 꼭 말이 된다는 보장은 없습니다. 같은 방식으로 생성 될 수 있는 문장에

학교는 나무에 빠진다

소는 열매에 먹힌다

같은 문장도 있는데, 이는 별로 자연스럽지 않게 느껴집니다.

예시 문장들의 확률론적 해석 중 간단한 방법은 다음과 같이 사용해 볼 수 있습니다.

나는 아이돌이 되어서 빛나고 싶어

아이돌이 만만한 직업이라고 생각해?

만만한 일이라고 생각하지 않아

라는 예시 문장들을 가지고 해석을 해보면, 우선 **아이돌**이라는 단어 뒤에는 **되어서**와 **만만한**이 오는 경우가 각각 한가지 있습니다. **만만한**이라는 단어의 뒤는 **직업이라고**와 **일이라고**가 오는 경우가 한가지씩 있습니다. **나는**의 뒤에는 **아이돌이**가, **일이라고** 다음은 **생각하지**, **생각하지** 다음에는 **않아**가 옵니다.

나는이라는 단어에서 시작해보면

나는 아이돌이 되어서 빛나고 싶어

나는 아이돌이 만만한 직업이라고 생각해?

나는 아이돌이 만만한 일이라고 생각하지 않아

와 같은 결과물이 나옵니다. 첫번째 문장은 우리가 예시로 제공한 문장과 동일하기 때문에 이상하지는 않지만, 두번째 문장은 앞과 뒤의 문맥이 달라서 자연스럽지 않습니다.

앞에서 소개한 두가지 모델의 장단점을 비교해보면 이렇습니다. 구조 모델링의 경우 확률론적 해석보다 문법적 규칙에 있어서 강력합니다. 확률론적 해석의 상대적인 장점은, 적어도 각각의 단어는 그 앞의 단어와는 유사한 맥락을 가진다는 점입니다. 구조 모델링의 경우 충분한 표본을 가지고, 형태소 분석과 병행하면 뼈대가 되는 문장 구조를 얻기에 좋아집니다. 확률론적 해석은 바로 앞의 단어에만 의존하지 않고, 앞의 n개 단어에 기반한다면 앞의 예시의 두번째 문장에서와 같이 갑자기 문맥이 달라지는 일을 다소 억제 할 수 있고, 역시 형태소 분석과 병행할 경우 동음다의어 때문에 생기는 문제를 어느정도 해결 할 수 있습니다.

그러면 우리가 문장 생성에 앞서서 트윗 작성을 하는 간단한 코드를 작성한 것처럼, 문장을 생성하는 간단하면서도 유의미한 구현체를 만들려면 어떻게 하면 좋을까요? 앞서 살펴보았듯이 문제는 크게 문법과 문맥 두 가지로 이루어집니다. 이 중 문법을 구조 모델링으로 상당 부분 해결하면, 문맥에만 집중할 수 있습니다. 그리고 일관적이고 한정된 주제와 맥락의 갖는 어휘들을 사용하면 문맥의 문제에도 도움이 됩니다. 이런 방식과 가장 가까운 실제 예시로는 주로 로봇 저널리즘이라고 불리는 알고리즘에 기반을 둔 뉴스 기사 생성이 있습니다.

그러면 우리는 기사 외의 어떤 것을 해볼 수 있을까요? 이 글의 제목으로 돌아가서, 저는 란코의 대사와 같은 문장을 만들고자 하였습니다. 다른 많은 가능성 중에 란코에 주목한 이유는 간단합니다. 평범한 문장을 템플릿으로 삼아서 새로운 문장을 만들어도, 중2병스러운 키워드 몇 개만으로도 나름의 개성이 드러나는 문장으로 바뀌게 됩니다. 그리고 어휘의 특성상 읽는 사람이 표면적인 의미를 가지고 말이 되지 않는다고 느끼는 것이 아니라, 다른 의미가 없나 나름의 추론과 해석을 시도하게 한다는데 의의가 있습니다. 요컨대 우리의 일상어보다 조금 더 문맥에서 자유로운 언어인 것입니다.

구조 모델링뿐만 아니라, 확률론적 해석도 좋은 결과물을 얻은 경우가 있습니다. 특유의 화법으로 유명한 **현 대한 민국 국가 원수** http://markov.mathnt.net/나, **미국 차기 대선후보** https:// liph.github.io/markov/에게 적용한 예시들은 정말 그럴싸해 보입니다.

추가로 조금 더 자세한 내용을 알고자 하는 분들에게 도움이 될 만한 자료나, 피드백 받은 내용등을 **https://github.com/a19641sk/RankoBot**에 올려놓을 예정입니다.

마지막으로 존경하는 검사님, 이 글은 저희집 수달이 작성했습니다. SSCC 1st