

# Ssapago



[광주 - 3반 - 4조]  
김하은 정유진 임준형

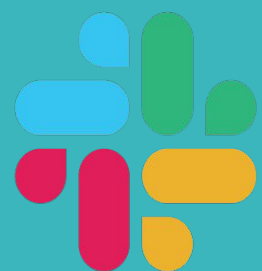
삼성 S/W Academy For Youth

# 챗봇 소개

/\* elice \*/

# 챗봇의 용도

해외기업과의 협업 중  
커뮤니케이션 과정에서 편리한  
언어번역으로 다양한 국적의  
사람들과 원활한 소통



**slack**

+

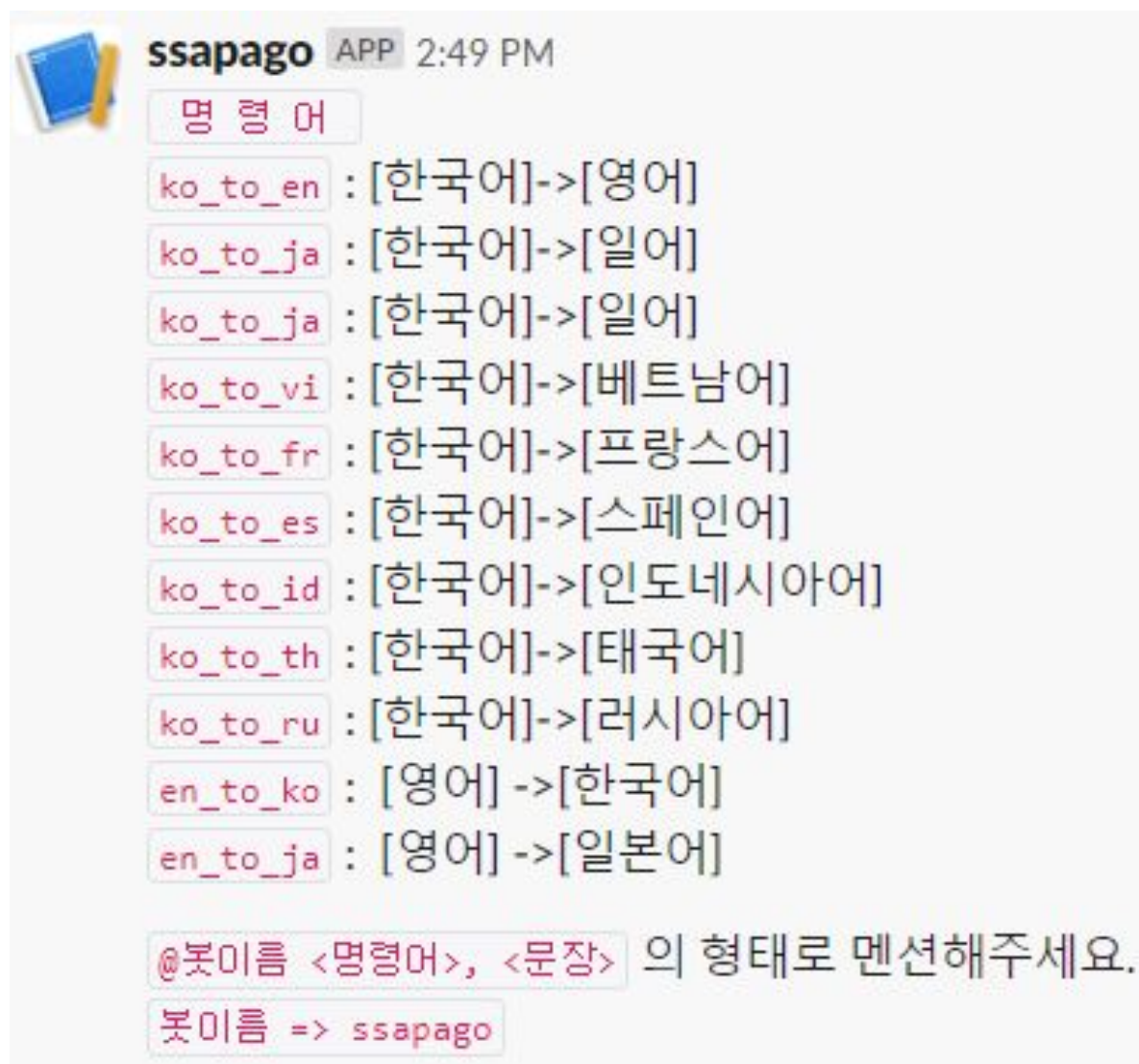


“번역”

`/* elice */`


# 기능


help 키워드





# 기능


## 언어 번역


 **임준형** 2:56 PM  
@ssapago ko\_to\_en, 나는 학교가 끝나고 친구들이랑 저녁을 먹었다.

 **ssapago** APP 2:56 PM  
I had dinner with my friends after school.

 **임준형** 2:57 PM  
@ssapago ko\_to\_ja, 나는 학교가 끝나고 친구들이랑 저녁을 먹었다.

 **ssapago** APP 2:57 PM  
私は学校が終わって友達と夕食を食べた。





 **임준형** 2:59 PM  
@ssapago en\_to\_ko, I had dinner with my friends after school.

 **ssapago** APP 2:59 PM  
나는 방과 후에 친구들과 저녁을 먹었다.

/\* elice \*/

# 기능

언어 확인

-  임준형 3:04 PM  
@ssapago ko\_to\_en, 나는 우리 반 친구들이 좋아!!! really!!
-  ssapago APP 3:04 PM  
한글만 입력하세요.
-  임준형 3:06 PM  
@ssapago en\_to\_ko, i love my friends!!!!!! 진짜루
-  ssapago APP 3:06 PM  
영어만 입력하세요.

# 차별점

사무실에 근무하는 팀원, 해외에 근무하는 팀원을 가리지 않고 신속히 커뮤니케이션을 하도록 도움을 주는 직관적이고 사용자 친화적인 인터페이스 Slack 안에서 번역기를 사용하기 때문에 빠른 언어 번역을 통한 커뮤니케이션이 가능하다.



# 예상 사용 유저

해외기업과의 커뮤니케이션  
과정에서 빠른 언어번역이  
필요한 기업의 관계자들

# 기대효과

해외기업과의 비교적 빠르고  
원활한 커뮤니케이션 가능

# 개발 과정

/\* elice \*/

# Papago NMT 번역



Papago의 인공 신경망 기반 기계 번역 기술로  
텍스트를 번역한 결과를 반환하는 RESTful API

`/* elice */`

# Papago NMT 번역 API 레퍼런스

언어 코드	언어
ko	한국어
en	영어
ja	일본어
zh-CN	중국어 간체
zh-TW	중국어 번체
vi	베트남어
id	인도네시아어
th	태국어
de	독일어
ru	러시아어
es	스페인어
it	이탈리아어
fr	프랑스어

## 참고 사항 [↗](#)

API를 요청할 때 다음 예와 같이 HTTP 요청 헤더에 **클라이언트 아이디**와 **클라이언트 시크릿**을 추가해야 합니다.

```
POST /v1/papago/n2mt HTTP/1.1
HOST: openapi.naver.com
User-Agent: curl/7.49.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Naver-Client-Id: {애플리케이션 등록 시 발급받은 클라이언트 아이디 값}
X-Naver-Client-Secret: {애플리케이션 등록 시 발급받은 클라이언트 시크릿 값}
Content-Length: 51
```

## 요청 예 [↗](#)

```
curl "https://openapi.naver.com/v1/papago/n2mt" \
-H "Content-Type: application/x-www-form-urlencoded; charset=UTF-8" \
-H "X-Naver-Client-Id: {애플리케이션 등록 시 발급받은 클라이언트 아이디 값}" \
-H "X-Naver-Client-Secret: {애플리케이션 등록 시 발급받은 클라이언트 시크릿 값}" \
-d "source=ko&target=en&text=안녕하세요." -v
```

/\* elice \*/

## 1. 코드 단축

```
if 'en_to_ko' in text:
    new_text=text.split(",")
    new_new_text= ",".join(new_text[1:])

    encText = urllib.parse.quote(new_new_text)
    data = "source=en&target=ko&text=" + encText
    url = "https://openapi.naver.com/v1/papago/n2mt"
    request = urllib.request.Request(url)
    request.add_header("X-Naver-Client-Id",client_id)
    request.add_header("X-Naver-Client-Secret",client_secret)
    response = urllib.request.urlopen(request, data=data.encode("utf-8"))
    rescode = response.getcode()
    if(rescode==200):
        response_body = response.read()
        return (response_body.decode('utf-8')[152:].split("\n")[0].replace(",
@UKZKYKWUS.", ""))
    else:
        return ("Error Code:" + rescode)
```

/\* elice \*/

```
trans = ['ko_to_en', 'en_to_ko', 'ko_to_ja', 'ko_to_vi', 'ko_to_fr', 'ko_to_es', 'ko_to_id', 'ko_to_th', 'ko_to_ru', 'en_to_ja']
```

```
def _papago(text, channel):
```

```
    new_text=text.split(",")
```

```
    idx = 0
```

```
    try:
```

```
        #idx = trans.index(new_text[0].replace("<@UKZKYKWUS> ", ""))
```

```
        idx = trans.index(new_text[0][13:])
```

```
    except:
```

```
        pass
```

```
# 중간생략
```

```
    encText = urllib.parse.quote(new_new_text)
```

```
    data = "source=" +trans[idx][:2] + "&target="+trans[idx][-2:] + "&text=" + encText
```

```
    url = "https://openapi.naver.com/v1/papago/n2mt"
```

```
    request = urllib.request.Request(url)
```

```
    request.add_header("X-Naver-Client-Id",client_id)
```

```
    request.add_header("X-Naver-Client-Secret",client_secret)
```

```
    response = urllib.request.urlopen(request, data=data.encode("utf-8"))
```

```
    rescode = response.getcode()
```

```
    if(rescode==200):
```

```
        response_body = response.read()
```

```
        return (response_body.decode('utf-8')[152:].split("\n")[0].replace(", @UKZKYKWUS.", ""))
```

```
    else:
```

```
        return ("Error Code:" + rescode)
```

```
/* elice */
```

## 2. input

```
if trans[idx] in text:
    new_new_text = ",".join(new_text[1:])

    encount = 0
    kocount = 0

    if trans[idx][:2] == 'ko':
        for i in new_new_text:
            if ord('a') <= ord(i) <= ord('z'):
                encount = encount + 1
    elif trans[idx][:2] == 'en':
        for i in new_new_text:
            if ord('가') <= ord(i) <= ord('힉'):
                kocount = kocount + 1

    if encount != 0:
        return "한글만 입력하세요."

    elif kocount !=0:
        return "영어만 입력하세요."

    else: #이하 생략
```

/\* elice \*/



### 3. thread

```
@slack_events_adaptor.on("app_mention")
def app_mentioned(event_data):
    channel = event_data["event"]["channel"]
    text = event_data["event"]["text"]
    message = _papago(text, channel)

    # 3초 이상 시 recall 되는 문제 (지연없이 동작하도록 쓰레드 이용)
    send_message_thread = Thread(target=send_message, args=(channel, message))
    send_message_thread.start()

def send_message(channel, message):
    slack_web_client.chat_postMessage(
        channel=channel,
        text=message
    )

def send_message_using_blocks(channel, my_blocks):
    slack_web_client.chat_postMessage(
        channel=channel,
        blocks=extract_json(my_blocks)
    )
```

/\* elice \*/

# 결과

/\* elice \*/

# 전체코드

```
chatbot.py ▸ _papago
1  # -*- coding: utf-8 -*-
2  import os
3  import sys
4  import re
5  import urllib.request
6
7  from flask import Flask
8  from slack import WebClient
9  from slackeventsapi import SlackEventAdapter
10
11 from threading import Thread
12
13 # 블록 클래스를 사용하는 데 필요한 import 문입니다.
14 from slack.web.classes import extract_json
15 from slack.web.classes.blocks import *
16
17 # auth_info
18 SLACK_TOKEN = 'xoxb-677130169825-692392862112-VtgUzrR4qT7tnJ770Hsdg8N6'
19 SLACK_SIGNING_SECRET = 'c725eb3defee4faf8ac188c949159e85'
20 client_id = "O1F4aJTMecfTU8KrqzNS"
21 client_secret = "uzWJPp6XgS"
22
23 app = Flask(__name__)
24 # /listening 으로 슬랙 이벤트를 받습니다.
25 slack_events_adaptor = SlackEventAdapter(SLACK_SIGNING_SECRET, "/listening", app)
26 slack_web_client = WebClient(token=SLACK_TOKEN)
27
```

/\* elice \*/

# 전체코드

```
# (이하 2019-07-12 수정) 코드 최적화
trans = ['ko_to_en', 'en_to_ko', 'ko_to_ja', 'ko_to_vi', 'ko_to_fr', 'ko_to_es', 'ko_to_id', 'ko_to_th', 'ko_to_ru', 'en_to_ja']
def _papago(text, channel):
    new_text=text.split(",")
    idx = 0
    try:
        #idx = trans.index(new_text[0].replace("<@UKZKYKWUS> ", ""))
        idx = trans.index(new_text[0][13:])
    except:
        pass

    if trans[idx] in text:
        new_new_text = ",".join(new_text[1:])

        encount = 0
        kocount = 0

        if trans[idx][:2] == 'ko':
            for i in new_new_text:
                if ord('a') <= ord(i) <= ord('z'):
                    encount = encount + 1
        elif trans[idx][:2] == 'en':
            for i in new_new_text:
                if ord('가') <= ord(i) <= ord('힉'):
                    kocount = kocount + 1

        if encount != 0:
            return "한글만 입력하세요."

        elif kocount !=0:
            return "영어만 입력하세요."
```

/\* elice \*/



# 전체코드

```
61         else:
62
63             encText = urllib.parse.quote(new_new_text)
64             data = "source=" + trans[idx][:2] + "&target="+trans[idx][-2:] + "&text=" + encText
65             url = "https://openapi.naver.com/v1/papago/n2mt"
66             request = urllib.request.Request(url)
67             request.add_header("X-Naver-Client-Id",client_id)
68             request.add_header("X-Naver-Client-Secret",client_secret)
69             response = urllib.request.urlopen(request, data=data.encode("utf-8"))
70             rescode = response.getcode()
71             if(rescode==200):
72                 response_body = response.read()
73                 return (response_body.decode('utf-8')[152:].split("\n")[0].replace(", @UKZKYKWUS.", ""))
74             else:
75                 return ("Error Code:" + rescode)
76
77     elif 'help' in text:
78         block1 = SectionBlock(
79             text="`명령어`"+ "\n"+
80             "`ko_to_en` : [한국어]->[영어]"+ "\n"+
81             "`ko_to_ja` : [한국어]->[일본어]"+ "\n"+
82             "`ko_to_ja` : [한국어]->[일본어]"+ "\n"+
83             "`ko_to_vi` : [한국어]->[베트남어]"+ "\n"+
84             "`ko_to_fr` : [한국어]->[프랑스어]"+ "\n"+
85             "`ko_to_es` : [한국어]->[스페인어]"+ "\n"+
86             "`ko_to_id` : [한국어]->[인도네시아어]"+ "\n"+
87             "`ko_to_th` : [한국어]->[태국어]"+ "\n"+
88             "`ko_to_ru` : [한국어]->[러시아어]"+ "\n"+
89             "`en_to_ko` : [영어] ->[한국어]"+ "\n"+
90             "`en_to_ja` : [영어] ->[일본어]"
91         )
```

/\* elice \*/

## 전체코드

```
95     block2 = SectionBlock(  
96         #text="`@`"+ "`ssapago <명령어>, <문장>` 의 형태로 멘션해주세요."  
97         text="`<@봇이름> <명령어>, <문장>` 의 형태로 멘션해주세요."+ "\n`봇이름 => ssapago`"  
98     )  
99     my_blocks = [block1,block2]  
100     slack_web_client.chat_postMessage(  
101         channel=channel,  
102         blocks=extract_json(my_blocks)  
103     )  
104 else:  
105     return "`@<봇이름> help` 와 같이 멘션해주세요."  
106
```

/\* elice \*/



## 전체코드

```
107 # 챗봇이 멘션을 받았을 경우
108 @slack_events_adaptor.on("app_mention")
109 def app_mentioned(event_data):
110     channel = event_data["event"]["channel"]
111     text = event_data["event"]["text"]
112
113     message = _papago(text, channel)
114     # print(channel)
115     # 3초 이상 시, recall 방지
116     send_message_thread = Thread(target=send_message, args=(channel, message))
117     send_message_thread.start()
118
119 def send_message(channel, message):
120     slack_web_client.chat_postMessage(
121         channel=channel,
122         text=message
123     )
124
125 def send_message_using_blocks(channel, my_blocks):
126     slack_web_client.chat_postMessage(
127         channel=channel,
128         blocks=extract_json(my_blocks)
129     )
130 # / 로 접속하면 서버가 준비되었다고 알려줍니다.
131 @app.route("/", methods=["GET"])
132 def index():
133     return "<h1>Server is ready.</h1>"
134
135
136 if __name__ == '__main__':
137     app.run('0.0.0.0', port=5000)
```

/\* elice \*/

# 개선사항

/\* elice \*/



# 개선사항

- “다른 국적 언어 -> 다른 국적 언어”

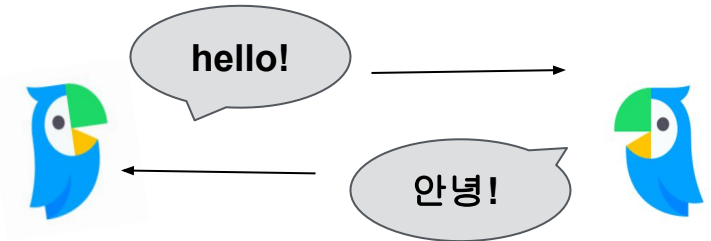


기존의 기능  
: 주로  
‘한국어’ ->  
‘타국 언어’

/\* elice \*/

# 만들어 보고 싶은 '챗봇'

## - “자동 번역 챗봇 커뮤니케이션”



: 다양한 국적의 언어를 이용한 챗봇들끼리 소통 시스템

## - “식사 메뉴 알려주는 챗봇”

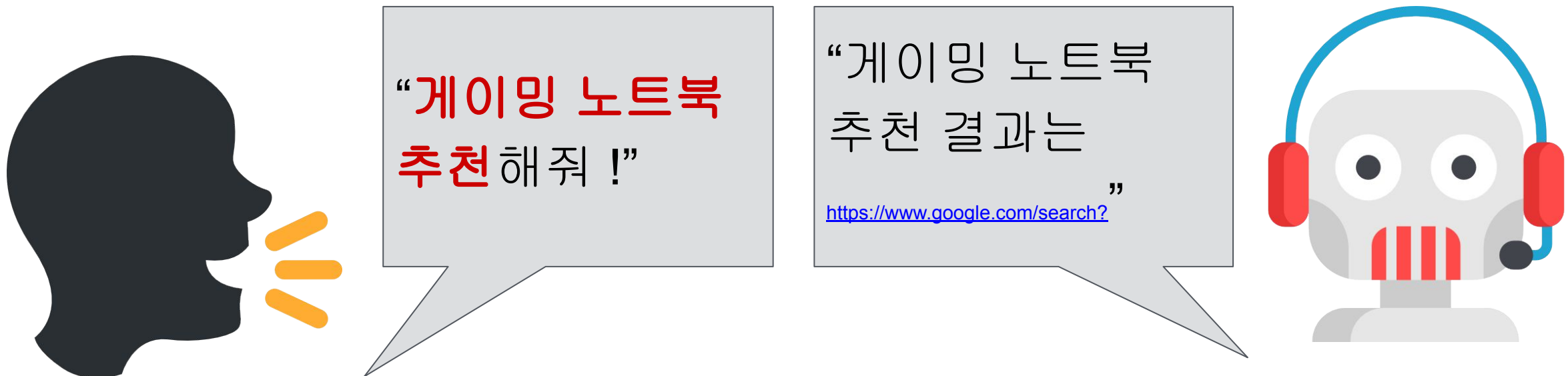


: ‘점심 메뉴’ 키워드를 통해 맛있는 SSAFY 점심 메뉴를 알려주는 챗봇

# 만들어 보고 싶은 '챗봇'

---

- “멘션 키워드를 분석하여 관련된 정보를 사용자에게 알려주는 챗봇”



감사합니다

/\* elice \*/