유튜버 음성분석을 통한 분야별 음성 특성 연구

[아이디어 구체화 과정]

1. 유튜버들은 보통 대중들이 원하는 행동을 하는 경우가 있다.

2. 대중들이 원하는 행동을 할 때 더욱더 많은 구독자를 보유할 수 있기 때문.

3. 유튜버들은 다양한 분야에서 활동한다.

4. 강연유튜브 채널에서는 설득력과 신뢰감있는 연설, 먹방은 신나고 업된 느낌, 일상유튜버는 편안한 느낌으로 행동한다.

5. 이 중 강연 유튜브 채널인 'TED'를 선택하여 데이터를 분석하였다.

6. TED의 수많은 강연을 조회수로 정렬하여 사람들에게 지지를 많이 받은 동영상을 분석하였다.

[데이터 분석 과정]

1. 영상의 타이틀과 링크 크롤링

2. PyTube API로 영상 다운로드

3. ffmpeg코덱과 subprocess모듈 이용하여 mp4파일을 mp3로 변환

4. 오픈소스 참고하여 음성파일을 음성파형 이미지로 변환.

---

[1] 영상 타이틀, 링크 크롤링 - BeautifulSoup API

In [4]:
```
import requests
from bs4 import BeautifulSoup as bs
```

In [5]:
```
html = requests.get("https://www.youtube.com/user/TEDtalksDire
soup = bs(html, 'html.parser')
```

In [3]:
```
print(bs.prettify(soup))
```

```
<!DOCTYPE html>
<html data-cast-api-enabled="true" lang="ko">
<head>
<style name="www-roboto">
@font-face{font-family:'Roboto';font-style:italic;font-weig
ht:500;src:local('Roboto Medium Italic'),local('Roboto-Medi
umItalic'),url(//fonts.gstatic.com/s/roboto/v18/KFOjCnqEu92
Fr1Mu51S7ACc6CsE.ttf)format('truetype');}@font-face{font-fa
mily:'Roboto';font-style:italic;font-weight:400;src:local
('Roboto Italic'),local('Roboto-Italic'),url(//fonts.gstati
c.com/s/roboto/v18/KFOkCnqEu92Fr1Mu51xIIzc.ttf)format('true
type');}@font-face{font-family:'Roboto';font-style:normal;f
ont-weight:500;src:local('Roboto Medium'),local('Roboto-Med
ium'),url(//fonts.gstatic.com/s/roboto/v18/KFOlCnqEu92Fr1Mm
EU9fBBc9.ttf)format('truetype');}@font-face{font-family:'Ro
boto';font-style:normal;font-weight:400;src:local('Roboto R
```

In [75]:
```
# 링크와 타이틀 가져오기
videos = soup.select('a.yt-uix-tile-link')
titles = []
for i in range(len(videos)):
titles.append(videos[i].get_text())
titles[i] = titles[i].replace('|', '')
titles[i] = titles[i].replace('?', '')
titles[i] = titles[i].replace('\'', '')
titles[i] = titles[i].replace(',', '')
titles[i] = titles[i].replace('.', '')
titles[i] = titles[i].replace(' ', '_')
# url과 추출한 link 합치기
videolist = []
for v in videos:
tmp = 'https://www.youtube.com' + v['href']
videolist.append(tmp)
```

In [76]:
```
print(videolist)
```

```
['https://www.youtube.com/watch?v=_QdPW8JrYzQ', 'https://ww
w.youtube.com/watch?v=eIho2S0ZahI', 'https://www.youtube.co
m/watch?v=arj7oStGLkU', 'https://www.youtube.com/watch?v=DF
jIi2hxxf0', 'https://www.youtube.com/watch?v=KM4Xe6Dlp0Y',
 'https://www.youtube.com/watch?v=iG9CE55wbtY', 'https://ww
w.youtube.com/watch?v=Dceyy0cX6J4', 'https://www.youtube.co
m/watch?v=Ks-_Mh1QhMc', 'https://www.youtube.com/watch?v=P_
```

```
6vDLq64gE', 'https://www.youtube.com/watch?v=GZGY0wPAnus',
 'https://www.youtube.com/watch?v=XFnGhrC_3Gs', 'https://ww
w.youtube.com/watch?v=xYemnKEKx0c', 'https://www.youtube.co
m/watch?v=8KkKuTCFvzI', 'https://www.youtube.com/watch?v=Cp
c-t-Uwv1I', 'https://www.youtube.com/watch?v=9kxL9Cf46VM',
 'https://www.youtube.com/watch?v=iCvmsMzlF7o', 'https://ww
w.youtube.com/watch?v=7jx0dTYUO5E', 'https://www.youtube.co
m/watch?v=qp0HIF3SfI4', 'https://www.youtube.com/watch?v=gn
uFrtTNUTc', 'https://www.youtube.com/watch?v=8jPQjjsBbIc',
 'https://www.youtube.com/watch?v=zIwLWfaAg-8', 'https://ww
w.youtube.com/watch?v=H_8y0WLm78U', 'https://www.youtube.co
m/watch?v=c0KYU2j0TM4', 'https://www.youtube.com/watch?v=Gi
gYWy2UmOY', 'https://www.youtube.com/watch?v=w2itwFJCgFQ',
 'https://www.youtube.com/watch?v=rrkrvAUbU9Y', 'https://ww
w.youtube.com/watch?v=BdHK_r9RXTc', 'https://www.youtube.co
m/watch?v=PdxPCeWw75k', 'https://www.youtube.com/watch?v=Rc
GyVTAoXEU', 'https://www.youtube.com/watch?v=CDsNZJTWw0w']
```

In [74]:
```
print(titles)
```

```
['This_is_what_happens_when_you_reply_to_spam_email__James_Ve
itch', 'How_to_speak_so_that_people_want_to_listen__Julian_Tr
easure', 'Inside_the_mind_of_a_master_procrastinator__Tim_Urb
an', 'The_orchestra_in_my_mouth__Tom_Thum', 'Looks_arent_ever
ything_Believe_me_Im_a_model__Cameron_Russell', 'Do_schools_k
ill_creativity__Sir_Ken_Robinson', 'The_agony_of_trying_to_un
subscribe__James_Veitch', 'Your_body_language_may_shape_who_y
ou_are__Amy_Cuddy', 'How_to_spot_a_liar__Pamela_Meyer', 'The_
art_of_misdirection__Apollo_Robbins', 'How_I_held_my_breath_f
or_17_minutes__David_Blaine', 'Strange_answers_to_the_psychop
ath_test__Jon_Ronson', 'What_makes_a_good_life_Lessons_from_t
he_longest_study_on_happiness__Robert_Waldinger', 'Why_we_do_
what_we_do__Tony_Robbins', 'A_Saudi_an_Indian_and_an_Iranian_
walk_into_a_Qatari_bar___Maz_Jobrani', 'The_power_of_vulnerab
ility__Brené_Brown', '10_things_you_didnt_know_about_orgasm__
Mary_Roach', 'How_great_leaders_inspire_action__Simon_Sinek',
'My_journey_to_yo-yo_mastery__BLACK', 'How_to_stay_calm_when_
you_know_youll_be_stressed__Daniel_Levitin', 'The_future_were
_building_--_and_boring__Elon_Musk', 'The_price_of_shame__Mon
ica_Lewinsky', 'The_power_of_introverts__Susan_Cain', 'Brain_
magic__Keith_Barry', 'The_astounding_athletic_power_of_quadco
pters__Raffaello_DAndrea', 'The_puzzle_of_motivation__Dan_Pin
k', 'Reggie_Watts_disorients_you_in_the_most_entertaining_wa
y', 'My_escape_from_North_Korea__Hyeonseo_Lee', 'How_to_make_
stress_your_friend__Kelly_McGonigal', 'New_bionics_let_us_run
_climb_and_dance__Hugh_Herr']
```

[2] 영상 다운로드 - PyTube API

In [7]:
```
!pip install pytube
```

```
Requirement already satisfied: pytube in c:\anaconda3\envs\ki
rbs_pt\lib\site-packages (9.5.1)
```

In [18]:
```
import pytube
```

In [60]:
```
for i in range(len(videolist)):
# 영상 가져오기
youtube = pytube.YouTube(videolist[i])
videos = youtube.streams.filter(file_extension='mp4').all()
#videos = youtube.streams.all()
print(i, videolist[i])
# 저장할 파일 경로
parent_dir = "D:\KIRBS_PT_2\download"
videos[0].download(parent_dir, titles[i])
```

```
0 https://www.youtube.com/watch?v=_QdPW8JrYzQ
1 https://www.youtube.com/watch?v=eIho2S0ZahI
2 https://www.youtube.com/watch?v=arj7oStGLkU
3 https://www.youtube.com/watch?v=DFjIi2hxxf0
4 https://www.youtube.com/watch?v=KM4Xe6Dlp0Y
5 https://www.youtube.com/watch?v=iG9CE55wbtY
6 https://www.youtube.com/watch?v=Dceyy0cX6J4
7 https://www.youtube.com/watch?v=Ks-_Mh1QhMc
8 https://www.youtube.com/watch?v=P_6vDLq64gE
9 https://www.youtube.com/watch?v=GZGY0wPAnus
10 https://www.youtube.com/watch?v=XFnGhrC_3Gs
11 https://www.youtube.com/watch?v=xYemnKEKx0c
12 https://www.youtube.com/watch?v=8KkKuTCFvzI
13 https://www.youtube.com/watch?v=Cpc-t-Uwv1I
14 https://www.youtube.com/watch?v=9kxL9Cf46VM
15 https://www.youtube.com/watch?v=iCvmsMzlF7o
```

[3] mp4를 mp3로 변환하기

In [36]:
```
!cd D:\KIRBS_PT_2\download
```

In [70]:
```
import subprocess
import os
default_filename = []
new_filename = []
for i in range(len(videolist)):
default_filename.append(titles[i] + ".mp4")
new_filename.append(titles[i] + ".mp3")
origin_filename = str(default_filename[i])
mp3_filename = str(new_filename[i])
origin_path = os.path.join(parent_dir, origin_filename)
mp3_path = os.path.join(parent_dir, mp3_filename)
# mp4를 mp3로 변환
```

```
subprocess.Popen(['ffmpeg', '-i', origin_path, mp3_path])
print(i, mp3_filename)
print('done')
```

```
12 What_makes_a_good_life_Lessons_from_the_longest_study_on
_happiness__Robert_Waldinger.mp3
13 Why_we_do_what_we_do__Tony_Robbins.mp3
14 A_Saudi_an_Indian_and_an_Iranian_walk_into_a_Qatari_bar_
__Maz_Jobrani.mp3
15 The_power_of_vulnerability__Brené_Brown.mp3
16 10_things_you_didnt_know_about_orgasm__Mary_Roach.mp3
17 How_great_leaders_inspire_action__Simon_Sinek.mp3
   My_journey_to_yo-yo_mastery__BLACK.mp3
19 How_to_stay_calm_when_you_know_youll_be_stressed__Daniel
_Levitin.mp3
20 The_future_were_building_--_and_boring__Elon_Musk.mp3
21 The_price_of_shame__Monica_Lewinsky.mp3
22 The_power_of_introverts__Susan_Cain.mp3
23 Brain_magic__Keith_Barry.mp3
24 The_astounding_athletic_power_of_quadcopters__Raffaello_
DAndrea.mp3
```

[4] 오디오 파형 이미지 그리기 - Pydub module과 오픈소스 활용

In [26]:
```
!pip install pydub
```

```
Collecting pydub
Downloading https://files.pythonhosted.org/packages/79/db/eaf
620b73a1eec3c8c6f8f5b0b236a50f9da88ad57802154b7ba7664d0b8/pyd
ub-0.23.1-py2.py3-none-any.whl
Installing collected packages: pydub
Successfully installed pydub-0.23.1
```

In [40]:
```
!cd D:\KIRBS_PT_2\download
```

In [71]:
```
import sys
from pydub import AudioSegment
from PIL import Image, ImageDraw
class Waveform(object):
bar_count = 107 # 음원 파형을 그리는데 사용할 총 막대기 개수
db_ceiling = 60 # 음원 파형들의 높이를 평준화(nomalize)시키는 값
def __init__(self, filename):
# 오디오파일의 위치를 받는다.
self.filename = filename
# AudioSegment 객체 생성(오디오파일 위치, 파일확장자)
audio_file = AudioSegment.from_file(self.filename, self.filenar
# 각 오디오 파형 막대의 높이를 계산할 때 사용될 볼륨 피크 값.(Array 데이터)
self.peaks = self._calculate_peaks(audio_file)
```

```
# ------------------------------------ #
# 구간별 볼륨값 데이터를 계산하는 메서드 #
# ------------------------------------ #
# 각각의 파형 막대의 높이를 계산하는데 쓰이는 볼륨값 데이터를 리스트로 리턴.
def _calculate_peaks(self, audio_file):
# 음원 파일의 길이를 막대의 개수로 나누어서 막대하나가 나타내게 될 음원의 부분
chunk_length = len(audio_file) / self.bar_count
# 각 부분의 소리 크기(rms)값을 리스트에 저장.
loudness_of_chunks = [
audio_file[i * chunk_length: (i + 1) * chunk_length].rms
for i in range(self.bar_count)]
# 구해진 리스트에서 가장 큰 값을 max_rms 값에 저장 후, 실수값으로 변경.
max_rms = max(loudness_of_chunks) * 1.00
# 각 막대의 소리크기를 최대값에 대한 비율로 맞춰주고, 평준화 시킨 후 정수로 변경
return [int((loudness / max_rms) * self.db_ceiling)
for loudness in loudness_of_chunks]
# 막대 이미지를 그리는 메서드
def _get_bar_image(self, size, fill):
# 전체 넓이와 전체 높이
width, height = size
bar = Image.new('RGBA', size, fill)
end = Image.new('RGBA', (width, 2), fill)
draw = ImageDraw.Draw(end)
draw.point([(0, 0), (3, 0)], fill='#c1c1c1')
draw.point([(0, 1), (3, 1), (1, 0), (2, 0)], fill='#555555')
bar.paste(end, (0, 0))
bar.paste(end.rotate(180), (0, height - 2))
return bar
# 막대 이미지들을 합치는 메서드
def _generate_waveform_image(self):
# 전체 넓이와 전체 높이를 가지는 비어있는 이미지 생성
im = Image.new('RGB', (840, 128), '#f5f5f5')
# 각 요소들을 하나씩 순회
for index, value in enumerate(self.peaks, start=0):
# 막대를 붙여넣기 할 X축과 Y축 위치 지정
column = index * 8 + 2
upper_endpoint = 64 - value
# 막대 넓이 값과 볼륨 크기 데이터의 두배 값을 높이 값으로 전달해주어 막대 생성
# 생성한 막대를 X, Y좌표에 붙여넣음
im.paste(self._get_bar_image((4, value * 2), '#424242'), (colum
return im
def save(self):
png_filename = self.filename.replace(
self.filename.split('.')[-1], 'png')
with open(png_filename, 'wb') as imfile:
self._generate_waveform_image().save(imfile, 'PNG')
```

In [72]:
```
for i in range(len(titles)):
mp3_filename = str(new_filename[i])
mp3_path = os.path.join(parent_dir, mp3_filename)
if __name__ == '__main__':
filename = sys.argv[1]
waveform = Waveform(mp3_path)
waveform.save()
```

In [ ]: