

주간 활동 보고서

목차

1. 활동시간 및 내용

- 스마트업 회의 및 활동 진행 시간
- 회의 및 활동 내용

2. 13주차 진행사항

- STT팀
- 요약알고리즘팀

3. 추후 일정

- 앞으로 남은 기간 동안의 일정

<주간 활동 보고서 13 주차> -2024.05.27(월) ~ 2024.06.02(일)-

1. 활동시간 및 내용

<스마트업 회의 및 활동 진행 시간>

-오프라인 회의 및 활동 시간

5 월 28 일(화) / 16 시~17 시 40 분 (1 시간 40 분) 오프라인

5 월 29 일(수) / 12 시~20 시 30 분 (8 시간 30 분) 오프라인

총 : 10 시간

-회의 내용

>> [스마트업] 13 주차 회의 및 활동 내용

-STT 코드 작성한 부분 라즈베리파이에서 작동 확인.

-평가에 대한 회의 진행

원래는 13 주차에 평가를 진행하려 했으나. 물품 배송이 늦어지는 관계로 14 주차에 평가를 진행하기로 함.

-13 주차 중간발표 영상 촬영&편집

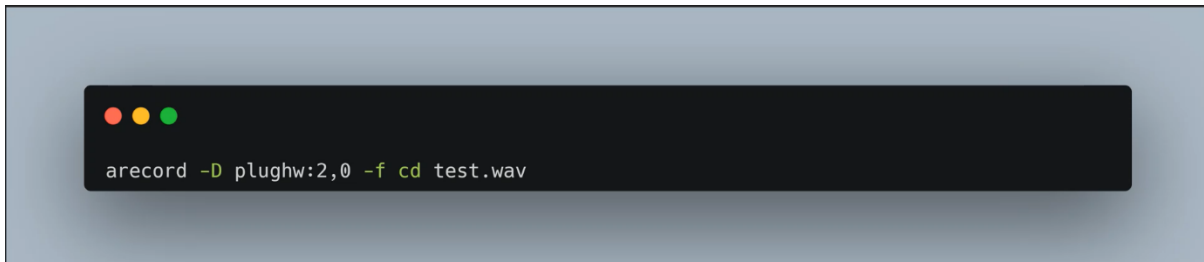
-음성을 STT 로 변환해주는 코드를 사용해
음성을 STT 로 변환하고 요약알고리즘 코드를
이용해 요약 테스트 진행

-STT & 요약알고리즘 결합 코드 보완

2. 13주차 진행사항

- 13 주차 진행사항에 대한 팀 별 정리 내용

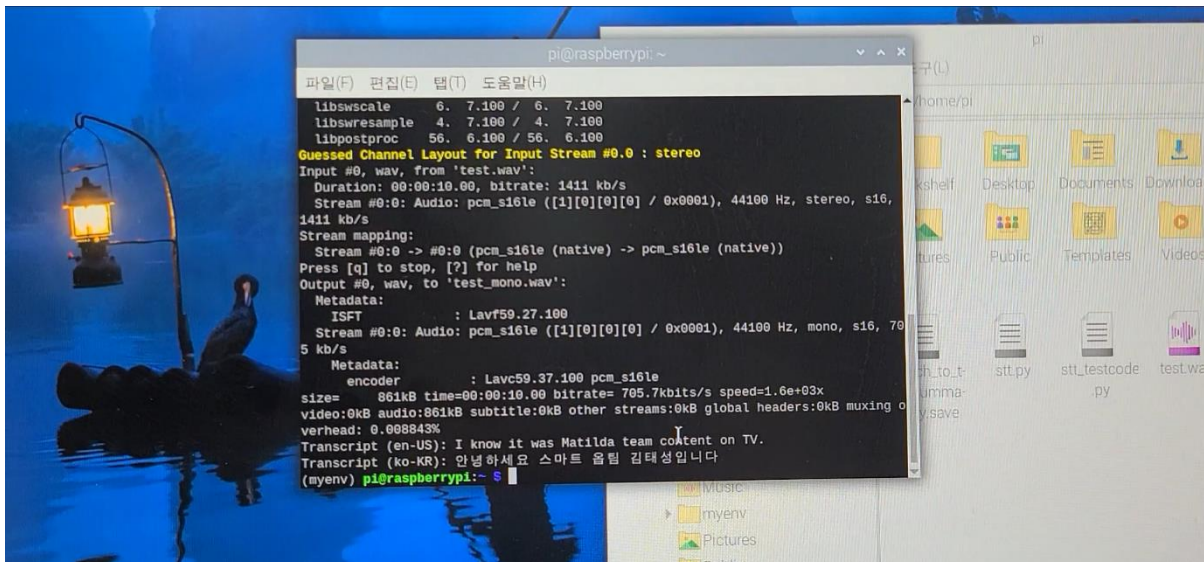
<STT 팀>



음성을 STT 로 변환하기 위해 음성녹음을 진행하는 코드입니다.

plughw 번호 2 > 사운드카드 , plughw 번호 0 > 마이크 /

코드의 hw 부분에 사운드카드와 마이크의 번호를 입력해 음성 녹음을 진행하는 과정입니다.



마이크로 음성을 입력하고 입력된 음성을 샘플링하여 텍스트로 도출한 결과가 나온 화면입니다.

```

import subprocess
import io
from google.cloud import speech
from google.oauth2 import service_account

def convert_to_mono(input_file, output_file):
    # ffmpeg를 사용하여 스테레오 파일을 모노로 변환
    command = ['ffmpeg', '-i', input_file, '-ac', '1', output_file]
    subprocess.run(command, check=True)

def transcribe_speech(file_path, language_codes):
    credentials =
service_account.Credentials.from_service_account_file('/home/pi/capston.json')
    client = speech.SpeechClient(credentials=credentials)

    with io.open(file_path, 'rb') as audio_file:
        content = audio_file.read()

    audio = speech.RecognitionAudio(content=content)

    for language_code in language_codes:
        config = speech.RecognitionConfig(
            encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
            sample_rate_hertz=44100,
            language_code=language_code
        )

        response = client.recognize(config=config, audio=audio)

        for result in response.results:
            print(f'Transcript ({language_code}): {result.alternatives[0].transcript}')

if __name__ == '__main__':
    input_file = 'test.wav'
    output_file = 'test_mono.wav'

    # 오디오 파일을 모노로 변환
    convert_to_mono(input_file, output_file)

    # 영어와 한국어로 변환
    transcribe_speech(output_file, language_codes=['en-US', 'ko-KR'])

```

입력된 음성을 STT 로 변환해주는 전체 코드입니다.

먼저, 이 코드가 어떤 일을 하는지 전체적인 부분에 대한 간략한 설명입니다.

1. 녹음한 음성 오디오(wav) 파일을 라즈베리파이 환경에 적용시킬 수 있도록 모노(mono) 오디오 파일로 변환합니다.

2. 변환된 모노(mono) 오디오 파일을 구글 클라우드 콘솔의 STT API 를 활용해 한국어와 영어로 변환합니다.

```
def convert_to_mono(input_file, output_file):  
    command = ['ffmpeg', '-i', input_file, '-ac', '1', output_file]  
    subprocess.run(command, check=True)
```


다음의 함수 코드를 이용해 녹음한 음성(wav) 파일을 모노(mono) 음성 파일로 변환합니다.

```
def transcribe_speech(file_path, language_codes):  
    credentials =  
    service_account.Credentials.from_service_account_file('/home/pi/capston.json')  
    client = speech.SpeechClient(credentials=credentials)  
  
    with io.open(file_path, 'rb') as audio_file:  
        content = audio_file.read()  
  
    audio = speech.RecognitionAudio(content=content)  
  
    for language_code in language_codes:  
        config = speech.RecognitionConfig(  
            encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,  
            sample_rate_hertz=44100,  
            language_code=language_code  
        )  
  
        response = client.recognize(config=config, audio=audio)  
  
        for result in response.results:  
            print(f'Transcript ({language_code}): {result.alternatives[0].transcript}')
```

이 부분의 코드는 구글 클라우드 음성인식 STT API 를 사용하여 오디오 파일을 텍스트로 변환하는 코드입니다.

이 함수는 다음과 같은 과정으로 작동합니다.

1. 서비스 계정 키 파일(json)을 사용하여 구글 클라우드 STT API 인증 정보를 설정합니다.
2. 오디오 파일을 바이너리 모드로 읽어옵니다.
3. 오디오 데이터를 객체로 변환한 후 , 주어진 언어 코드에 대해 음성 인식 설정을 구성합니다.
4. 구글 클라우드 음성 인식 API를 호출하여 오디오를 텍스트로 변환합니다.
5. 변환된 텍스트를 출력합니다.



```
if __name__ == '__main__':  
    input_file = 'test.wav'  
    output_file = 'test_mono.wav'  
  
    convert_to_mono(input_file, output_file)  
  
    transcribe_speech(output_file, language_codes=['en-US', 'ko-KR'])
```

코드의 메인 함수입니다. 이 부분에서는 앞서 정의한 2 가지 기능들을 순서대로 호출합니다.

이 함수는 'test.wav' 음성 파일을 모노(mono)파일로 변환하고 변환된 'test_mono.wav' 파일을 한국어와 영어로 변환합니다.

이러한 과정을 통해 음성 파일을 STT 로 변환하게 됩니다.

<요약알고리즘팀>

이번 요약 알고리즘 개발에 이어 STT 와 결합한 프로그램입니다.

```
1  import os
2  import subprocess
3  from google.cloud import speech
4  import io
5  from google.oauth2 import service_account
6
7  from konlpy.tag import Kkma
8  from konlpy.tag import Okt
9  from sklearn.feature_extraction.text import TfidfVectorizer
10 from sklearn.feature_extraction.text import CountVectorizer
11 from sklearn.preprocessing import normalize
12 import numpy as np
13
14 def convert_to_mono(input_file, output_file):
15     # ffmpeg를 사용하여 스테레오 파일을 모노로 변환
16     command = ['ffmpeg', '-i', input_file, '-ac', '1', output_file]
17     subprocess.run(command, check=True)
18
```

먼저, convert_to_mono 함수입니다. 이 함수는 음성 인식의 정확도를 높이기 위해 ffmpeg 를 사용하여 스테레오 오디오 파일을 모노로 변환합니다.

```
19 def transcribe_speech(file_path, language_code='ko-KR'):
20     credentials = service_account.Credentials.from_service_account_file('/home/pi/capston.json')
21     client = speech.SpeechClient(credentials=credentials)
22
23     with io.open(file_path, 'rb') as audio_file:
24         content = audio_file.read()
25
26     audio = speech.RecognitionAudio(content=content)
27     config = speech.RecognitionConfig(
28         encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
29         sample_rate_hertz=44100,
30         language_code=language_code
31     )
32
33     response = client.recognize(config=config, audio=audio)
34
35     full_transcript = ""
36     for result in response.results:
37         full_transcript += result.alternatives[0].transcript
38
```

그 다음으로 transcribe_speech 함수는 Google Cloud Speech-to-Text API 를 사용하여 오디오 파일을 텍스트로 변환합니다.

이 부분에서는 Google Cloud 인증서와 언어 코드를 설정하고, API 를 통해 오디오 파일의 내용을 텍스트로 변환하여 저장합니다.


```

39     save_transcript_in_chunks(full_transcript, 1500, 'transcript')
40
41 def save_transcript_in_chunks(text, chunk_size, base_filename):
42     # 텍스트를 chunk_size 만큼 나눠서 저장
43     for i in range(0, len(text), chunk_size):
44         chunk = text[i:i + chunk_size]
45         filename = f"{base_filename}_{i // chunk_size + 1}.txt"
46         with open(filename, 'w', encoding='utf-8') as f:
47             f.write(chunk)
48

```

변환된 텍스트는 save_transcript_in_chunks 함수를 통해 일정한 크기로 나누어 저장합니다.

이는 텍스트 파일이 너무 길어지는 것을 방지하고, 요약에 적절하도록 조절합니다.

```

49 v if __name__ == '__main__':
50     input_file = 'test.wav'
51     output_file = 'test_mono.wav'
52
53     # 오디오 파일을 모노로 변환
54     convert_to_mono(input_file, output_file)
55
56     # 한국어로 변환 및 텍스트 파일로 저장
57     transcribe_speech(output_file, language_code='ko-KR')
58
59
60     # 문장 추출
61 v class SentenceTokenizer(object):
62 v     def __init__(self):
63         self.kkma = Kkma()
64         self.okt = Okt()
65 v         self.stopwords = ["아", "휴", "아이구", "아이쿠", "아이고", "어", "나", "우리", "저희", "따라", "의해",
66             , "을", "를", "에", "의", "가", '중인', '만큼', '마찬가지']
67
68
69 v     def text2sentences(self, text):
70         sentences = self.kkma.sentences(text)
71         for idx in range(0, len(sentences)):
72             if len(sentences[idx]) <= 10:
73                 sentences[idx-1] += (' ' + sentences[idx])
74                 sentences[idx] = ''
75         return sentences
76
77 v     def get_nouns(self, sentences):
78         nouns = []
79         for sentence in sentences:
80             if sentence != '':
81                 nouns.append(' '.join([noun for noun in self.okt.nouns(str(sentence))
82                     if noun not in self.stopwords and len(noun) > 1]))
83         return nouns
84
85

```

```

# TF-IDF 모델 생성 및 그래프 생성
class GraphMatrix(object):
    def __init__(self):
        self.tfidf = TfidfVectorizer()
        self.cnt_vec = CountVectorizer()
        self.graph_sentence = []

    def build_sent_graph(self, sentence):
        tfidf_mat = self.tfidf.fit_transform(sentence).toarray()
        self.graph_sentence = np.dot(tfidf_mat, tfidf_mat.T)
        return self.graph_sentence

    def build_words_graph(self, sentence):
        cnt_vec_mat = normalize(self.cnt_vec.fit_transform(sentence).toarray().astype(float), axis=0)
        vocab = self.cnt_vec.vocabulary_
        return np.dot(cnt_vec_mat.T, cnt_vec_mat), {vocab[word] : word for word in vocab}

#TextRank 알고리즘 적용
class Rank(object):
    def get_ranks(self, graph, d=0.85):
        A = graph
        matrix_size = A.shape[0]
        for id in range(matrix_size):
            A[id, id] = 0
            link_sum = np.sum(A[:,id])
            if link_sum != 0:
                A[:, id] /= link_sum
                A[:, id] *= -d
                A[id, id] = 1
        B = (1-d) * np.ones((matrix_size, 1))
        ranks = np.linalg.solve(A, B)
        return {idx: r[0] for idx, r in enumerate(ranks)}

```

```

121 #TextRank Class 구현
122 class TextRank(object):
123     def __init__(self, text):
124         self.sent_tokenize = SentenceTokenizer()
125
126
127         self.sentences = self.sent_tokenize.text2sentences(text)
128
129         self.nouns = self.sent_tokenize.get_nouns(self.sentences)
130
131         self.graph_matrix = GraphMatrix()
132         self.sent_graph = self.graph_matrix.build_sent_graph(self.nouns)
133         self.words_graph, self.idx2word = self.graph_matrix.build_words_graph(self.nouns)
134
135         self.rank = Rank()
136         self.sent_rank_idx = self.rank.get_ranks(self.sent_graph)
137         self.sorted_sent_rank_idx = sorted(self.sent_rank_idx, key=lambda k: self.sent_rank_idx[k], reverse=True)
138
139         self.word_rank_idx = self.rank.get_ranks(self.words_graph)
140         self.sorted_word_rank_idx = sorted(self.word_rank_idx, key=lambda k: self.word_rank_idx[k], reverse=True)
141
142     def summarize(self, sent_num=3):
143         summary = []
144         index=[]
145         for idx in self.sorted_sent_rank_idx[:sent_num]:
146             index.append(idx)
147         index.sort()
148
149         for idx in index:
150             summary.append(self.sentences[idx])
151         return summary
152
153     def keywords(self, word_num=10):
154         rank = Rank()
155         rank_idx = rank.get_ranks(self.words_graph)
156         sorted_rank_idx = sorted(rank_idx, key=lambda k: rank_idx[k], reverse=True)
157
158         keywords = []
159         index=[]
160
161         for idx in sorted_rank_idx[:word_num]:
162             index.append(idx)
163
164         #index.sort()
165         for idx in index:
166             keywords.append(self.idx2word[idx])
167
168         return keywords

```

STT 를 통해 텍스트 파일로 저장했다면 저번 영상에 보여드렸던
부분의 코드들을 통해 요약을 진행합니다.

```

169
170 v def read_file(file_path):
171 v     with open(file_path, 'r', encoding='utf-8') as file:
172 v         return file.read()
173
174 v if __name__ == '__main__':
175     # 기본 파일 이름 설정
176     base_filename = 'transcript'
177
178     idx = 1
179     while True:
180         file_path = f"{base_filename}_{idx}.txt"
181         if not os.path.exists(file_path):
182             break
183         text_content = read_file(file_path)
184
185         # 텍스트의 길이가 300자 이상인 경우에만 TextRank를 사용하여 요약 및 키워드 추출
186         if len(text_content) >= 300:
187             try:
188                 textrank = TextRank(text_content)
189                 # 요약 출력
190                 print(f"Summary of {file_path}:")
191                 for row in textrank.summarize(3):
192                     print(row)
193                 # 키워드 출력
194                 print(f"Keywords of {file_path}:")
195                 print(textrank.keywords())
196             except ValueError as e:
197                 print(e)
198
199         idx += 1

```

만약 텍스트파일의 길이가 300 자 이하 라면 요약하지 않습니다.
 현재 stt 부분은 스킵하고 직접 "transcript_(번호).txt"를 생성하여
 실행해본 결과 정상적인 실행이 확인 되었습니다.

이상이 STT 와 결합할 요약 알고리즘 코드입니다.

3. 추후 일정

- 앞으로 남은 기간 동안의 일정입니다.

[14 주차]

- 물품배송 오면 조립을 진행할 계획입니다.
- 평가단을 구성하여 요약 내용 적절성에 대한 평가를 진행할 계획입니다.

[15 주차]

- 캡스톤디자인 최종 발표