『외국어교육』 제26집, 2023. 12. (51-70) ISSN 2233-6095

Python을 이용한 AI Hub 교육용 한국인의 영어 음성 데이터 활용

정 현 성
(한국교원대 교수)

한국교원대학교
외국어교육연구소

2023. 12.

Python을 이용한 AI Hub 교육용 한국인의 영어 음성 데이터 활용*

정 현 성 (한국교원대 교수)

I. 들어가는 말

2022년과 2023년에 걸쳐 AI 기술 및 제품서비스 개발에 필요한 AI 인프라를 지원하기 위해서 AI 통합 플랫폼인 AI Hub가 구축되었다(과학기술정보통신부/한국지능정보사회진흥원 2023). AI 데이터 영역에서는 지능정보산업 인프라 조성사업으로 추진한 14개 분야의 AI 학습용 데이터와 국내외 기관/기업에서 보유한 AI 학습용 데이터를 공개하였다. 2023년 12월 현재 교육용으로 제작한 한국인 발화 외국어 음성 데이터는 '영어, 중국어, 일본어, 스페인어, 프랑스어, 독일어, 러시아어' 등 총 7개 언어의 데이터가 구축되어 있다. 그 중 '교육용 한국인의 영어 음성 데이터'는 1,000명이 총 1,000시간을 녹음한 데이터이다. 중 · 고등학생이 50% 참여하고, 성인이 50% 참여하였다. '교육용 한국인의 중국어 · 일본어 음성 데이터'는 각 언어별로 500명이 녹음하였고, 언어별로 500시간이 녹음되었다. '교육용 한국인의 외국어(영 · 중 · 일 제외) 음성 데이터'는 언어당 각 250명이 녹음에 참여하였고, 언어별로 각 250시간의 녹음이 진행되었다. 세 데이터 모두 40%는 발음 평가 데이터, 60%는 말하기 평가 데이터로 구성되어 있다(한승희 2023). 본 연구에서는 '교육용 한국인의 영어 음성 데이터'를 중심으로 Python을 이용하여 데이터의 정보를 추출하고 활용하는 방법에 관하여 살펴보고자 한다.

II. 데이터 검색 및 다운로드

AI Hub 홈페이지에서 회원 가입을 한 후 로그인을 하면 'AI 데이터 찾기'나 '찾으시는 데이터를 입력해 주세요'라는 돋보기 검색 창이 나타난다. 이 검색 창에서 자신이 원하는 언어를 입력하면 다양한 데이터 목록을 볼 수 있다. '영어'를 입력하면 다운로드가 가능한 데이터 목록을 볼 수 있다. 그 중 '교육용 한국인의 영어 음성 데이터'를 다운로드 받는 방법을 살펴

* 이 논문에 사용된 Python 코드는 2023년 7월 8일에 진행된 한국음성학회 산하 실험음성학연구회 하계워크숍에서 성신여자대학교 윤태진 교수님이 제공해 주신 것을 응용하여 사용하였습니다. 일부 코드는 ChatGPT(https://chat.openai.com/)의 도움을 받아 작성하였습니다.

보자.

해당 데이터의 다운로드 단추를 누르면 팝업 창이 뜨면서 '037.교육용 한국인의 영어 음성 데이터' 앞에 '+' 표시가 제시된다. '+' 표시를 누르면 세부적인 데이터의 폴더 구조를 확인할 수 있다. '01-1.정식개발데이터'를 누르면 'Validation' 폴더와 'Training' 폴더가 나오는데, 'Training' 폴더의 데이터는 실제로 분석을 하거나 예측 모델링을 수행하기 위한 훈련 데이터이고, 'Validation' 폴더의 데이터는 훈련 데이터를 통해 도출된 분석과 예측 모델링이 얼마나 적합한지 모델링 등을 통해 예측된 수치(평가 점수 등)와 실제 수치를 비교해 검증하기 위한 검증 데이터이다. 'Validation' 폴더와 'Training' 폴더를 누르면 둘 다 동일한 '01.원천데이터'와 '02.라벨링데이터' 하위 폴더를 가지고 있다. '원천데이터' 폴더에는 실제 발화된 음성 wav 파일이 zip 압축 파일 형태로 저장되어 있고, '라벨링데이터'에는 개별 음성 파일에 대한 녹음 정보, 발화자 정보, 녹음 환경 정보 등의 메타 정보와 발음 및 말하기 평가 라벨링 정보 등이 zip 압축 파일 형태로 저장되어 있다. 개별 '라벨링데이터' 파일은 JSON JavaScript Object Notation 형식으로 되어 있기 때문에 그대로 활용할 수 없고, Python을 활용하여 정보를 추출하여 일반인이 활용 가능한 CSV 형식으로 변환해야 하기 때문에 비교적 데이터에 대한 진입 장벽이 높다고 할 수 있다.

'Validation' 폴더 하위 폴더인 '라벨링데이터' 폴더의 압축 파일 중 'VL_SPK_en_ED_IS.zip' 파일에서 'VL'은 'Validation' 데이터라는 의미이고, 'SPK'는 '말하기 평가 데이터'라는 의미이고, 'en'은 영어 데이터라는 의미이다. 같은 폴더의 가장 마지막 파일명은 'VL_PRN_en_NA_NA.zip'인데 'PRN'은 '발음 평가 데이터'라는 의미이다. '원천데이터' 의 각 파일명은 'VL' 대신 'VS' 시작하는 것만 다르고 '라벨링데이터'의 파일명과 서로 상응하기 때문에 '라벨링데이터' 파일의 음성 파일을 확인하기 위해서는 '원천데이터' 폴더의 'VS'로 시작하는 동일한 파일을 확인하며 된다.

'Training' 폴더 하위의 '라벨링데이터'의 파일명 'TL_SPK_en_ED_IS.zip'과 '원천데이터'의 파일명 'TS_SPK_en_ED_IS.zip'에서 'TL'은 훈련데이터의 라벨링이라는 의미이고, 'TS'는 훈련데이터의 음성이라는 의미이다. '라벨링데이터'는 개별 파일의 크기가 상대적으로 크지 않아 동일한 폴더의 파일을 한꺼번에 다운로드 받으면 되지만, '원천데이터'의 파일 크기는 작게는 4KB에서 크게는 29GB에 달하기 때문에 한꺼번에 다운로드 받지 말고, 필요한 압축 파일을 하나씩 다운로드 받는 것이 좋다.

본 연구에서는 발음 평가 데이터를 다운로드 받는 과정을 더 자세히 살펴보기로 하자. 발음 평가 데이터 중 훈련 데이터인 'Training' 폴더 하위의 '원천데이터' 압축 파일인 'TS_PRN_en_NA_NA.zip'을 다운로드 받은 후 압축을 풀면 동일한 이름의 폴더 안에 총 91,594개의 음성 파일이 들어 있다. 이에 상응하는 '라벨링데이터'의 압축 파일인 'TL_PRN_en_NA_NA.zip'을 다운로드 받은 후 압축을 풀면 동일한 이름의 폴더 안에 동일한 수의 라벨링 파일이 JSON 형식으로 들어 있다. 검증 데이터인 'Validation' 폴더 하위의 '원천데이터'와 '라벨링데이터'도 동일한 방식으로 다운로드 받은 후 압축을 풀면 된다. 검증 데이

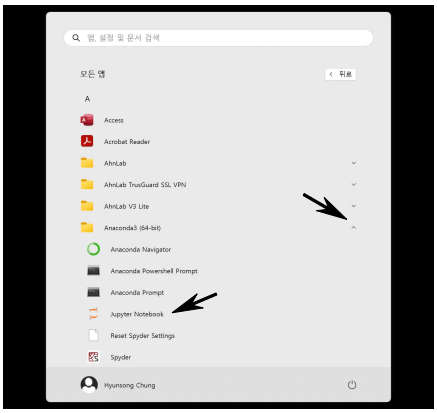
터는 각각 훈련 데이터의 12.5% 정도에 해당하는 11,450개의 파일이 들어있다.

III. Python을 실행하고 JSON 파일의 정보 추출하기

이렇게 다운로드 받은 음성 파일은 특별히 다른 형태로 변환할 필요성이 없지만, '라벨링데이터'의 JSON 형식의 파일은 통계 분석 등을 위하여 데이터프레임 dataframe으로 변환한 후 CSV 형식으로 저장할 필요가 있다. 이것을 위해서 Python 프로그램을 활용하는 것이 가장 적절한데, 이 장에서는 우선 Python 프로그램을 어떻게 구동할 수 있을지 살펴본 후, Python을 이용해 JSON 파일의 정보를 추출하는 방법에 관해 살펴보기로 하자.

1. Anaconda 설치 후 Jupyter Notebook의 Python 실행하기

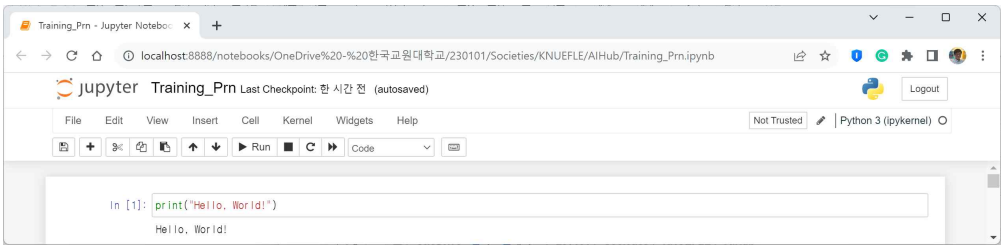
일반 PC에서 손쉽게 Python을 실행하기 위해서는 'https://www.anaconda.com/download'에서 자신의 컴퓨터 사양과 시스템에 맞게 'Anaconda'라는 플랫폼을 설치하는 것이 좋다. 'Anaconda'를 설치하면, PC의 경우 '작업 표시줄'의 '시작'을 누른 후 오른쪽 상단의 '모든 앱'을 누르면 'A'로 시작하는 목록에서 'Anaconda3 (64-bit)'를 찾을 수 있다. [그림 1]과 같이 바로 오른쪽 아래 화살표를 누르면 목록에서 'Jupyter Notebook'을 찾을 수 있는데, 그것을 누르면 'Jupyter Notebook'이라는 Python 실행 도구를 구동할 수 있다.



[그림 1] Jupyter Notebook 설치 장소

'Jupyter Notebook'을 실행하면 컴퓨터에 따라 10초에서 20초의 시간이 흐른 후 본인의 인터넷 브라우저에 jupyter 폴더 페이지가 나타난다. 폴더 페이지에서 자신이 작업을 할 디렉토리로 이동하거나 새로운 폴더를 만들어 작업을 실행할 수 있다. 본 논문에서는 AIHub라는

폴더에서 작업을 하는 예시를 보여주고자 한다. AIHub 폴더로 가서 [그림 2]와 같이 오른쪽 상단의 'New' 단추에서 'Python 3(ipykernel)'를 누르면 Python 스크립트를 입력하고 실행할 수 있는 창이 나타난다. 먼저 입력 창에 [그림 2]와 같은 명령어를 입력한 후 상단의 '▶Run'을 눌러보면, 'Hello, World!'가 출력되는 것을 확인할 수 있다. 입력하는 파일의 제목을 정하고, 저장하기 위해서 첫 번째 명령어를 실행한 이후, 상단의 'Untitled'를 자신이 알아보기 쉬운 이름으로 바꾸어 저장해 보자. 이 논문에서는 'Training_Prn'으로 저장하였다.

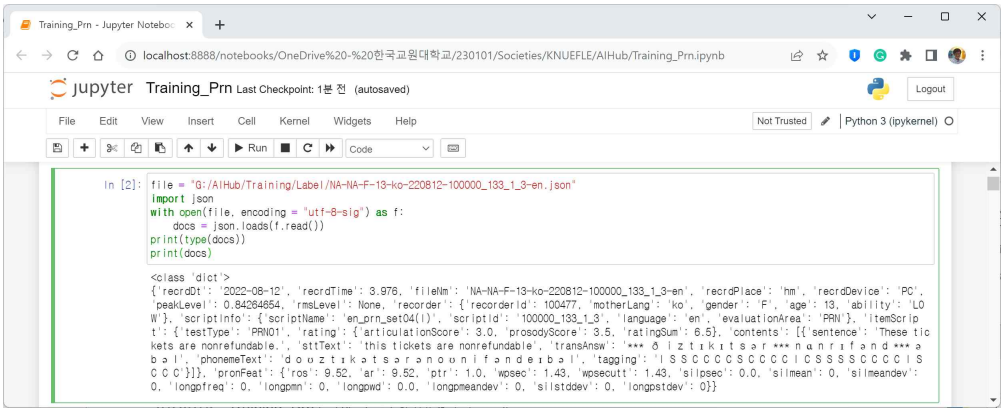


[그림 2] Jupyter Notebook에서 'Hello, World!' 출력하기

Jupyter Notebook은 이처럼 입력 창에 Python 명령어를 입력한 후 '▶Run'을 누르면 그 명령이 실행되어 입력 창 바로 아래에 결과물을 보여주거나, 필요할 경우 별도의 파일로 저장할 수 있는 기능이 있다. 위의 과정이 순조롭게 진행되었다면, 본격적으로 AIHub의 데이터를 어떻게 다루는지 다음 절에서 살펴보자.

2. JSON 파일 불러들여 메타 정보 저장하기

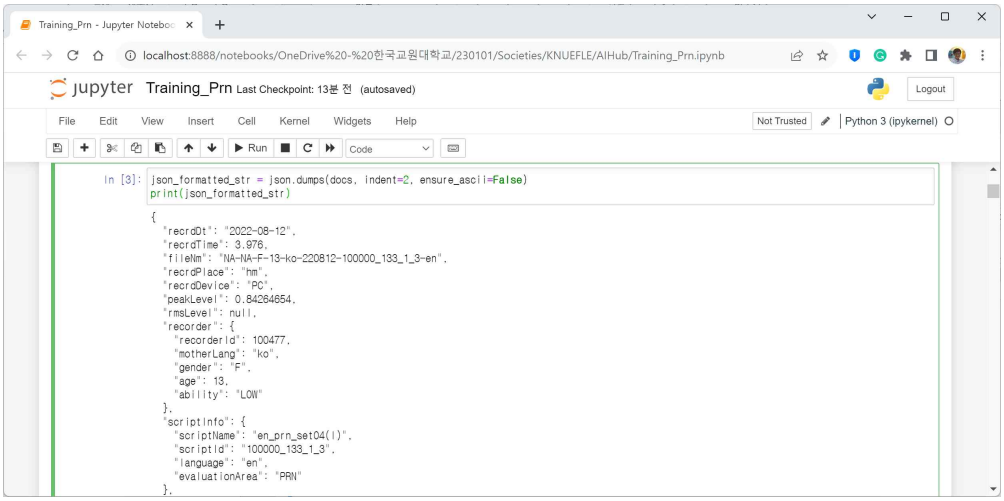
본 연구에서는 훈련데이터의 원천데이터와 라벨링데이터를 모두 G 드라이브의 AIHub 폴더 하위에 있다고 가정하고 작업을 진행하였다. 우선 입력 창에서 [그림 3]과 같이 스크립트를 입력한 후 실행해 보자.



[그림 3] 파일 경로 지정 후 JSON 파일 데이터 출력하기

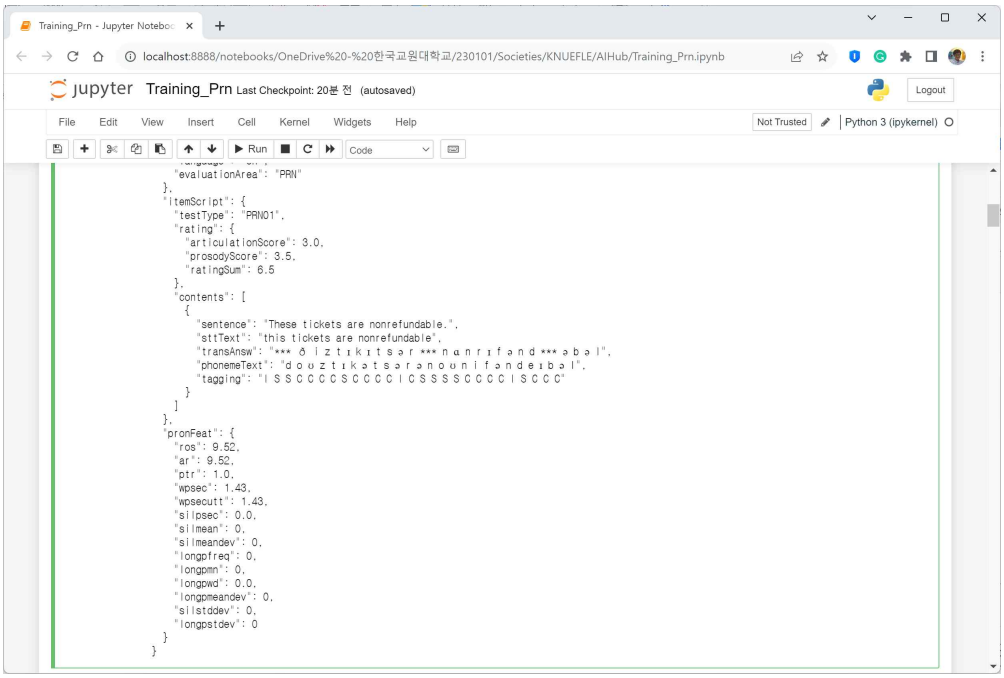
우선 'file = "G:/AIHub/Training/Label/NA-NA-F-13-ko-220812-100000_133_1_3-en.json"'은 읽어 드릴 파일의 파일 경로를 지정하는 부분이다. 해당 경로에 있는 JSON 파일의 경로를 'file' 변수에 할당한다. 'import json'은 JSON 형식의 데이터를 다루기 위해 Python 내장 모듈인 'json'을 불러오는 명령어이다. 'with open(file, encoding="utf-8-sig") as f:'는 지정된 파일을 열고 파일 내용을 읽어오는 부분이다. 'open' 함수를 사용하여 파일을 열고 'encoding="utf8"'을 통해 UTF-8 인코딩을 사용하여 파일을 읽는다. UTF-8 인코딩을 사용하는 것은 파일 안에 한글이 포함되어 있을 수 있기 때문이다. 'with'를 사용하면 파일을 열고 나서 자동으로 파일을 닫아 준다. 'docs = json.loads(f.read( ))'는 파일 내용을 문자열로 읽어온 후, 'json.loads( )' 함수를 사용하여 JSON 형식의 문자열을 Python 데이터 타입으로 변환한다. 변환된 데이터는 'docs' 변수에 할당된다. 여기서 'loads'는 'load strring'의 약어로, JSON 형식의 문자열을 로드하여 Python 데이터 타입으로 변환하는 역할을 한다. 'print(type(docs))'는 변환된 데이터의 데이터 타입을 출력하는데, 이 부분은 주로 디버깅이나 데이터 타입 확인을 위해 사용된다. 'print(docs)'는 변환된 데이터를 출력한다. 이 부분에서는 변환된 데이터가 Python에서 사용할 수 있는 형태로 출력된다. 이 스크립트를 실행하면 해당 JSON 파일의 데이터가 스크립트 하단의 <class 'dict'> 내용과 같이 출력된다.

다음은 Python에서 처리한 데이터를 [그림 4]와 같이 다시 JSON 형식의 문자열로 변환하고, 그 문자열을 보기 좋게 정렬해 보자. 이 과정은 데이터를 다룰 때 가독성을 높이고 디버깅 debugging을 용이하게 하는데 도움이 된다.



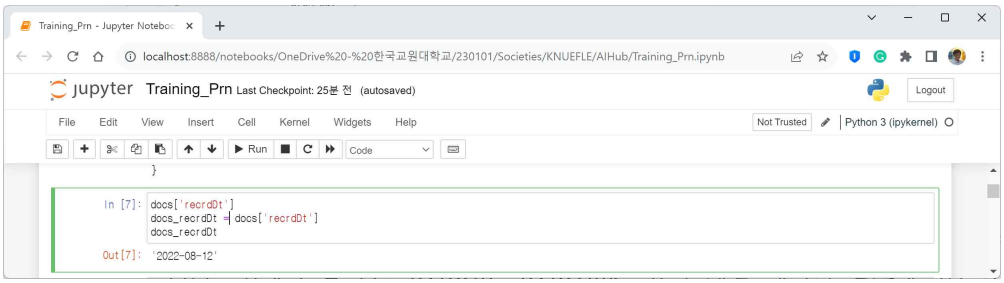
[그림 4] JSON 형식의 문자열 변환

우선 'json_formatted_str = json.dumps(docs, indent=2, ensure_ascii=False)'은 'json.dumps( )' 함수를 사용하여 Python 데이터를 JSON 형식의 문자열로 변환한다. 이때 'docs'는 이미 이전에 JSON으로 변환된 데이터이다. 'indent=2'는 들여쓰기를 2칸으로 지정하여 출력 문자열을 읽기 쉽게 만들어준다. 'ensure_ascii=False'는 ASCII 이외 문자를 그대로 출력하도록 하는 옵션이다. 'print(json_formatted_str)'는 변환된 JSON 형식의 문자열을 출력한다. 이 부분에서는 들여쓰기와 함께 보기 좋게 정리된 형태로 JSON 문자열이 출력된다. 가려진 나머지 문자열은 [그림 5]와 같다.



[그림 5] JSON 파일 문자열 후반부

다음은 위에서 출력된 'recrdDt,' 'recrdTime' 등의 개별 데이터 필드에 접근하여 그 값을 변수에 할당하는 과정을 [그림 6]에서 살펴보자.



[그림 6] 개별 필드를 변수에 할당하기

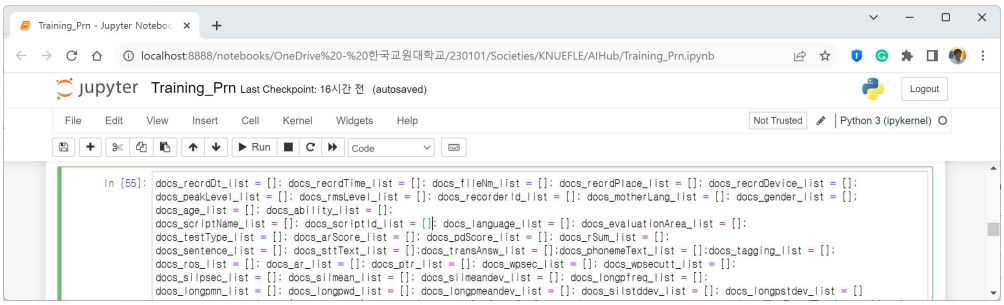
'docs['recrdDt']는 'docs'라는 변수가 가리키는 JSON 데이터에서 'recrdDt'라는 필드에 해당하는 값을 가져온다. 이 부분은 JSON 데이터의 특정 필드에 접근하는 역할을 한다. 'docs_recrdDt = docs['recrdDt']'는 가져온 'recrdDt' 필드의 값을 'docs_recrdDt' 변수에 할당한다. 이를 통해 나중에 코드에서 해당 값으로 더 쉽게 사용할 수 있게 된다. 'docs_recrdDt'는 앞에서 가져온 'recrdDt' 필드의 값을 출력한다. 변수에 할당된 값은 이후에 필요한 작업에

사용할 수 있다. 다른 개별 필드에 대해서도 필드의 이름만 바꾸어 동일한 작업을 실행하면 된다. 이 논문에서 활용할 발음 평가 데이터에 사용된 39개 개별 필드와 그 설명은 <표 1>과 같다(과학기술정보통신부/한국지능정보사회진흥원 2023).

<표 1> 발음 평가 데이터에 포함된 필드명

필드명	설명	필드명	설명	필드명	설명
recrdDt	녹음 일자	scriptId	스크립트 아이디	ar	철자 기반 조음 속도
recrdTime	음성 녹음 길이	language	스크립트 언어	ptr	음소-시간 비
fileNm	파일명	evaluationArea	평가 유형	wpsec	단어기반 조음 속도
recrdPlace	녹음 장소	testType	문제	wpsecutt	단어 기반 발화 속도
recrdDevice	녹음 기기	articulationScore	발음의 정확성	silpsec	초당 평균 휴지 개수
peakLevel	파형의 가장 높은 값	prosodyScore	운율의 유창성	silmean	평균 휴지 길이
rmsLevel	전체 레벨의 평균값	ratingSum	합산 점수	silmeandev	휴지 길이 평균 편차
recorderId	녹음 사용자 식별 아이디	sentence	스크립트 문장	longfreq	긴 휴지 빈도
motherLang	모국어	sttText	STT 결과물	longpmn	긴 휴지 평균 길이
gender	성별	transAnsw	문장 단위 표준 발음	longpwd	단어 간 평균 긴 휴지 개수
age	연령	phonemeText	문장 단위 음소 인식 결과	longpmeandev	긴 휴지 평균 편차
ability	언어 실력	tagging	문장 단위 오류 유형 라벨링	silstddev	휴지 길이 표준 편차
scriptName	스크립트 파일명	ros	철자 기반 발화 속도	longstdev	긴 휴지 표준 편차

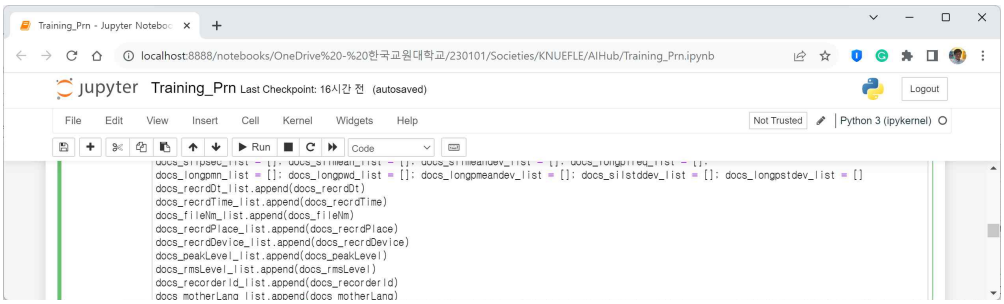
다음으로 [그림 7]과 같이 서로 다른 데이터 필드에 대한 정보를 저장할 빈 리스트를 미리 생성하는 과정이 필요하다. 이러한 리스트는 이후에 데이터를 추출하여 저장하기 위한 용도로 사용된다.



[그림 7] 빈 리스트 생성하기

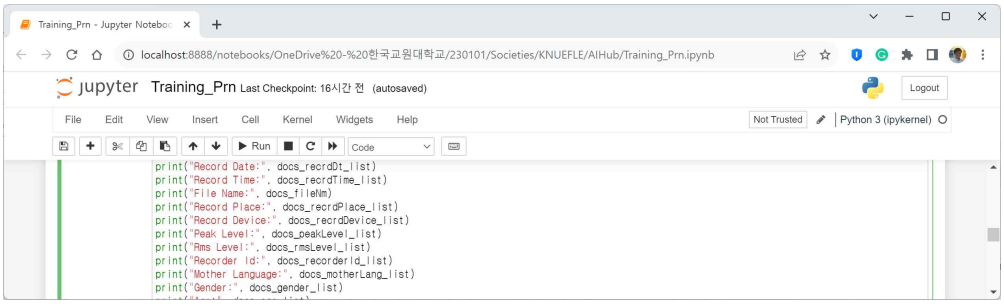
예를 들어 'docs_recrdDt_list = [ ]'는 'docs_recrdDt_list'라는 이름의 빈 리스트를 생성하여 추후 'recrdDt' 필드에 대한 데이터를 저장하게 된다. <표 1>에 있는 39개 필드 전체에 대해서 동일한 스크립트를 생성하면 된다.

다음은 [그림 8]과 같이 이렇게 생성된 빈 리스트에 데이터를 추가하는 작업을 한다.



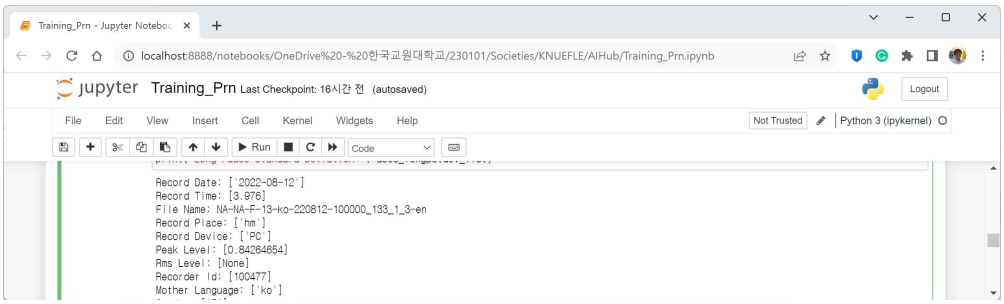
[그림 8] 빈 리스트에 데이터 추가하기

'docs_recrdDt_list.append(docs_recrdDt)'는 이전 단계에서 생성된 'docs_recrdDt_list'라는 빈 리스트에, 그 이전에 언급한 'doc_recrdDt' 변수에 저장된 'recrdDt' 필드의 데이터, 즉 '2022-8-22'를 추가하는 과정이다. 39개 전체 필드에 관해서 동일한 방법으로 코드를 작성한다. 데이터가 제대로 추가되었는지 확인하려면 [그림 9]와 같이 'print' 기능을 활용하여 추가된 데이터를 출력해 볼 수 있다.



[그림 9] 데이터 출력하기

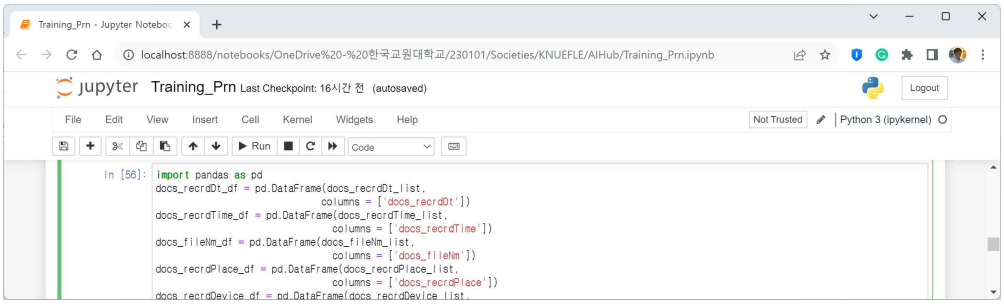
'print("Record Date:", docs_recrdDt_list)'는 'Record Date'라는 문구와 함께, 이전 단계의 'docs_recrdDt_list' 리스트에 추가한 'recrdDt' 필드의 데이터를 출력한다. 역시 39개 필드 전체에 대해서 동일한 코드를 작성하면 된다. 위 세 단계의 스크립트를 실행하면 [그림 10]과 같은 출력 화면이 나타난다.



[그림 10] 출력된 데이터

이러한 작업을 통해 특정 데이터 필드에 속한 값들을 화면에 표시하여 사용자가 데이터를 확인할 수 있다. 이렇게 출력함으로써 데이터의 정확성을 확인하거나 분석 결과를 확인하는 등의 용도로 활용할 수 있다.

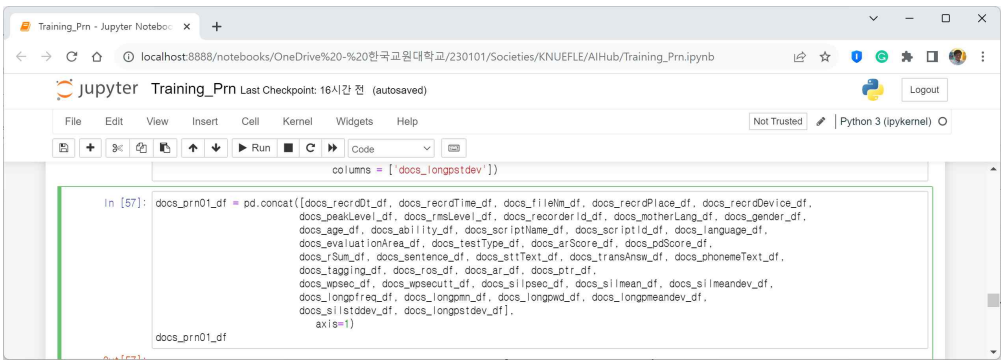
다음은 이러한 값을 통계 분석 등에 활용하기 위해서 데이터프레임으로 변환하는 작업이 필요하다. [그림 11]은 'pandas' 라이브러리를 사용하여 데이터프레임을 생성하는 Python 스크립트이다.



[그림 11] pandas 라이브러리를 사용해 데이터프레임 생성하기

데이터프레임은 표 형태의 데이터 구조로, 행과 열이 있는 데이터를 효과적으로 다루기 위한 자료 구조이다. 'import pandas as pd'는 'pandas' 라이브러리를 'pd'라는 별칭으로 불러들인다. 이렇게 하면 'pd'를 사용하여 'pandas' 라이브러리의 기능을 호출할 수 있다. 'docs_recrdDt_df = pd.DataFrame(docs_recrdDt_list, columns = ['docs_recrdDt'])'는 'pd.DataFrame' 함수를 사용하여 'docs_recrdDt_list'에 입력한 리스트를 기반으로 데이터프레임을 생성한다. 'columns' 옵션을 사용하여 데이터프레임의 열 이름을 'docs_recrdDt'로 설정한다. 이 코드는 'recrdDt' 데이터를 담고 있는 'docs_recrdDt_list' 리스트를 데이터프레임으로 변환하는 과정을 나타낸다. 39개 전체 필드에 대해서 필드 이름만 교체해 동일한 코드를 반복 입력하면 된다.

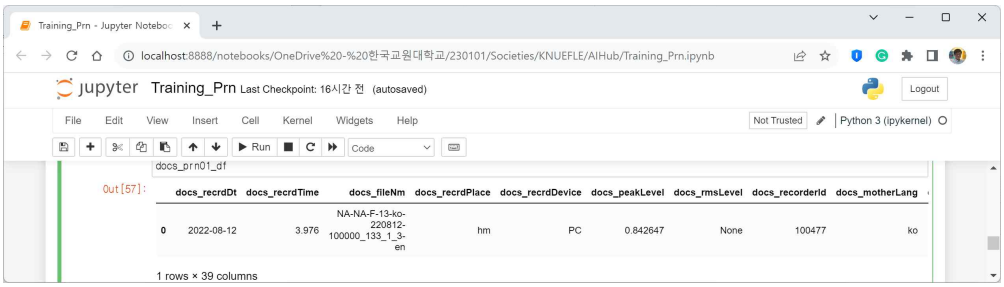
다음은 이렇게 생성한 여러 개의 데이터프레임을 하나로 합치는 concatenation 작업을 수행해야 한다.



[그림 12] 데이터프레임 하나로 합치기

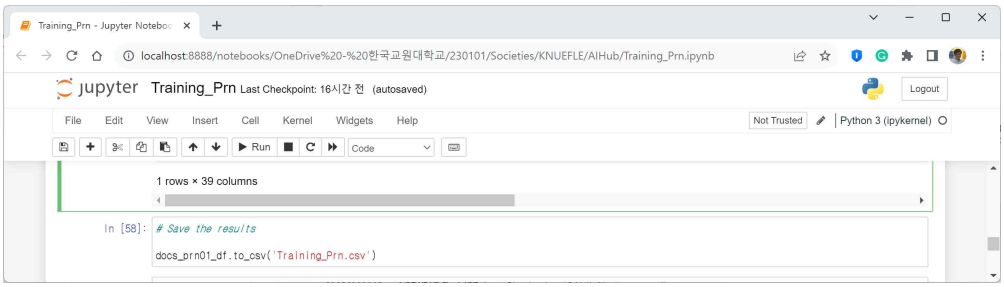
[그림 12]에서 'pd.concat([...], axis=1)'은 'pd.concat( )' 함수를 사용하여 여러 개의 데이터프레임을 가로로 합치는 기능을 한다. 여기서 'axis=1'은 열 방향으로 합치라는 의미이다. 'concat( )' 함수의 첫 번째 매개 변수에는 합칠 데이터프레임들의 39개 리스트를

‘[docs_recrdDt_df, docs_recrdTime_df, docs_fileNm_df, ..., docs_longpstdev_df]’ 형태로 나열한다. 합쳐진 결과는 ‘doc_prn01_df’라는 변수에 할당하고, 이 변수에는 모든 데이터프레임이 가로로 합쳐져서 저장된다. 변수의 이름은 자신이 가장 알아보기 쉬운 명칭으로 정하면 된다. 이렇게 합쳐진 데이터프레임은 이후에 다양한 분석이나 시각화 작업에 활용될 수 있다. ‘doc_prn01_df’ 코드를 실행하면 이렇게 합쳐진 데이터프레임을 [그림 13]과 같이 확인할 수 있다.



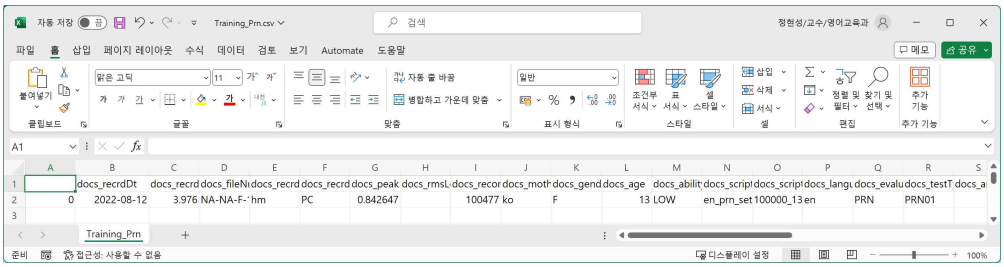
[그림 13] 데이터프레임 확인하기

‘NA-NA-F-13-ko-220812-100000_133_1-3-en.json’ 파일에 대해 위에 제시한 모든 과정을 거쳐 39개 필드의 개별 값을 하나의 열에 모두 합쳐서 ‘doc_prn01_df’라는 데이터프레임으로 저장한 것이다. 이렇게 저장된 데이터프레임은 [그림 14]와 같이 CSV 파일로 저장할 수 있다.



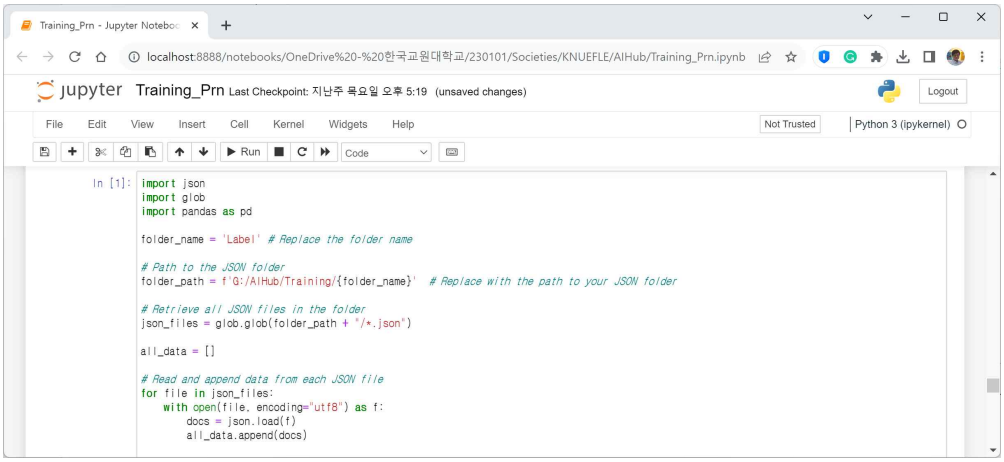
[그림 14] CSV 파일로 저장하기

‘docs_prn01_df.to_csv(‘Training_Prn.csv’)’는 위에서 저장한 ‘docs_prn01_df’라는 데이터프레임을 ‘Training_Prn.csv’라는 CSV 파일로 저장하라는 스크립트이다. 이 코드를 실행하면 Python 스크립트가 실행되고 있는 동일한 폴더에 ‘Training_Prn.csv’라는 파일이 생성되어 있는 것을 확인할 수 있고, [그림 15]와 같이 엑셀 등을 활용해 파일의 내용을 확인할 수 있다. 생성 파일의 이름도 자신이 가장 알아보기 쉬운 것으로 정해주면 된다.



[그림 15] 저장된 CSV 파일 엑셀에서 보기

지금까지 하나의 JSON 파일을 어떻게 다루고 데이터프레임으로 변환해 CSV 파일로 저장할 수 있는가에 대해서 살펴보았다. 본 연구에서 다루는 발음 평가 데이터의 훈련데이터만 하더라도 이러한 JSON 파일이 91,594개 있는데, 이 파일을 위와 같이 하나씩 처리한다는 것은 비현실적이고 비경제적이다. 그래서 모든 파일을 한꺼번에 처리할 수 있는 대안이 필요하다. 지금부터 91,594개의 파일을 Python 스크립트를 활용해 어떻게 한꺼번에 처리할 수 있을지 살펴보자.

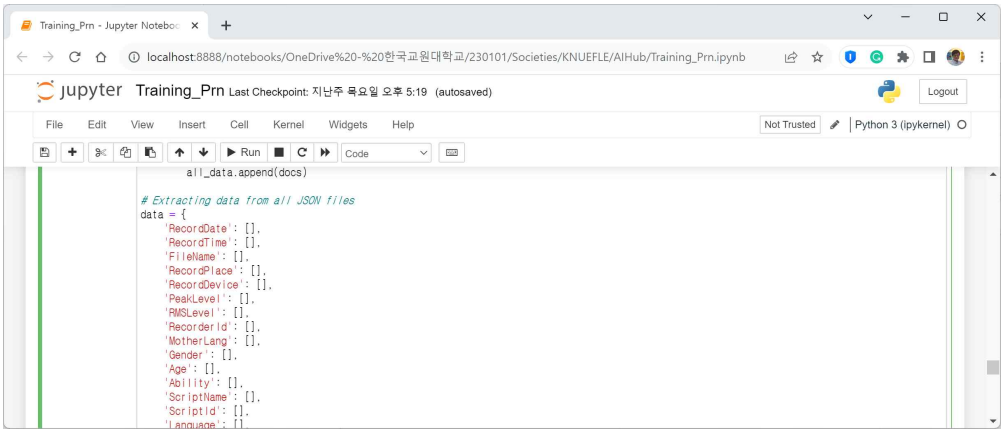


[그림 16] 라이브러리 불러와 경로 정하고 폴더의 모든 JSON 파일 한꺼번에 읽어 들이기

[그림 16]에서는 우선 특정 폴더에 있는 여러 개의 JSON 파일을 읽어와서 그 내용을 하나의 리스트에 저장하는 작업을 위해 위와 같은 코드를 작성하였다. 본 논문을 위해서 '라벨링 데이터'는 G:/AIHub/Training/Label 폴더에 저장해 두었다. 'import json'은 JSON 형식의 데이터를 다루기 위해 Python 내장 모듈인 'json'을 불러온다. 'import glob'은 파일 경로를 패턴 매칭하여 파일들의 목록을 가져오기 위해 'glob' 모듈을 불러온다. 'import pandas as pd'는 데이터프레임을 생성하고 다루기 위해 'pandas' 라이브러리를 불러온다. 'folder_name =

'Label''은 JSON 파일들이 있는 폴더의 이름을 'folder_name' 변수에 할당한다. 'folder_path = f'G:/AIHub/Training/{folder_name}''는 JSON 파일들이 있는 폴더의 전체 경로를 'folder_path' 변수에 할당한다. 여기서 'f-string'을 사용하여 폴더 이름을 포함한 전체 경로를 만든다. 'json_files = glob.glob(folder_path + "/*.json")'는 지정된 폴더에서 확장자가 '.json'인 모든 파일을 목록을 가져와서 'json_files' 변수에 할당한다. 'all_data = [ ]'는 JSON 파일들의 내용을 저장할 빈 리스트인 'all_data'를 생성한다. 'for file in json_files:'는 'json_files' 리스트에 있는 각 JSON 파일에 대해 그 다음 이어지는 명령을 반복한다. 'with open(file, encoding="utf8") as f:'는 각 JSON 파일을 UTF-8 인코딩을 사용해 읽기 위해 열고, 파일 내용을 'f' 변수에 저장한다. 'docs = json.load(f)'는 'json.load( )' 함수를 사용하여 JSON 파일의 내용을 Python 데이터로 변환하고, 그 결과를 'docs' 변수에 할당한다. 'all_data.append(docs)'는 변환된 데이터를 'all_data' 리스트에 추가한다. 이러한 코딩은 그 과정이 하나의 JSON을 읽어 들여 처리할 때와 동일하지만, 'Label'이라는 폴더 안에 있는 모든 JSON 파일을 처리할 수 있다는 것에서 차이가 있다. 이러한 과정을 통해 모든 JSON 파일의 내용이 'all_data' 리스트에 저장되며, 이후 이 리스트를 사용하여 데이터프레임을 생성할 수 있다.

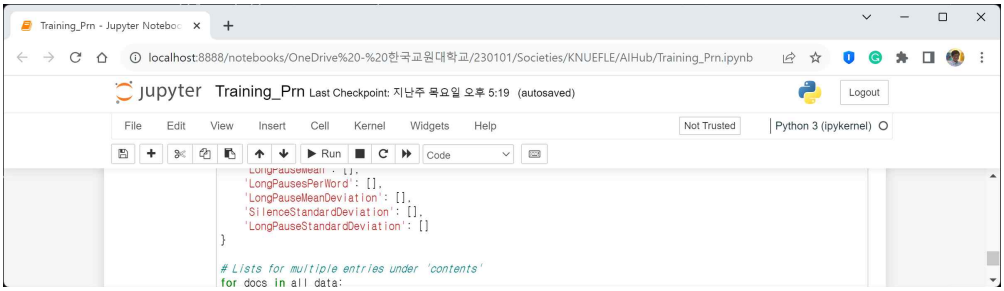
다음으로 Python에서 [그림 17]과 같이 사용할 데이터를 저장하기 위한 빈 리스트를 생성하는 작업을 해야한다.



[그림 17] 빈 리스트 생성하기

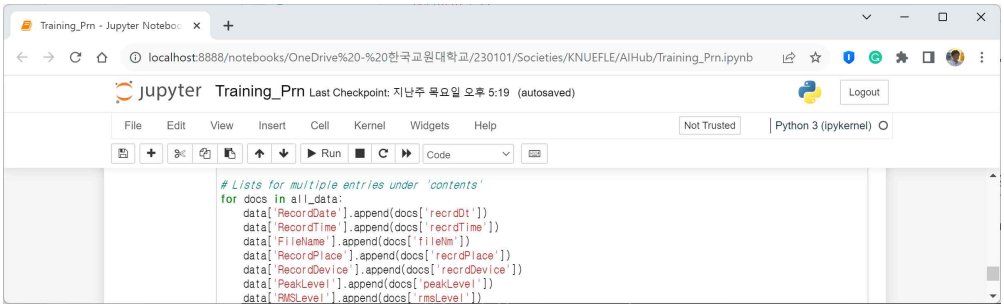
'data = {'는 'data'라는 변수에 새로운 딕셔너리를 생성한다. ''RecordData': [ ],'는 'RecordDate'라는 키에 빈 리스트가 할당된다. 이 리스트는 추후 'RecordDate'에 해당하는 데이터를 저장할 용도로 사용된다. 39개의 필드에 대해 동일한 작업을 수행하고, 개별 키 이름은 자신이 가장 알아보기 쉬운 것으로 정하면 된다. 마지막 필드인 'LongPauseStandardDeviatioin' 키에 빈 리스트를 할당한 이후에는 [그림 18]과 같이 ',' 없이

'}'를 사용해 코드를 종료한다.



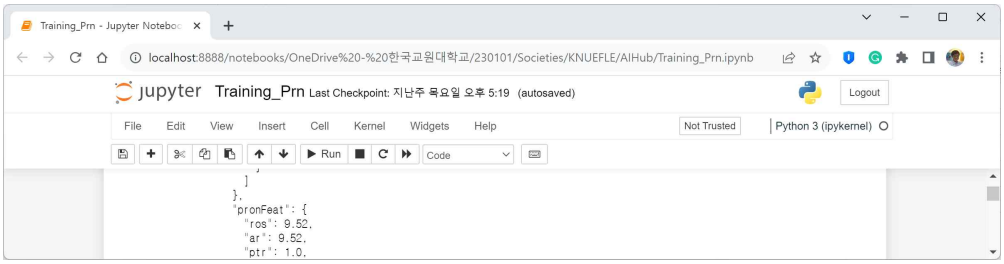
[그림 18] 빈 리스트 생성하기 마지막 열

다음으로 [그림 16]에서 생성된 'all_data'에 있는 각각의 JSON 데이터에서 개별 필드에 속한 데이터를 추출하여 미리 정의된 데이터에 해당 값들을 추가하는 작업을 [그림 19]와 같이 수행한다.



[그림 19] 데이터 추출 후 해당 값 추가하기

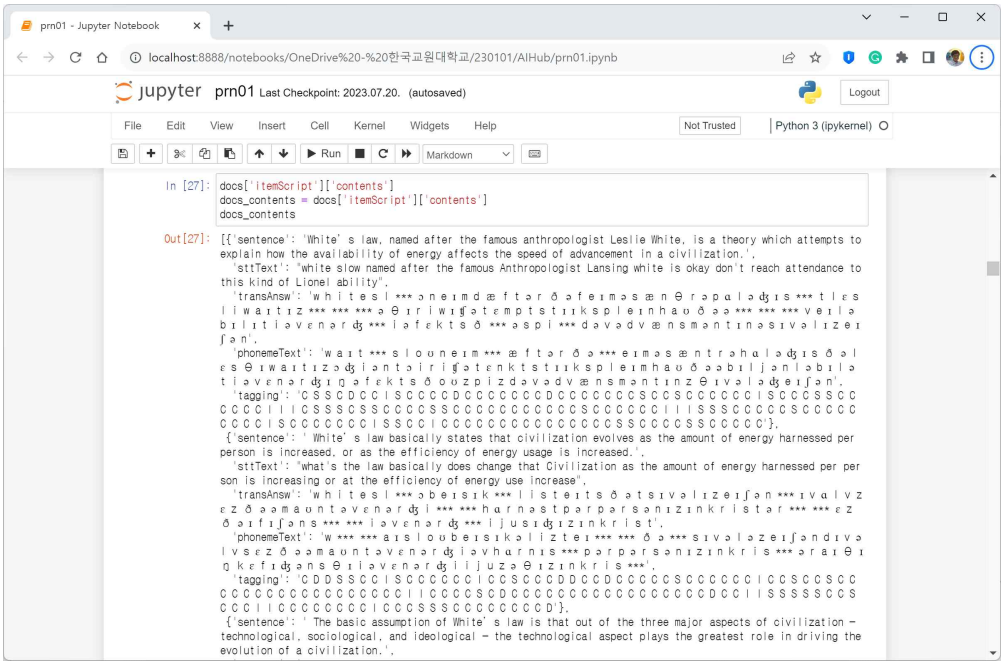
'for docs in all_data:'는 'all_data' 리스트에 잇는 각 JSON 데이터에 대해 반복하라는 의미이다. 'data['RecordDate'].append(docs['recrdDt'])'는 현재 반복되고 있는 JSON 데이터에서 첫 번째 필드인 'recrdDt' 필드의 값을 추출하여 'data' 딕셔너리의 'RecordDate' 키에 해당하는 리스트에 해당 값을 추가한다. 이것 또한 39개 필드에 대해 동일한 코드를 입력하면 된다. 이때 주의할 점은 JSON 데이터에서 필드가 계층적으로 이루어져 있을 경우 상위 필드와 하위 필드를 모두 입력해 주어야 한다는 것이다.



[그림 20] 계층적 데이터 필드 예시

[그림 20]에 나오는 JSON 데이터를 보면 'ros' 필드는 'pronFeat' 하위에 있다. 이러한 경우에는 상위 필드와 하위 필드를 'data['RateofSpeech'].append(docs['pronFeat']['ros'])'와 같이 입력하여 데이터에 추가하면 된다.

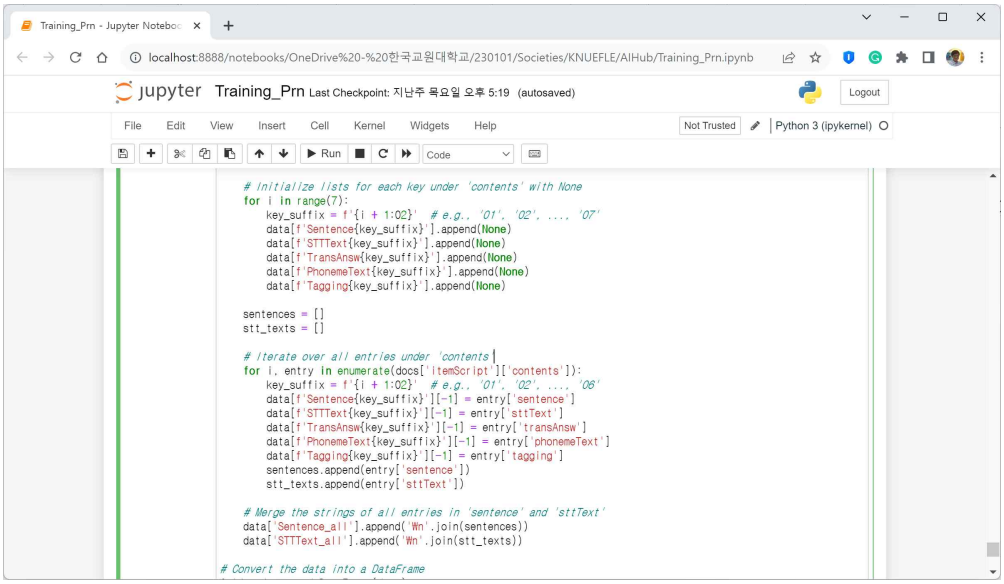
이렇게 JSON 데이터를 처리하다 보면 몇 가지 문제에 직면하게 된다. 'contents' 필드의 개별 하위 필드에 두 개 이상의 데이터가 존재하는 경우가 있다.



[그림 21] 복수 데이터 출력 예시

[그림 21]을 보면 'contents' 하위의 'sentence, sttText, transAnsw, phonemeText, tagging'에 두 개 이상의 데이터가 존재하는 것을 확인할 수 있다. 이런 경우에는 각각을 읽어 들인 후

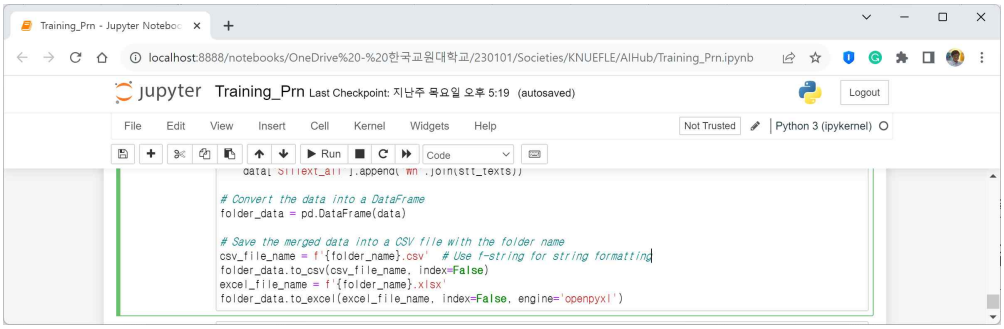
모두 합치는 방법으로 데이터를 처리하였다. 즉, JSON 데이터에서 'itemScript' 필드 하위의 'contents' 에 있는 각 항목을 순회하면서 필요한 정보를 추출하고, 미리 정의된 데이터 구조에 해당 정보를 추가하는 작업을 수행하였다.



[그림 22] 복수 데이터 처리하기

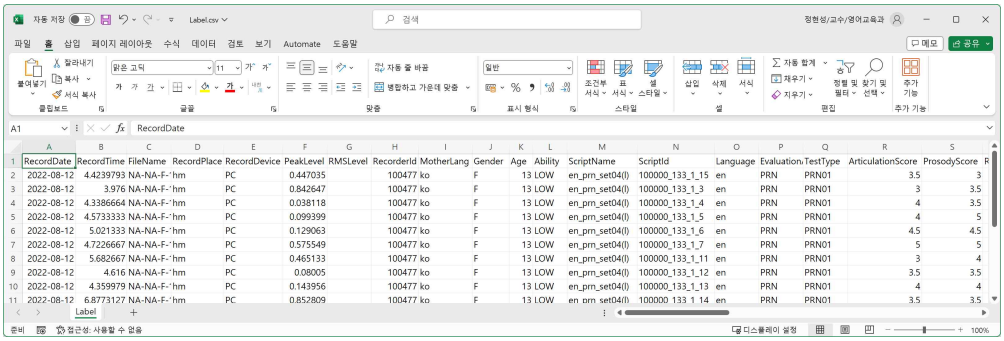
우선 'for i in range(7):'는 0부터 6까지 총 7번 반복하는 루프 loop를 시작한다. 각 JSON 데이터를 검토한 결과 개별 항목이 최대 6번 반복되는 것을 확인하였고, 이에 따라 한 번 더 여유를 두어 개별 항목을 7번 반복하도록 설정하였다. 'key_suffix = f'{i + 1:02}'는 반복 중인 인덱스에 1을 더한 후, 이를 두 자리 숫자로 표현하는 문자열을 생성한다. 예를 들어, 첫 번째 반복에서는 '01', 두 번째에서는 '02' 등과 같이 이름이 붙는다. 'data[f'Sentence{key_suffix}'].append(None)' 등은 빈 리스트에 None 값을 추가하는 과정으로, 'Sentence'와 현재 반복 중인 인덱스를 조합한 키에 해당하는 리스트를 초기화한다. 'STText, TransAnsw, PhonemeText, Tagging' 모두 이름만 바꾸고 동일한 코드를 입력한다. 이렇게 되면 'Sentence01, ..., Sentence07; STTText01, ..., STTTex07' 등과 같이 우선 빈 리스트를 초기화하고, 이후에 주어진 데이터에서 필요한 정보를 추출하여 해당 리스트에 추가한다. 마지막으로, 'Sentence_all, STTText_all'에는 모든 'contents'에 속한 'sentence' 및 'sttText'를 합친 문자열이 추가된다. 'transAnsw, phonememText, tagging'에 대해서도 동일하게 문자열을 합칠 수 있지만 이 논문에서는 생략하였다.

마지막으로 [그림 23]과 같이 'data' 딕셔너리에 저장된 데이터를 데이터프레임으로 변환하고, 그 결과를 CSV와 Excel 파일로 저장하는 작업을 수행할 수 있다.



[그림 23] 데이터프레임을 CSV와 Excel 파일로 저장하기

'folder_data = pd.DataFrame(data)'는 'data' 딕셔너리를 이용하여 데이터프레임인 'folder_data'를 생성한다. 이는 데이터를 효과적으로 다루기 위한 표 형태의 구조를 가진 자료 구조이다. 'csv_file_name = f'{folder_name}.csv''는 이미 앞에서 정한 폴더 이름 'Label'을 이용하여 CSV 파일의 이름을 생성하고, 'f-string'을 사용하여 폴더 이름과 파일 확장자인 '.csv'를 결합한다. 'folder_data.to_csv(csv_file_name, index=False)'는 데이터프레임을 CSV 파일로 저장한다. 'index=False'는 행 인덱스를 파일에 저장하지 않도록 하는 옵션이다. 'excel_file_name = f'{folder_name}.xlsx''는 Excel 파일의 이름을 생성하는 코드이고, 'folder_data.to_excel(excel_file_name, index=False, engine='openpyxl')'는 데이터프레임을 Excel 파일로 저장한다. 'engine='openpyxl''은 Excel 파일을 생성할 때 사용할 엔진을 지정하는 옵션이다. 이렇게 함으로써, 원본 데이터를 데이터프레임으로 변환하고, CSV 파일과 Excel 파일로 저장하는 과정이 완료된다. 저장된 CSV 파일을 열어보면 [그림 24]와 같이 데이터를 확인할 수 있다.



[그림 24] 저장된 CSV 파일

IV. 결론

이 논문에서는 AIHub에서 제공하는 음성 데이터를 검색하고 다운로드 하는 방법에 대해서 소개하고, 그 중 '교육용 한국인의 영어 음성 데이터'에 있는 JSON 파일의 데이터를 Python을 이용해 추출하고 저장하는 방법을 단계별로 소개하였다. '교육용 한국인의 영어 음성 데이터'는 '훈련 데이터'와 '검정 데이터'로 구성되어 있고, 각 데이터는 다시 소리 파일인 '원천 데이터'와 메타 정보 및 분석 정보가 담겨있는 '라벨링데이터'로 구성되어 있다. '라벨링데이터'의 파일은 JSON 파일로 만들어져 있기 때문에, Python을 이용해 JSON 파일을 데이터프레임으로 변환해 CSV 파일로 저장해야한다. 일반 PC에서는 Anaconda를 설치한 후 Jupyter Notebook을 실행하면 Python 프로그램을 구동해 JSON 파일을 조작할 수 있다. 이 논문에서는 발음 평가 데이터에 대해 하나의 JSON 파일을 불러들여 메타 정보를 데이터프레임으로 변환한 후 저장하는 방법과, 하나의 폴더에 있는 모든 JSON 파일을 반복적으로 불러들여 한꺼번에 저장하는 방법에 대해 단계적으로 소개하였다. Python을 활용하여 JSON 파일을 처리하면 몇 주간 소요되는 자료 처리 작업을 반자동화하여 몇 분 만에 처리할 수 있다는 큰 장점이 있다. 이렇게 저장된 데이터를 활용해 R이나 SPSS와 같은 통계 프로그램으로 연구자의 목적에 맞는 연구를 진행할 수 있을 것이다.

참고문헌

과학기술정보통신부/한국지능정보사회진흥원(2023): 교육용 한국인의 영어 음성 데이터. 사이트 주소: https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=&topMenu=&aihubDataSe=data&dataSetSn=71463

한승희(2023): 한국인 발화 다국어 AI Hub 데이터셋 설계 및 구축: 다국어 통번역 낭독체 데이터와 언어교육용 한국인 발화 외국어 음성 데이터, 실린 곳: 한국음성학회 2023 봄 학술대회 논문집(16-27), 서울: 한국음성학회.

Abstract

Manipulation of English Speech Data of Koreans for Education in AI Hub Using Python

CHUNG, Hyunsong (Prof. Korea National Univ. of Educ.)

In this paper, we present the construction of speech data in AI Hub and demonstrate the manipulation of English speech data from Koreans for educational purposes using Python. The 'English speech data of Koreans for education' comprises both training and validation data, each containing sound files in WAV format and corresponding labeling files in JSON format. Processing each JSON file is facilitated using Python within a Jupyter Notebook environment. Metadata from a collection of JSON files stored in a specified folder can be extracted and organized into a structured dataframe. The content of each JSON file within the specified folder is loaded, and relevant information is extracted and stored in a dictionary named 'data.' Subsequently, the collected data is converted into a dataframe using the pandas library. The resulting dataframe is then saved as both a CSV file and an Excel file, containing structured data extracted from the JSON files. The code presented in this study efficiently processes JSON data, structures it into a dataframe, and exports the results into CSV and Excel files for further analysis.

key words:
에이아이허브, 파이썬, 음성 데이터, 제이슨 파일
AI Hub, Python, Speech data, JSON files

논문투고일: 2023. 11. 30.
논문심사일: 2023. 12. 20.
게재확정일: 2023. 12. 24.