**코드 매뉴얼**

실행 파일 및 분석 모듈 매뉴얼

Ver.1.0

2023.10.31

**개정 이력**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **개정**  **번호** | **개정일자** | **작성자** | **개정사유** | **개정내용** | **승인일자** | **비고** |
| 1 | 2023-10-31 | KT | 최초작성 |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

목차

[1. 개요 5](#_Toc149592865)

[**1.1. 분석 환경** 5](#_Toc149592866)

[**1.1.1 OS/소프트웨어** 5](#_Toc149592867)

[**1.1.2 하드웨어** 5](#_Toc149592868)

[**1.1.3 초기 디렉토리 구조 및 기능 설명** 5](#_Toc149592869)

[2. 실행 파일 설명 6](#_Toc149592870)

[**2.1. 환경 세팅 모듈** 6](#_Toc149592871)

[**2.1.1. 개요** 6](#_Toc149592872)

[**2.1.2. 실행 방법** 6](#_Toc149592873)

[**2.1.3. 상세 스크립트** 6](#_Toc149592874)

[**2.2. 이력서 직무 분류 모델 학습** 6](#_Toc149592875)

[**2.2.1. 개요** 6](#_Toc149592876)

[**2.2.2. 실행 방법** 7](#_Toc149592877)

[**2.2.3. 실행 관련 파일 목록** 7](#_Toc149592878)

[**2.2.4. Input/Output 데이터 정보** 7](#_Toc149592879)

[**2.2.5. 실행 옵션** 7](#_Toc149592880)

[**2.2.6. 실행 결과 예시** 8](#_Toc149592881)

[**2.2.7. 상세 스크립트** 9](#_Toc149592882)

[**2.3. 이력서 직무 예측** 9](#_Toc149592883)

[**2.3.1. 개요** 9](#_Toc149592884)

[**2.3.2. 실행 방법** 9](#_Toc149592885)

[**2.3.3. 실행 관련 파일 목록** 11](#_Toc149592886)

[**2.3.4. Input/Output 데이터 정보** 11](#_Toc149592887)

[**2.3.5. 실행 옵션** 11](#_Toc149592888)

[**2.3.6. 실행 결과 예시** 11](#_Toc149592889)

[**2.3.7. 상세 스크립트** 12](#_Toc149592890)

[**2.4. 예측 직무별 키워드 추출** 12](#_Toc149592891)

[**2.4.1. 개요** 12](#_Toc149592892)

[**2.4.2. 실행 방법** 12](#_Toc149592893)

[**2.4.3. 실행 관련 파일 목록** 14](#_Toc149592894)

[**2.4.4. Input/Output 데이터 정보** 14](#_Toc149592895)

[**2.4.5. 실행 옵션** 14](#_Toc149592896)

[**2.4.6. 실행 결과 예시** 14](#_Toc149592897)

[**2.4.7. 상세 스크립트** 14](#_Toc149592898)

[3. 기타 기능 모듈 파일 설명 15](#_Toc149592899)

[**3.1. utils/Logging.py** 15](#_Toc149592900)

[**3.2. utils/Timer.py** 16](#_Toc149592901)

[**3.3. utils/Read\_File.py** 17](#_Toc149592902)

[**3.4. utils/Preprocessing.py** 20](#_Toc149592903)

[**3.5. utils/Modeling.py** 22](#_Toc149592904)

# **1. 개요**

본 매뉴얼 파일은 하기 기능들을 수행하기 위한 매뉴얼 문서임

* 분석 환경 세팅
* 이력서 파일 내 텍스트 추출
* 이력서 내 텍스트에 대한 데이터 전처리 수행
* 이력서에 대한 직무 분류 모델 구축
* 구축된 직무 분류 모델을 활용한 이력서별 예측 결과 저장
* 예측된 직무별 이력서 내 주요 키워드 추출

## **1.1. 분석 환경**

### **1.1.1 OS/소프트웨어**

* OS: Ubuntu 20.04.6 LTS
* 소프트웨어: Python 3.7.10

### **1.1.2 하드웨어**

* CPU: Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
* RAM: 377GB

### **1.1.3 초기 디렉토리 구조 및 기능 설명**

|  |  |  |  |
| --- | --- | --- | --- |
| **디렉토리 구조도** | **디렉토리별 기능 설명** | | |
| **No** | **이름** | **설명** |
|  | 01 | code | 분석을 위한 코드파일이 담긴 폴더 |
| 02 | data | 분석 데이터 저장 폴더 |
| 03 | log | 분석 수행 시 로그 파일 저장 폴더 |
| 04 | model | 이력서 직무 분류 모델 저장 폴더 |
| 05 | output | 분석 결과 저장 폴더 |

# **2. 실행 파일 설명**

[이력서 직무 분류 모델 학습], [이력서 직무 예측], [직무별 키워드 추출] 작업 실행 프로세스 설명

## **2.1. 환경 세팅 모듈**

### **2.1.1. 개요**

1) 실행 시기

* Python 라이브러리 설치를 위한 환경 세팅 시 최초 1회 실행

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | os, pathlib, warnings, traceback, subprocess | - |
| 2 | 기타 기능 모듈 | utils.Logging | Logger (info, error) |

### **2.1.2. 실행 방법**

|  |
| --- |
| **실행 Command Line (Bash)** |
| > python install.py |
| **실행 Command 예시 (Bash)** |
| > python install.py |

### **2.1.3. 상세 스크립트**

* Code/install.py 파일 참고

## **2.2. 이력서 직무 분류 모델 학습**

### **2.2.1. 개요**

1) 실행 시기

* 학습될 이력서 파일이 일정 수준 적재되었거나, 업데이트 주기를 두고 모델을 업데이트 하는 형식으로 실행 권장

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | os,, traceback, warnings, argparse, glob, numpy, pandas, json, pickle, time, datetime, re, gluonnlp, scikit-learn, pytorch, random, torchvision, kobert, transformers | - |
| 2 | 기타 기능 모듈 | utils.Logging | Logger (info, error) |
| 3 | utils.Timer | Timer (start, stop) |
| 4 | utils.Proprecess | clean, noun\_extractor |
| 5 | utils.Modeling | BERTDataset, BERTClassifier, calc\_accuracy |
| 6 | utils.Read\_File | text\_to\_text\_file, pdf\_to\_text\_by\_pdfplumber, docx\_to\_text\_by\_docx2txt, doc\_to\_text\_by\_aspose |

### **2.2.2. 실행 방법**

|  |
| --- |
| **실행 Command Line (Bash)** |
| > python code/run.py -t {학습할이력서폴더경로[target\_path]} -d {모델학습일자[train\_date]} |
| **실행 Command 예시 (Bash)** |
| > python code/run.py -t data/01\_raw\_data -d 20231013  [실행 Command 예시를 위한 이력서 폴더 구조 예시] |

### **2.2.3. 실행 관련 파일 목록**

|  |  |  |
| --- | --- | --- |
| **No.** | **파일 위치** | **기능** |
| 1 | config.json | 이력서 직무 분류 모델 구축을 위한 매개변수 값 저장 파일 |
| **2** | **code/run.py** | **이력서 직무 분류 모델 학습 및 모델 저장 프로세스를 수행하는 Python 파일** |

### **2.2.4. Input/Output 데이터 정보**

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **경로** | **파일명** | **파일 설명** |
| Input  데이터 | data/01\_raw\_data/ | {직무 대분류}/{직무 중분류}/{이력서 파일} (ex. 1. 경영STAFF/1. 인사/이력서파일명.pdf) | 모델 학습에 활용될 이력서 파일[[1]](#footnote-1) |
| Output  데이터 | model/ | resume\_classification\_model\_*{train\_date}*[[2]](#footnote-2).pth | 학습된 이력서 직무 분류 모델 파일 |
| label\_encoder\_*{train\_date}*.pkl | 학습에 사용된 이력서 직무 구분값 인코더 파일 |

### **2.2.5. 실행 옵션**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **옵션** | **플래그** | **입력 인자** | **필수여부** | **비고** |
| 1 | 모델학습대상이력서폴더경로 | -t (--target\_path) | *{target\_date}* (ex. data/01\_raw\_data) | Y | 이력서 직무 분류 모델을 학습할 이력서 파일이 담긴 폴더명으로, 폴더 내 구성은 [{직무 대분류}/{직무 중분류}/{이력서 파일}] 형식으로 구성되어야함 |
| 2 | 모델학습일자 | -d (--train\_date) | *{train\_date}* (ex. 20231013) | Y |  |
| 3 | config파일 경로 | -c (--config\_path) | *{config\_file\_path}* (ex.code/config.json) | N (default: code/config.json) | 모델 매개변수 정보를 담은 config 파일 경로 |

### **2.2.6. 실행 결과 예시**

|  |  |  |  |
| --- | --- | --- | --- |
| **예시 실행 Command (Bash)** | | | **예시 실행 시 활용한 GPU 자원 및 소요 시간** |
| > python code/run.py -t data/01\_raw\_data -d 20231013 | | | 학습 이력서 파일 수: 22,311 개 🡪 GPU 메모리: 약 26 GB / 소요시간: 약 4시간 40분 |
| **예시 실행 후 디렉토리 결과** | | | |
|  | | | |
| **생성 디렉토리(파일)별 기능 설명** | | | |
| **No.** | **경로** | **설명** | |
| 1 | data/02\_text\_data/*{train\_date}*/ | 이력서 원본 파일(PDF, 워드, 텍스트) 내 텍스트를 추출하여 txt 파일로 변환한 폴더 | |
| 2 | data/03\_meta\_data/resume\_*{train\_date}*.csv | 학습할 이력서 파일 메타 정보(파일명, 파일경로, 직무 분류 등)을 저장한 파일 | |
| 3 | data/03\_meta\_data/total\_*{train\_date}*.csv | 이력서 파일 별로 텍스트 내용을 읽어 저장한 파일 | |
| 4 | data/04\_tsv\_data/train\_*{train\_date}*.tsv  data/04\_tsv\_data/validation\_*{train\_date}*.tsv  data/04\_tsv\_data/test\_*{train\_date}*.tsv | 이력서 직무 분류 모델 구축 시 이력서 데이터를 학습/검증/평가 데이터로 나눠 저장한 파일 | |
| 5 | model/kobert\_from\_pretrained/ model/kobert\_news\_wiki\_ko\_cased-1087f8699e.spiece kobert\_v1.zip | 이력서 직무 분류 모델 구축에 활용되는 전이 모델(Kobert)의 초기 형태 관련 파일 (이력서 데이터 학습 전 모델 관련) | |
| 6 | model/label\_encoder\_*{train\_date}*.pkl | 학습에 사용된 이력서 직무 구분값 인코더 파일 | |
| 7 | model/resume\_classifcation\_model\_*{train\_date}*.pth | 학습된 이력서 직무 분류 모델 파일 | |

### **2.2.7. 상세 스크립트**

* Code/run.py 파일 참고

## **2.3. 이력서 직무 예측**

### **2.3.1. 개요**

1) 실행 시기

* 이력서 파일에 대한 직무를 예측하고자 할 때 활용

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | os,, traceback, warnings, argparse, glob, numpy, pandas, json, pickle, time, datetime, re, gluonnlp, scikit-learn, pytorch, random, torchvision, kobert, transformers | - |
| 2 | 기타 기능 모듈 | utils.Logging | Logger (info, error) |
| 3 | utils.Timer | Timer (start, stop) |
| 4 | utils.Proprecess | clean, noun\_extractor |
| 5 | utils.Modeling | BERTDataset, BERTClassifier, calc\_accuracy |
| 6 | utils.Read\_File | text\_to\_text\_file, pdf\_to\_text\_by\_pdfplumber, docx\_to\_text\_by\_docx2txt, doc\_to\_text\_by\_aspose |

### **2.3.2. 실행 방법**

|  |
| --- |
| **실행 Command Line (Bash)** |
| > python code/predict.py -t {예측대상이력서폴더경로[target\_path]} -d {모델학습일자[train\_date]} |
| **실행 Command 예시 (Bash)** |
| > python code/predict.py -t data/01\_raw\_data -d 20231013  [실행 Command 예시를 위한 이력서 폴더 구조 예시] |

### **2.3.3. 실행 관련 파일 목록**

|  |  |  |
| --- | --- | --- |
| **No.** | **파일 위치** | **기능** |
| 1 | config.json | 이력서 직무 분류 모델 활용을 위한 매개변수 값 저장 파일 |
| **2** | **code/predict.py** | **이력서 직무 예측 수행 및 예측 결과 파일 저장 프로세스를 수행하는 Python 파일** |

### **2.3.4. Input/Output 데이터 정보**

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **경로** | **파일명** | **파일 설명** |
| Input  데이터 | *{target\_path}* | {직무 대분류}/{직무 중분류}/{이력서 파일} (ex. 1. 경영STAFF/1. 인사/이력서파일명.pdf) | 예측 대상 이력서 폴더 경로 |
| model/*{train\_date}*[[3]](#footnote-3) | resume\_classification\_model.pth | 가장 최근에 학습된 이력서 직무 분류 모델 파일 |
| label\_encoder.pkl | 가장 최근 모델 학습에 사용된 이력서 직무 구분값 인코더 파일 |
| Output  데이터 | output/*{train\_date}* | resume\_classification\_result\_*{now}*[[4]](#footnote-4).csv | 예측 대상 이력서에 대한 직무 예측 결과 파일 |

### **2.3.5. 실행 옵션**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **옵션** | **플래그** | **입력 인자** | **필수여부** | **비고** |
| 1 | 예측대상이력서폴더경로 | -t (--target\_path) | *{target\_date}* (ex. data/01\_raw\_data) | Y | 직무를 예측하 이력서 파일이 담긴 폴더명으로, 폴더 내 구성은 [{직무 대분류}/{직무 중분류}/{이력서 파일}] 형식으로 구성되어야함 |
| 2 | 모델학습일자 | -d (--train\_date) | *{train\_date}* (ex. 20231013) | Y |  |
| 3 | config파일 경로 | -c (--config\_path) | *{config\_file\_path}* (ex.code/config.json) | N (default: code/config.json) | 모델 매개변수 정보를 담은 config 파일 경로 |

### **2.3.6. 실행 결과 예시**

|  |  |  |  |
| --- | --- | --- | --- |
| **예시 실행 Command (Bash)** | | | **예시 실행 시 활용한 GPU 자원 및 소요 시간** |
| > python code/predict.py -t data/01\_raw\_data -d 20231013 | | | 예측 이력서 파일 수: 22,311 개 🡪 GPU 메모리: 약 2 GB / 소요시간: 약 15분 |
| **예시 실행 후 디렉토리 결과** | | | |
|  | | | |
| **생성 디렉토리(파일)별 기능 설명** | | | |
| **No.** | **경로** | **설명** | |
| 1 | output/resume\_classification\_result\_*{now}*.csv | 예측 대상 이력서에 대한 직무 예측 결과 | |

### **2.3.7. 상세 스크립트**

* Code/predict.py 파일 참고

## **2.4. 예측 직무별 키워드 추출**

### **2.4.1. 개요**

1) 실행 시기

* 이력서 직무 예측 결과 파일이 존재할 때, 예측된 직무별 이력서 내 작성된 키워드를 추출 시 활용

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | os, ast, pathlib, warnings, traceback, argparse, pandas, json, collections, pickle, time, datetime, keybert, transformers, kiwipiepy | - |
| 2 | 기타 기능 모듈 | utils.Logging | Logger (info, error) |
| 3 | utils.Timer | Timer (start, stop) |
| 4 | utils.Proprecess | clean, noun\_extractor |
| 5 | utils.Modeling | BERTDataset, BERTClassifier, calc\_accuracy |
| 6 | utils.Read\_File | text\_to\_text\_file, pdf\_to\_text\_by\_pdfplumber, docx\_to\_text\_by\_docx2txt, doc\_to\_text\_by\_aspose |

### **2.4.2. 실행 방법**

|  |
| --- |
| **실행 Command Line (Bash)** |
| > python code/keywords.py -d {모델학습일자(train\_date)} |
| **실행 Command 예시 (Bash)** |
| > python code/keywords.py -d 20231013  [실행 Command 예시를 위한 이력서 폴더 구조 예시] |

### **2.4.3. 실행 관련 파일 목록**

|  |  |  |
| --- | --- | --- |
| **No.** | **파일 위치** | **기능** |
| 1 | config.json | 이력서 직무 분류 모델 활용을 위한 매개변수 값 저장 파일 |
| **2** | **code/keywords.py** | **이력서 직무 예측 결과에 따라 직무별 키워드 추출 프로세스를 수행하는 Python 파일** |

### **2.4.4. Input/Output 데이터 정보**

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **경로** | **파일명** | **파일 설명** |
| Input  데이터 | output/*{train\_date}* | pred\_resume\_classification\_result\_yyyymmdd\_hhmmss.csv (ex. pred\_resume\_classification\_result\_20231030\_053524.csv) | 직무 예측 결과 파일 |
| Output  데이터 | output/*{train\_date}* | keyword\_*{train\_date}*[[5]](#footnote-5).csv | 예측 대상 이력서에 대한 직무 예측 결과 파일 |

### **2.4.5. 실행 옵션**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **옵션** | **플래그** | **입력 인자** | **필수여부** | **비고** |
| 1 | 모델학습일자 | -d (--train\_date) | *{train\_date}* (ex. 20231013) | Y |  |
| 2 | config파일 경로 | -c (--config\_path) | *{config\_file\_path}* (ex.code/config.json) | N (default: code/config.json) | 모델 매개변수 정보를 담은 config 파일 경로 |

### **2.4.6. 실행 결과 예시**

|  |  |  |  |
| --- | --- | --- | --- |
| **예시 실행 Command (Bash)** | | | **예시 실행 시 활용한 GPU 자원 및 소요 시간** |
| > python code/keywords.py -d 20231013 | | | 키워드 추출 이력서 파일 수: 22,311 개 🡪소요시간: 약 4시간 |
| **예시 실행 후 디렉토리 결과** | | | |
|  | | | |
| **생성 디렉토리(파일)별 기능 설명** | | | |
| **No.** | **경로** | **설명** | |
| 1 | output/*{train\_date}*/keyword \_*{train\_date}*.csv | 예측 직무별 주요 키워드 추출 결과 파일 | |

### **2.4.7. 상세 스크립트**

* Code/keywords.py 파일 참고

# **3. 기타 기능 모듈 파일 설명**

## **3.1. utils/Logging.py**

1) 개요

* 분석 수행 과정에 대한 Log 기록 기능 모듈

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | os, sys, logging, datetime | - |

3) 내장 기능

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **메소드 명** | **기능** | **입력** | **출력** |
| 1 | Logger() | Logger 객체 생성 | 현재 분석 프로세스 이름  로그 파일 저장 경로 | 분석 프로세스 수행 관련 로그 파일 |
| 2 | info() | info 레벨의 Log를 기록 | Log로 남길 텍스트/객체 등 | 로그 파일 및 CLI 내 print 형태로 출력 |
| 3 | error() | error 레벨의 Log를 기록 | Log로 남길 텍스트/객체 등 | 로그 파일 및 CLI 내 print 형태로 출력 |

4) 상세 스크립트 (python)

|  |
| --- |
| **필요 라이브러리 불러오기** |
| import os  import sys  import logging  from datetime import datetime |
| **Log 기록을 위한 Logger 객체 정의** |
| # Logger 정의  class Logger:  def \_\_init\_\_(self, file = sys.argv[0], proc\_name = None, log\_path = None):  self.today = datetime.today().strftime(format = '%Y%m%d')  self.file\_name = os.path.basename(file)  self.log\_path = log\_path  if not proc\_name:  self.logger = logging.getLogger(f"{self.file\_name}")  else:  self.logger = logging.getLogger(f"{proc\_name}")    if len(self.logger.handlers) == 0:  formatter = logging.Formatter(u'%(asctime)s [%(levelname)s] %(message)s')  # StreamHandler  stream\_handler = logging.StreamHandler()  stream\_handler.setFormatter(formatter)    # FileHandler  file\_handler = logging.FileHandler(self.log\_path)  file\_handler.setFormatter(formatter)  self.logger.addHandler(stream\_handler)  self.logger.addHandler(file\_handler)  self.logger.setLevel(logging.INFO)  def info(self, value):  self.logger.info("%s (at %s)" % (str(value), self.file\_name))  def error(self, value):  self.logger.error("%s (at %s)" % (str(value), self.file\_name)) |

## **3.2. utils/Timer.py**

1) 개요

* 분석 수행 과정에 대한 소요 시간 측정 기능 모듈

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | time | - |

3) 내장 기능

|  |  |  |
| --- | --- | --- |
| **No.** | **메소드명** | **기능** |
| 1 | start() | 지정된 Timer 측정 시작 |
| 2 | stop() | 지정된 Timer 측정 종료 및 소요 시간 Log 출력 - 예시: |

4) 상세 스크립트 (python)

|  |
| --- |
| **필요 라이브러리 불러오기** |
| import time |
| **분석 프로세스 시간 측정을 위한 Timer 객체 정의** |
| class TimerError(Exception):  """A custom exception used to report errors in use of Timer class"""  class Timer:  def \_\_init\_\_(self, logger = None):  self.\_start\_time = None  if logger:  self.logger = logger  def start(self):  """Start a new timer"""  if self.\_start\_time is not None:  raise TimerError(f"Timer is running. Use .stop() to stop it")  self.\_start\_time = time.perf\_counter()  def stop(self):  """Stop the timer, and report the elapsed time"""  if self.\_start\_time is None:  raise TimerError(f"Timer is not running. Use .start() to start it")  elapsed\_time = time.strftime("%H:%M:%S", time.gmtime(round(time.perf\_counter() - self.\_start\_time, 4)))    self.\_start\_time = None  if self.logger:  self.logger.info(f"Elapsed time: {elapsed\_time} seconds")  else:  print(f"Elapsed time: {elapsed\_time} seconds") |

## **3.3. utils/Read\_File.py**

1) 개요

* 이력서 원본 파일(docx, doc, pdf, txt)의 텍스트를 읽기 위한 함수를 정의한 기능 모듈

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | re, pdfplumber, doc2txt, aspose | - |

3) 내장 기능

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **함수명** | **기능** | **입력** | **출력** |
| 1 | text\_to\_txt\_file() | 이력서 파일에서 추출한 텍스트 저장 함수 | 추출한 텍스트 전문 추출한 텍스트를 저장할 경로 | - |
| 2 | pdf\_to\_text\_by\_pdfplumber() | PDF 이력서 파일에서의 텍스트 추출 및 저장 함수 | 추출할 이력서 파일 경로 추출한 텍스트를 저장할 경로 | 이력서 파일에서 추출한 텍스트 전문 |
| 3 | docx\_to\_text\_by\_docx2txt() | Word 이력서 파일에서의 텍스트 추출 및 저장 함수 | 추출할 이력서 파일 경로 추출한 텍스트를 저장할 경로 | 이력서 파일에서 추출한 텍스트 전문 |
| 4 | doc\_to\_text\_by\_aspose() | Word (doc) 이력서 파일에서의 텍스트 추출 및 저장 함수 | 추출할 이력서 파일 경로 추출한 텍스트를 저장할 경로 | 이력서 파일에서 추출한 텍스트 전문 |

4) 상세 스크립트 (python)

|  |
| --- |
| **필요 라이브러리 불러오기** |
| import pdfplumber  import docx2txt  import aspose.words as aw  import re |
| **원본 이력서 파일로부터의 텍스트 추출 및 저장 함수 정의** |
| def text\_to\_txt\_file(text, tgt\_path):  """    이력서 파일에서 추출한 텍스트 저장 함수  parameter  ----------  text(str): 추출한 텍스트 전문  tgt\_path(str): 추출한 텍스트를 저장할 경로      """    with open(tgt\_path, 'w') as f:  f.write(text)    def pdf\_to\_text\_by\_pdfplumber(src\_path, tgt\_path): # src\_path: 이력서 파일 경로, tgt\_path: 이력서 텍스트 로드 파일 저장 경로 (.txt)    """    PDF 이력서 파일에서의 텍스트 추출 및 저장 함수  parameter  ----------  src\_path(str): 추출할 이력서 파일 경로  tgt\_path(str): 추출한 텍스트를 저장할 경로      return  ----------  output(str): 이력서 파일에서 추출한 텍스트 전문    """    output\_ = []  with pdfplumber.open(src\_path) as pdf:  pages = pdf.pages  # print('총 페이지 수: {0}'.format(len(pages)))  for page in pages:  text = page.extract\_text()  output\_.append(text)  output = '\n\n'.join(output\_)    if tgt\_path is None:  return output  else:  text\_to\_txt\_file(output, tgt\_path)  return output    def docx\_to\_text\_by\_docx2txt(src\_path, tgt\_path): # src\_path: 이력서 파일 경로, tgt\_path: 이력서 텍스트 로드 파일 저장 경로 (.txt)    """    Word 이력서 파일에서의 텍스트 추출 및 저장 함수  parameter  ----------  src\_path(str): 추출할 이력서 파일 경로  tgt\_path(str): 추출한 텍스트를 저장할 경로      return  ----------  output(str): 이력서 파일에서 추출한 텍스트 전문    """    output = docx2txt.process(src\_path)    if tgt\_path is None:  return output  else:  text\_to\_txt\_file(output, tgt\_path)  return output  def doc\_to\_text\_by\_aspose(src\_path, tgt\_path): # src\_path: 이력서 파일 경로, tgt\_path: 이력서 텍스트 로드 파일 저장 경로 (.txt)    """    Word (doc) 이력서 파일에서의 텍스트 추출 및 저장 함수  parameter  ----------  src\_path(str): 추출할 이력서 파일 경로  tgt\_path(str): 추출한 텍스트를 저장할 경로      return  ----------  output(str): 이력서 파일에서 추출한 텍스트 전문    """    # DOC 파일 로드  text = aw.Document(src\_path)  # DOC를 TXT로 저장  text.save(tgt\_path)  # 저장된 TXT 파일 정제  with open(tgt\_path, 'r') as f:  output = ''.join(f.readlines()).strip()  output = re.sub('Evaluation Only. Created with Aspose.Words. Copyright 2003-2023 Aspose Pty Ltd.', '', output)  output = re.sub('Created with an evaluation copy of Aspose.Words. To discover the full versions of our APIs please visit: https://products.aspose.com/words/', '', output)  output = re.sub(r'\ufeff', '', output)    if tgt\_path is None:  return output  else:  text\_to\_txt\_file(output, tgt\_path)  return output |

## **3.4. utils/Preprocessing.py**

1) 개요

* 텍스트 데이터 전처리 함수를 정의한 기능 모듈

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | re, kiwipiepy | - |

3) 내장 기능

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **메소드 명** | **기능** | **입력** | **출력** |
| 1 | clean() | 불용어(한글, 영어, 숫자 제외) 제거 함수 | 불용어를 제거할 텍스트 | 불용어 제거 처리한 텍스트 |
| 2 | noun\_extractor() | 한글 명사 및 외래어 키워드 추출 함수 | 추출할 텍스트 전문 | 텍스트 내 명사/외래어 리스트 |
| 3 | gain\_keywords() | 이력서 내 키워드 추출 함수 | 추출할 텍스트 전문 | 이력서 내 키워드 리스트 |

4) 상세 스크립트 (python)

|  |
| --- |
| **필요 라이브러리 불러오기** |
| import re  from kiwipiepy import Kiwi  from kiwipiepy.utils import Stopwords  kiwi = Kiwi() |
| **불용어(한글, 영어, 숫자 제외) 제거 함수 정의** |
| def clean(text: str) -> str:    """    불용어(한글, 영어, 숫자 제외) 제거 함수  parameter  ----------  text(str): 불용어를 제거할 텍스트      return  ----------  clean\_text(str): 불용어 제거 처리한 텍스트    """  text = re.sub('[\n\t]', ' ', text) # 줄바꿈, 탭 문자 제거  text = re.sub(r'\u2028', ' ', text) # 특수 유니코드 제거  text = re.sub(r'\u2029', ' ', text) # 특수 유니코드 제거  text = re.sub(r'(?:\x01|\x02|\x03|\x04|\x05|\x06|\x07|\x08|\t|\x0b|\x0c|\r|\x0e|\x0f|\x10|\x11|\x12|\x13|\x14|\x15|\x16|\x17|\x18|\x19|\x1a|\x1b|\x1c|\x1d|\x1e)', '', text) # 특수 문자 제거  try:  clean\_text = re.sub('[^가-힣a-zA-Z0-9\s]', "", text) # 한글, 영어, 숫자 제외 삭제  except:  clean\_text = text  return clean\_text |
| **한글 명사 및 외래어 키워드 추출 함수 정의** |
| def noun\_extractor(text, stopwords):    """    한글 명사 및 외래어 키워드 추출 함수  parameter  ----------  text(str): 추출할 텍스트 전문      return  ----------  results(list): 텍스트 내 명사/외래어 리스트    """    try:  results = []  result = kiwi.analyze(text)  for token, pos, \_, \_ in result[0][0]:  if (len(token) != 1) and (pos.startswith('N') or pos.startswith('SL')) and (token.lower() not in stopwords):  results.append(token)  except:  results = []  return results |
| **이력서 내 키워드 추출 함수** |
| def gain\_keywords(text:str=None, model=None, stop\_words:list=None, top\_n:int=None) -> list:    """    이력서 내 키워드 추출 함수  parameter  ----------  text(str): 추출할 텍스트 전문  model(obj): 키워드 추출 모델 객체  stop\_words(list): 불용어 단어 리스트  top\_n: 추출할 키워드 순위 범위 (ex. top\_n = 10 -> 10순위 키워드까지 추출)      return  ----------  keywords(list): 텍스트 내 키워드 리스트    """    try:  keywords = [k for k, \_ in model.extract\_keywords(text, keyphrase\_ngram\_range=(1, 1), stop\_words=None, top\_n=top\_n)]  except:  keywords = []  return keywords |

## **3.5. utils/Modeling.py**

1) 개요

* 이력서 직무 분류 모델 관련 함수를 정의한 기능 모듈

2) 필요 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **구분** | **패키지 명** | **함수명** |
| 1 | Python 라이브러리 | random, numpy, gluonnlp, torch, torchvision, kobert, trasnfomers | - |

3) 내장 기능

|  |  |  |
| --- | --- | --- |
| **No.** | **메소드명** | **기능** |
| 1 | BERTDataset() | 이력서 직무 분류 모델 함수 활용을 위한 데이터로더 정의 |
| 2 | BERTClassifier() | 이력서 직무 분류 모델의 구조 및 기능 함수 정의 |
| 3 | calc\_accuracy() | 이력서 직무 분류 모델의 성능 평가를 위한 정확도 함수 정의 |

4) 상세 스크립트 (python)

|  |
| --- |
| **필요 라이브러리 불러오기** |
| import numpy as np  import gluonnlp as nlp  import torch  import random  import torch.backends.cudnn as cudnn  from torch import nn  import torch.nn.functional as F  import torch.optim as optim  from torch.utils.data import Dataset, DataLoader  import torchvision.models as models  from kobert.utils import get\_tokenizer  from kobert.pytorch\_kobert import get\_pytorch\_kobert\_model  from transformers import AdamW  from transformers.optimization import get\_cosine\_schedule\_with\_warmup |
| **이력서 직무 분류 모델 함수 정의** |
| # 토큰화, 패딩, 어텐션 마스크, 배치 및 데이터로더 설정  class BERTDataset(Dataset):  def \_\_init\_\_(self, dataset, sent\_idx, label\_idx, bert\_tokenizer, max\_len,  pad, pair):  transform = nlp.data.BERTSentenceTransform(  bert\_tokenizer, max\_seq\_length=max\_len, pad=pad, pair=pair)  self.sentences = [transform([i[sent\_idx]]) for i in dataset]  self.labels = [np.int32(i[label\_idx]) for i in dataset]  def \_\_getitem\_\_(self, i):  return (self.sentences[i] + (self.labels[i], ))  def \_\_len\_\_(self):  return (len(self.labels))  class BERTClassifier(nn.Module):  def \_\_init\_\_(self,  bert,  hidden\_size = 768,  num\_classes=None, ##주의: 클래스 수 바꾸어 주세요!##  dr\_rate=None,  params=None):  super(BERTClassifier, self).\_\_init\_\_()  self.bert = bert  self.dr\_rate = dr\_rate  self.classifier = nn.Linear(hidden\_size , num\_classes)  if dr\_rate:  self.dropout = nn.Dropout(p=dr\_rate)  def gen\_attention\_mask(self, token\_ids, valid\_length):  attention\_mask = torch.zeros\_like(token\_ids)  for i, v in enumerate(valid\_length):  attention\_mask[i][:v] = 1  return attention\_mask.float()  def forward(self, token\_ids, valid\_length, segment\_ids):  attention\_mask = self.gen\_attention\_mask(token\_ids, valid\_length)  \_, pooler = self.bert(input\_ids = token\_ids, token\_type\_ids = segment\_ids.long(), attention\_mask = attention\_mask.float().to(token\_ids.device))  if self.dr\_rate:  out = self.dropout(pooler)  return self.classifier(out)    def calc\_accuracy(X,Y):  max\_vals, max\_indices = torch.max(X, 1)  train\_acc = (max\_indices == Y).sum().data.cpu().numpy()/max\_indices.size()[0]  return train\_acc |

1. 이력서 파일 형식은 pdf, docx, doc, dotx, txt 형식이어야 분석 가능 [↑](#footnote-ref-1)
2. {train\_date}는 모델 학습을 진행한 날짜 형식[yyyymmdd]으로 표현됨(ex. 20231013) [↑](#footnote-ref-2)
3. {train\_date}는 직무분류모델 학습 날짜를 의미하며 날짜 형식[yyyymmdd)으로 표현됨(ex. 20231013) [↑](#footnote-ref-3)
4. {now}는 이력서 직무 예측 실행 시점을 의미하며 시간 형식(yyyymmdd\_hhmmss)으로 표현됨(ex. 20231019\_063927) [↑](#footnote-ref-4)
5. {now}는 이력서 직무 예측 실행 시점을 의미하며 시간 형식(yyyymmdd\_hhmmss)으로 표현됨(ex. 20231019\_063927) [↑](#footnote-ref-5)