Dart

# Dart 사업보고서 리스트 크롤링(crawler\_dart\_crp\_list.py)

### **필요한 패키지 다운로드**

|  |
| --- |
| 1. import pandas as pd 2. import numpy as np 3. from bs4 import BeautifulSoup 4. from urllib.request import urlopen 5. import webbrowser 6. import requests 7. import re 8. import datetime 9. import sys |

### **프로그레스바**

|  |
| --- |
| 1. def progressBar(value, endvalue, bar\_length=20): 2. percent = float(value) / endvalue 3. arrow = '-' \* int(round(percent \* bar\_length) - 1) + '>' 4. spaces = ' ' \* (bar\_length - len(arrow)) 5. sys.stdout.write("\rPercent: [{0}] {1}%".format(arrow + spaces, int(round(percent \* 100)))) 6. sys.stdout.flush() |

### **상장 법인 목록에 있는 기업들의 사업보고서 목록 크롤링하는 함수 정의**

* 예시 url: <http://dart.fss.or.kr/api/search.xml?auth=a2f0104fc81d7dc05f8817768aae6c55169516a6&crp_cd=005930&start_dt=20140101&bsn_tp=A001>
* 필요한 변수들:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 필수 | start\_date | 검색 시작 날짜(예시:20140101) |
| 2 | 필수 | end\_date | 검색 종료 날짜(예시:20181231) |
| 3 | 선택 | crp\_name | 확인하고 싶은 회사 이름, 입력하지 않으면 상장법인목록에 있는 모든 회사 |

* 반환값은 데이터 프레임
* 사업보고서만 가져온다. (A001: 사업보고서, A002: 반기보고서, A003: 분기보고서, A004: 등록법인결산서류(자본시장법이전), A005: 소액공모법인 결산서류)

|  |
| --- |
| 1. def crawler\_dart\_crp\_list(start\_date, end\_date, crp\_name=None): 2. print("---사업보고서 리스트 가져오기---") 3. if crp\_name is not None: 4. company\_codes = pd.DataFrame(company\_codes.iloc[list(np.where(company\_codes["회사명"]==crp\_name)[0])]) 6. crp\_cd\_list = company\_codes["종목코드"] 7. df = pd.DataFrame(columns=["crp\_nm", "crp\_cd", "rcp\_dt", "rpt\_nm", "rcp\_no"]) 8. start = datetime.datetime.now() 10. for com\_idx in range(0, len(company\_codes)): 11. url = "http://dart.fss.or.kr/api/search.xml?auth=" + auth\_key + "&crp\_cd=" + crp\_cd\_list.iloc[ 12. com\_idx] + "&start\_dt=" + str(start\_date) + "&end\_dt=" + str(end\_date) + "&bsn\_tp=A001" 13. # 현재 url의 XML의 response를 읽는다 14. resultXML = urlopen(url) 15. result = resultXML.read() 16. xmlsoup = BeautifulSoup(result, 'html.parser') 17. # 현재 url에서 존재하는 한 회사의 사업보고서 리스트 18. data = pd.DataFrame() # 사업보고서 리스트를 넣을 데이터 19. te = xmlsoup.findAll("list") 20. if len(te) == 0: continue 21. tmp = '' 22. # 현재 페이지 소스에서 리스트 당 필요한 변수들 저장하기 23. for t in te: 24. if tmp == t.rpt\_nm.string.split()[1]: continue 25. temp = pd.DataFrame(([[t.crp\_nm.string, t.crp\_cd.string, t.rcp\_dt.string, 26. t.rpt\_nm.string, t.rcp\_no.string]]), 27. columns=["crp\_nm", "crp\_cd", "rcp\_dt", "rpt\_nm", "rcp\_no"]) 28. data = pd.concat([data, temp]) 29. tmp = t.rpt\_nm.string.split()[1] 31. data = data.reset\_index(drop=True) 32. df = pd.concat([df, data]) 34. progressBar(com\_idx, len(company\_codes)) 36. end = datetime.datetime.now() 37. print("\n걸린시간 : ", end - start) 39. return df |

### 진짜 크롤링

* API key: 자신의 API key가 적혀있는 텍스트 파일 이름을 확장자 빼고 입력하기

My txt file name for API key : api\_key\_suyoung

* 상장법인목록: KIND 의 상장법인목록의 종목코드 이용(아래 사이트에서 엑셀 파일 다운로드)
* <http://kind.krx.co.kr/corpgeneral/corpList.do?method=loadInitPage>
* 결과를 저장할 파일 이름을 확장자 빼고 입력하기

My result file name: samsung\_list

* 함수 crawler\_dart\_crp\_list(검색시작일자, 검색종료일자, ‘회사이름’)

result = crawler\_dart\_crp\_list(20150101,20181231,'삼성전자')

|  |
| --- |
| 1. if \_\_name\_\_ == "\_\_main\_\_": 2. ## API key 3. with open(input("My txt file name for API key : ")+".txt",'r') as f: 4. auth\_key = f.read() 6. ## 상장법인목록 7. company\_codes = pd.read\_excel('상장법인목록.xlsx', converters={'종목코드': str})[["회사명","종목코드"]] 9. ## 결과 파일 이름 10. filename = input("My result file name : ") 12. result = crawler\_dart\_crp\_list(20150101,20181231,'삼성전자') # 예시 14. # 데이터(사업보고서 리스트) 엑셀로 저장 15. writer = pd.ExcelWriter(filename+'.xlsx') 16. result.to\_excel(writer, 'Sheet1') 17. writer.save() |

# Dart 사업보고서 리스트 크롤링(crawler\_dart\_text.py)

### **필요한 패키지 다운로드**

|  |
| --- |
| 1. import pandas as pd 2. import numpy as np 3. from bs4 import BeautifulSoup 4. from urllib.request import urlopen 5. import webbrowser 6. import requests 7. import re 8. import datetime 9. import sys |

### **프로그레스바**

|  |
| --- |
| 1. def progressBar(value, endvalue, bar\_length=20): 2. percent = float(value) / endvalue 3. arrow = '-' \* int(round(percent \* bar\_length) - 1) + '>' 4. spaces = ' ' \* (bar\_length - len(arrow)) 5. sys.stdout.write("\rPercent: [{0}] {1}%".format(arrow + spaces, int(round(percent \* 100)))) 6. sys.stdout.flush() |

### **텍스트 크롤링에 필요한 함수 정의1**

* 접수번호(rcp\_no)에 해당하는 모든 하위 보고서 URL을 추출하여 리스트로 반환하는 함수
* 예시 url: <http://dart.fss.or.kr/dsaf001/main.do?rcpNo=20181228000183>
* 예시 doc\_url\_tmpl: view-source:http://dart.fss.or.kr/report/viewer.do?rcpNo=20181228000183&dcmNo=6431765&eleId=19&offset=402114&length=10395&dtd=dart3.xsd

|  |
| --- |
| 1. def get\_sub\_report\_urls(rcp\_no): 2. doc\_urls = [] 3. url = "http://dart.fss.or.kr/dsaf001/main.do?rcpNo=%s" % (rcp\_no) 4. r = requests.get(url) 5. reg = re.compile('viewDoc\((.\*)\);') 6. params = [] 7. matches = reg.findall(r.text) 8. for m in matches: 9. params.append(m.replace("'", "").replace(" ", "").split(",")) 10. doc\_url\_tmpl = "http://dart.fss.or.kr/report/viewer.do?rcpNo=%s&dcmNo=%s&eleId=%s&offset=%s&length=%s&dtd=%s" 11. for p in params: 12. if rcp\_no == p[0]: 13. doc\_urls.append(doc\_url\_tmpl % tuple(p)) 14. return doc\_urls |

### **해당 url에서 텍스트를 가져오는 함수**

* 'V. 이사의 경영진단 및 분석의견' 텍스트를 포함한 글(get\_sub\_report\_urls의 리스트 중 하나) 찾기
* 보통의 형식에서는 idx=18에 있는데, 정정보고서 등은 이 양식을 지키지 않아 새롭게 찾고 없으면 빈칸

|  |
| --- |
| 1. def get\_text\_urls(rcp\_no): 2. url = get\_sub\_report\_urls(rcp\_no) 3. le = len(url) 4. html\_text = requests.get(url[18]) 6. if "V. 이사의 경영진단 및 분석의견" not in html\_text: 7. idx = 0 8. while True: 9. if le == 0 or idx == le: 10. html\_text = '' 11. break 12. url2 = url[idx] 13. r = requests.get(url2) 14. html\_text = r.text # 해당 url의 텍스트 저장 15. # 없으면 다시 찾기 16. if "V. 이사의 경영진단 및 분석의견" in html\_text: 17. break 18. else: 19. idx += 1 21. return html\_text |

### **사업보고서 리스트로 텍스트를 가져오는 함수**

* 위의 함수를 모두 이용하게 됨
* 접수번호의 url에 접속하여 이사회의 분석의견 텍스트를 가져온다.
* 이사의 분석의견 자체가 없을 경우 해당 사업보고서는 가져오지 않는다.
* ‘이사의 경영진단 및 분석의견’에 ‘해당사항없습니다’라고 나타나는 경우는 가져오게 된다.
* 반환값은 데이터 프레임

|  |
| --- |
| 1. def crawler\_dart\_text(crp\_lists): 3. txt = pd.DataFrame(columns=["회사명", "회사코드", "접수날짜", "보고서 명", "접수번호", "text"]) 4. start = datetime.datetime.now() 6. for idx in range(0, len(crp\_lists)): 7. crp = crp\_lists[idx] # idx번째 사업보고서의 변수 리스트(회사명 등, crp[4]는 접수번호) 8. html\_txt = get\_text\_urls(str(crp[4])) # 접수번호의 url에 접속하여 텍스트 가져오기 10. try: 11. a = len(html\_txt) 12. except: 13. a = 0 14. if a < 1: 15. continue 17. txt2 = pd.Series([crp[0], crp[1], crp[2], crp[3], crp[4], html\_txt], 18. index=["회사명", "회사코드", "접수날짜", "보고서 명", "접수번호", "text"]) 19. txt = txt.append(txt2, ignore\_index=True) 21. progressBar(idx, len(crp\_lists)) 23. end = datetime.datetime.now() 24. print("\n걸린시간 : ", end - start) 25. return txt |

### **진짜진짜 크롤링**

* 이제 위에서 정의한 함수들을 토대로 텍스트를 가져와보자.
* 크롤링했던 회사들의 사업보고서 리스트로부터 해당 사업보고서의 접수번호를 가져와 크롤링한다.
* My crp\_list file name 에 사업보고서 리스트 파일 이름 입력
* My result file name 에 결과를 저장할 파일 이름 입력

|  |
| --- |
| 1. if \_\_name\_\_ == "\_\_main\_\_": 2. # 크롤링했던 회사들의 사업보고서 리스트 가져오기 3. crplist\_filename = input("My crp\_list file name : ") 4. crp\_lists = pd.read\_excel(crplist\_filename+".xlsx") 5. crp\_lists = dat[['crp\_nm','crp\_cd','rcp\_dt','rpt\_nm','rcp\_no']].values.tolist() 7. dart\_text = crawler\_dart\_text(crp\_lists) 9. ## 결과를 저장할 파일 이름(확장자 빼고) 10. filename = input("My result file name: ") 12. # 데이터(사업보고서 리스트) 엑셀로 저장 13. writer = pd.ExcelWriter(filename+'.xlsx') 14. dart\_text.to\_excel(writer, 'Sheet1') 15. writer.save() 17. # 데이터 확인 18. dart\_text.head() |

# Dart 단어수 세기(text\_pos\_neg.py)

* 네이버 애널리스트 자료의 단어수, 긍부정 단어수를 세는 함수와 동일

# Dart 단어수 세기2(analyzer\_dart\_text.py)

* 위와 다른 방법으로 단어수를 셈. 사용하지 않아도 무관

### **필요한 패키지 다운로드**

|  |
| --- |
| 1. import pandas as pd 2. from pandas import Series, DataFrame 3. import sys 4. import datetime 5. import re 6. from nltk.tokenize import sent\_tokenize 7. import nltk 8. from nltk.corpus import stopwords 9. from nltk.tokenize import word\_tokenize 10. import re 11. from konlpy.tag import Twitter 12. tagger = Twitter() # Twitter 태깅 함수 |

### **프로그레스바**

|  |
| --- |
| 1. def progressBar(value, endvalue, bar\_length=20): 2. percent = float(value) / endvalue 3. arrow = '-' \* int(round(percent \* bar\_length) - 1) + '>' 4. spaces = ' ' \* (bar\_length - len(arrow)) 5. sys.stdout.write("\rPercent: [{0}] {1}%".format(arrow + spaces, int(round(percent \* 100)))) 6. sys.stdout.flush() |

### **텍스트 데이터 받기**

|  |
| --- |
| 1. data = pd.read\_excel("dart\_text\_list.xlsx", encoding = 'UTF-8') 2. text = data["텍스트"] |

### **텍스트 전처리 1: 불필요한 html 소스 제거**

* 토큰화했을 때 '이사의 경영진단 및 분석의견'이 포함된 텍스트 이후로만 가져옴
* 올바르게 단어수를 세기 위해 가능한 숫자와 의미있는 영어문자는 제거하지 않음
* 한글만, 혹은 빠른 전처리를 원한다면 위의 text\_pos\_neg.py 파일 실행
* 한글이 아닌 텍스트를 제거하기 전 의미있는 텍스트들을 따로 처리(예: R&D는 의미있을 수 있음)

|  |
| --- |
| 1. processed\_text = [] # 전처리한 데이터를 저장할 빈 리스트 2. start = datetime.datetime.now() 3. for tmp in text: 4. sent\_text = sent\_tokenize(tmp) # 문장단위 토큰화 5. for text\_idx in range(0, len(sent\_text)): 6. if '이사의 경영진단 및 분석의견' in sent\_text[text\_idx]: 7. sent\_text = sent\_text[text\_idx + 1:] 8. break 9. ## 텍스트 전처리 10. sent\_text = [re.sub("\r", ' ', text) for text in sent\_text] 11. sent\_text = [re.sub("\n", ' ', text) for text in sent\_text] 12. sent\_text = [re.sub("<.{1,}>", ' ', text) for text in sent\_text] 13. sent\_text = [re.sub(" .\.", ' ', text) for text in sent\_text] 14. sent\_text = [re.sub("\d+[\.\,]\d\*[\.\,]\*\d\*", '숫자 ', text) for text in sent\_text] 15. ## 한글이 아닌 텍스트 중 의미있는 것들을 미리 처리 16. sent\_text = [re.sub("&amp;", '앤', text) for text in sent\_text] # &가 &amp;로 읽힘 17. sent\_text = [re.sub("&nbsp;", ' ', text) for text in sent\_text] # 공백이 &nbsp;로 읽힘 18. ## 아래부터 특수문자 제거 19. sent\_text = [re.sub("\.", ' ', text) for text in sent\_text] 20. sent\_text = [re.sub('[ⅢXI,;\(\)%\'\"」』『:\“\”ㆍ-]', ' ', text) for text in sent\_text] 21. sent\_text = [re.sub(" +", ' ', text) for text in sent\_text] 22. sent\_text = [' '.join(sent\_text)] # 토큰화된 텍스트들의 리스트를 다 붙임(대신 문장 구분이 없어짐) 23. processed\_text.append(sent\_text) # 각 사업보고서들의 텍스트를 리스트로 받아서 넣음 24. progressBar(len(processed\_text), len(text)) 25. end = datetime.datetime.now() 26. print("\n걸린시간 : ", end - start) |

### **데이터 확인해보기**

|  |
| --- |
| 1. # 원래 데이터 2. text[0] 3. # 전처리한 데이터 4. processed\_text[0] |

### **단어 수 구하기**

* word\_tokenize는 space, punctuation기준으로 나눔, 위에서 구두점을 제거하였으므로 공백 기준으로 나눔 즉, 단어들로 나눔

|  |
| --- |
| 1. # 아래 반복문을 돌리기 전 예시로 돌려보기 2. tmp = processed\_text[0] 3. text\_token = [word\_tokenize(text) for text in tmp] # 단어들로 나눈다 4. text\_token[0] # 단어들 리스트 5. len(text\_token[0]) # 단어 수 |

* 약 4천개에 12분 정도

|  |
| --- |
| 1. word\_cnt, txt\_token = [], [] # 데이터 저장 빈 텍스트 2. start = datetime.datetime.now() 3. for tmp in processed\_text: 4. text\_token = [word\_tokenize(text) for text in tmp] 5. # text\_token = [y for x in text\_token for y in x] 6. txt\_token.append(text\_token[0]) 7. word\_cnt.append(len(text\_token[0])) # 단어들로 나뉜 토큰들의 길이 즉, 단어수 8. progressBar(len(word\_cnt), len(processed\_text)) 9. end = datetime.datetime.now() 10. print("\n걸린시간 : ", end - start) |

### **데이터 확인해보기**

|  |
| --- |
| 1. print(len(word\_cnt)) 2. word\_cnt[:5] # 단어수 3. text\_token[0] # 단어들 |

### **긍부정 사전 데이터 받기**

* 긍정단어와 부정단어 사전을 받은 후 단어들의 리스트로 만든다.

|  |
| --- |
| 1. # 긍정단어 목록 받기 2. # knu의 긍정 사전이나, 필요할 시 다른 사전을 가져와도 무방 3. f = open("pos\_pol\_word.txt", 'r', encoding = 'UTF-8') 4. pos = f.read() 5. f.close() 6. # 부정단어 목록 받기 7. f = open("neg\_pol\_word.txt", 'r', encoding = 'UTF-8') 8. neg = f.read() 9. f.close() 10. # 긍정단어 목록(사전)을 단어들의 리스트로 만듦 11. pos\_dic = pos.split('\n') # 사전을 단어들의 리스트로 만듦 12. pos\_dic[0] = re.sub("\ufeff",'',pos\_dic[0]) # 인코딩 오류 13. print(pos\_dic[:10]) # 긍정단어 목록 확인 14. # 부정단어 리스트 15. neg\_dic = neg.split('\n') 16. neg\_dic[0] = re.sub("\ufeff",'',neg\_dic[0]) 17. neg\_dic[:10] 18. # 여기서 단어수를 확인해보자 ----- 토큰화하는 함수마다 달라서 확인 필요 19. print(len(tagger.nouns(processed\_text[0][0]))) 20. tagger.nouns(processed\_text[0][0])[0] |

### **사업보고서에서 텍스트의 명사 추출하기**

* 위의 긍부정 사전의 단어들과 비교하기 위함.
* 용이한 비교를 위해 2음절 이상의 명사만 가져온다.

|  |
| --- |
| 1. start = datetime.datetime.now() 2. text\_tag = [] 3. for tmp in processed\_text: 4. text\_token = tagger.nouns(tmp[0]) 5. text\_token = [text for text in text\_token if len(text) >= 2] 6. text\_tag.append(text\_token) 7. progressBar(len(text\_tag), len(processed\_text)) 8. end = datetime.datetime.now() 9. print("\n걸린시간 : ", end - start) |

### **사업보고서 텍스트의 명사 전처리**

* 품사 태깅에서 단어만 가져온 후 불용어들을 제거하고 단어들을 리스트에 넣는다.
* 불용어 리스트 (본인의 판단에 따라 더 추가 가능) : 텍스트 분석에 필요가 없거나 사업보고서 특성상 당연히 많이 나와 분석에 의미 없는 단어들

|  |
| --- |
| 1. # nltk 모듈을 이용한 텍스트 단어들 2. text\_list = [] 3. for tmp in text\_tag: 4. tmp2 = nltk.Text(tmp) 5. text\_list.append(tmp2) 6. ## 불용어 제거 7. text\_vocab, text\_ko = [], [] 8. stop\_words = ['사항','제기','사업', '대하'] 9. for tmp in text\_list: 10. ko = [word for word in tmp if word not in stop\_words] # 불용어 제거 11. ko = nltk.Text(ko) # 텍스트의 단어들만 가져옴 12. text\_ko.append(ko) 13. text\_vocab.append(ko.vocab()) 14. # 문서의 단어들 15. doc\_words = [list(ko) for ko in text\_ko] |

### **데이터 확인해보기**

|  |
| --- |
| 1. # 단어 확인(가장 많이 나오는 단어 순) 2. text\_list[4000].vocab() 3. # 단어 4. text\_vocab[:5] 5. # 문서의 단어 목록 6. doc\_words[:2] # 예측, 정보, 대한, 주의, 당사, 보고서 등 |

### **긍부정단어 목록에서 단어들 가져오기**

* 용이한 비교를 위해 2음절 이상의 명사만 추출
* 감사한, 감사하다 모두 명사만 추출하면 감사, 감사가 되어 중복되므로 중복 제거

|  |
| --- |
| 1. # 긍정 사전 2. pos\_words = [tagger.nouns(doc) for doc in pos\_dic if len(tagger.nouns(doc))!=0] #긍정사전의 단어들로부터 명사 추출, 명사 없으면 스킵 3. pos\_words = [re.findall(r'[가-힣]{2,10}', word) for y in pos\_words for word in y] #추출된 명사들 중 2음절 이상인 명사만 추출 4. pos\_words = [y for x in pos\_words for y in x] #unlist 5. pos\_words = set(pos\_words) # 세트 형 구조로 만듦(중복되지 않게) 6. pos\_words = [word for word in pos\_words if word not in stop\_words] # 불용어 없애고 리스트로 만듦 7. pos\_words[:10] # 원동력, 기상, 애하, 가르침, 더없이 등 8. # 부정 사전 9. neg\_words = [tagger.nouns(doc) for doc in neg\_dic if len(tagger.nouns(doc))!=0] 10. neg\_words = [re.findall(r'[가-힣]{2,10}', word) for y in neg\_words for word in y] 11. neg\_words = [y for x in neg\_words for y in x] 12. neg\_words = set(neg\_words) 13. neg\_words = [word for word in neg\_words if word not in stop\_words] 14. neg\_words[:10] # 불량, 요사, 모순, 선천, 실행 |

### **긍정단어 개수**

* 긍정 단어가 사업보고서 텍스트의 단어 리스트에 몇 개가 포함되었는가
* 각 긍정단어가 사업보고서 텍스트의 단어들에 있는지 논리값을 가져오고 개수 세기. 오래 걸림.

|  |
| --- |
| 1. start = datetime.datetime.now() 2. pos\_words\_cnt, cnt = [], 0 3. for ko in text\_ko: 4. cnt = sum([word in ko for word in pos\_words]) 5. pos\_words\_cnt.append(cnt) 6. progressBar(len(pos\_words\_cnt), len(text\_ko)) 7. end = datetime.datetime.now() 8. print("\n걸린시간 : ", end - start) 9. # 단어 수 확인 10. pos\_words\_cnt[:5] |

### **부정단어 개수**

* 부정 단어가 사업보고서 텍스트의 단어 리스트에 몇 개가 포함되었는가
* 여기서 부정사전 내 단어수가 긍정사전보다 많으므로 부정단어의 수가 더 많을 수밖에 없다.

|  |
| --- |
| 1. start = datetime.datetime.now() 2. neg\_words\_cnt, cnt = [], 0 3. for ko in text\_ko: 4. cnt = sum([word in ko for word in neg\_words]) 5. neg\_words\_cnt.append(cnt) 6. progressBar(len(neg\_words\_cnt), len(text\_ko)) 7. end = datetime.datetime.now() 8. print("\n걸린시간 : ", end - start) 9. # 단어 수 확인 10. neg\_words\_cnt[:5] |

### **데이터 프레임 합치기**

|  |
| --- |
| 1. # 데이터프레임으로 합칠 데이터 형태(필요한 변수들) 2. data2 = data[['회사명','회사코드','보고서 명','접수번호','접수날짜']] 3. proc\_text = pd.Series(processed\_text, name = '텍스트') 4. word\_cnt = pd.Series(word\_cnt, name = '단어수') 5. pos\_words\_cnt = pd.Series(pos\_words\_cnt, name = '긍정단어수') 6. neg\_words\_cnt = pd.Series(neg\_words\_cnt, name = '부정단어수') 7. data2 = pd.concat([data2, proc\_text, word\_cnt, pos\_words\_cnt, neg\_words\_cnt], axis=1) 8. data2.head() 9. data2.tail() 10. ## 엑셀로 저장 11. writer = pd.ExcelWriter('./dart\_text\_wordcnt.xlsx') 12. data2.to\_excel(writer,'Sheet1') 13. writer.save() |