전체적인 프로그램 구조입니다. 더 자세한 설명은 코드내에 주석으로 달아두었습니다.

Naver finance

# Naver Finance 크롤링(crawler\_naver\_finance.py)

1. 데이터를 크롤링하는 부분

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104 |  | # - driver\_path : chromedriver.exe가 설치되어 있는 경로  # - path : 엑셀파일이 저장될 위치(None : 현재위치)  # - start\_date, end\_date : 검색기간 설정(None : 전체기간)  # - pageNum : 가져올 페이지 수 (None : 검색된 전체 페이지)    def naver\_finace\_crawling(driver\_path,path = None,start\_date=None,end\_date=None,pageNum=None):      try:          driver = webdriver.Chrome(driver\_path)      except:          print("chrome driver 경로를 확인해주세요")          return False,False        if path is None:          file\_path = "./"      else:          file\_path = path      try:          os.mkdir(file\_path)          print("%s폴더 생성"%path)      except:          print("path 있음")          driver.implicitly\_wait(3)        page = 1      stock\_list = [] # 데이터 저장      url\_date = "searchType=writeDate"      url = 'https://finance.naver.com/research/company\_list.nhn?'      #end\_flag = False;        if start\_date is not None:          url = url + url\_date + '&writeFromDate=' +start\_date +"&writeToDate=" + end\_date          if end\_date is None:              print("start date가 있으면 end date도 필요합니다")              return False, False        while True:            # 페이지수를 입력받은 경우          if pageNum is not None:              if pageNum == 0:                  break              pageNum -= 1            c\_url = url +'&page=' + str(page)          print(c\_url)          driver.get(c\_url) # 해당 url로 이동            html = driver.page\_source            # 테이블 정보 get          table = driver.find\_elements\_by\_xpath('//\*[@id="contentarea\_left"]/div[3]/table[1]/tbody/tr')            for i,t in enumerate(table):                # 테이블의 행을 읽는다. 이때 줄바꿈 행도 읽기 때문에 rows의 길이 확인              rows = t.find\_elements\_by\_tag\_name('td')              if len(rows) == 6:                  temp = ['0' for \_ in range(6)] # 저장할 공간을 미리 할당                    for k,row in enumerate(rows):                      temp[k] = row.text                        # pdf경로는 tag가 다름                      if row.get\_attribute("class") == "file":                          try:                              temp[k] = row.find\_element\_by\_tag\_name("a").get\_attribute('href')                          except:                              temp[k] = None                    stock\_list.append(temp)            # 마지막 페이지인 경우 stop          pagination = driver.find\_element\_by\_xpath('//\*[@id="contentarea\_left"]/div[3]/table[2]/tbody/tr/td[@class="on"]')          if pagination.text != str(page):              break          page += 1        driver.quit()      # DataFrame으로 변환 -> excel저장 용이      try :          stocks = DataFrame(stock\_list)      except:          print(stock\_list)          print("저장할 데이터가 없거나 형식이 잘못되었습니다")          return False, stock\_list        stocks.columns = ["종목명","제목","증권사","pdf","작성일","조회수"]      #엑셀로 저장      # 시간정보를 파일명에 더해서 덮여쓰여지는 것을 방지(존재하는 파일명을 사용하면 덮어쓰기된다)      file\_path = os.path.join(file\_path,"naver\_fiance" +format(time.time(),'.0f')+ ".xlsx")      print(file\_path)        try:          writer = pd.ExcelWriter(file\_path)          stocks.to\_excel(writer,'Sheet1')          writer.save()          print("%s에 저장했습니다" % file\_path)        except:          print("\n저장하지 못했습니다")        return file\_path, stocks  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. 함수 실행

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | if \_\_name\_\_ == "\_\_main\_\_":      chromedriver = input("Chromedriver 위치 : ")      excel\_path = input("네이버 경제 데이터 저장받을 위치( n - 현재 위치): ")      start\_date = input("검색조건 - 시작날짜(yyyy-mm-dd) ( n  - all): ")      end\_date = input("검색조건 - 종료날짜(yyyy-mm-dd) ( n - all): ")      pagenum = input("페이지 수 ( n - all) : ")        if excel\_path == 'n':          excel\_path = None        if start\_date == 'n':          start\_date = None        if end\_date == 'n':          end\_date = None        if pagenum == 'n':          pagenum = None      else:          pagenum = int(pagenum)        filename, data = naver\_finace\_crawling(chromedriver,excel\_path,start\_date,end\_date,pagenum) | [cs](http://colorscripter.com/info#e) |

# pdf다운로드 (pdfdown.py)

### **Pdf 다운**

( 다운받지 못한 list를 반환해서 메인함수에서 저장)

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | ## pdf다운로드 함수  # - data : excel데이터  # - dir\_path : pdf를 저장할 경로  def pdf\_download(data,dir\_path=None):        pdf\_list = []        ## pdf 경로를 지정해주지 않으면 임의의 경로에 저장      if dir\_path is None:          down\_path = "./pdf"      else:          down\_path = dir\_path        ## 다운받을 경로폴더 생성      try:          os.mkdir(down\_path)      except:          print("폴더 있음")        for i,pdf\_url in enumerate(data["pdf"]):          progressBar(i,len(data["pdf"]))          try:              filename = os.path.join(down\_path,pdf\_url.split('/')[-1])          except:              #pdf가 없는 경우              pass          try:              ## url로 이동해서 pdf다운              request.urlretrieve(pdf\_url,filename)          except:              ## url이 잘못된 경우              pdf\_list.append(pdf\_url)              print("예외 발생")      return pdf\_list  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. **함수실행Pdf를 text변환 (pdfread.py)**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | if \_\_name\_\_ == "\_\_main\_\_":      excelfile = input("excel파일명 : ")      downpath = input("pdf 저장받을 위치( n - ./pdf): ")        if downpath == 'n':          downpath = None        ## excel파일 열기      excel\_data = pd.read\_excel(excelfile)      ## pdf다운      pdfs = pdf\_download(excel\_data,downpath)        ## 잘못된 url이 있는 목록 저장      if pdfs is not None:          with io.open(os.path.join("다운받지못한pdf"+".txt"),'w',encoding='utf8') as f:              for pt in pdfs:                  f.write(pt)                  f.write("\n")              f.close()  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

# Pdf를 text 변환(pdfread.py)

### **Pdf를 text로 읽음**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | # - data : pdf파일  # - txtfp : 읽지못한 pdf목록을 open한 것    def pdfparser(data,txtfp):        fp = open(data, 'rb')      rsrcmgr = PDFResourceManager()      retstr = io.StringIO()      codec = 'utf-8'      laparams = LAParams()      device = TextConverter(rsrcmgr, retstr, codec=codec, laparams=laparams)      # Create a PDF interpreter object.      interpreter = PDFPageInterpreter(rsrcmgr, device)      # Process each page contained in the document.      filename = data      for i,page in enumerate(PDFPage.get\_pages(fp)):          try:              interpreter.process\_page(page)              data =  retstr.getvalue()          except:              print(filename+"파일 %d번째 페이지 읽지 못함"% (i+1))              txtfp.write(filename+"파일 %d번째 페이지 읽지 못함"% (i+1))              txtfp.write("\n")              pass        return data  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

### **폴더에서 읽지 않은 pdf목록을 확인**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | # - dirname : pdf파일 경로  # - savefolder : pdf to txt를 저장할 경로    def pdfread(dirname,savefolder=None):        # 해달 폴더에 있는 파일명 가져옴      file\_list = os.listdir(dirname)      path\_filenames = [dirname+'/'+i for i in file\_list]        # text파일 저장할 폴더 생성      if savefolder is None:          down\_path = "./textfolder"      else:          down\_path = savefolder      try:          os.mkdir(down\_path)      except:          print("savefolder 있음")        # 확장자명을 제외한 파일명만 추출      text\_list = os.listdir(down\_path)      text\_list = [os.path.splitext(i)[0] for i in text\_list]        print("총 %d개의 파일 존재"%len(file\_list))        f = io.open(os.path.join("읽지못한pdf"+".txt"),'w',encoding='utf8')        for i,file in enumerate(file\_list):          if file in text\_list:              continue            # pdf 파일만 read          if '.pdf' == os.path.splitext(file)[-1]:              progressBar(i,len(file\_list))              try:                  pdf2txt = pdfparser(path\_filenames[i],f)              except:                  print("%s 읽지못함"%file)                  continue                try:                  #txt파일 저장                  with io.open(os.path.join(down\_path,file+".txt"),'w',encoding='utf8') as pdf\_f:                      pdf\_f.write(pdf2txt)                  pdf\_f.close()              except:                  f.write('%s를 txt파일 저장 실패'%file)                  f.write('\n')                  print('%s를 txt파일 저장 실패'%file)        f.close()  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

### **함수실행**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | if \_\_name\_\_ == "\_\_main\_\_":      pdfpath = input("pdf 위치 : ")      downpath = input("txt 저장받을 위치( n - ./textfolder): ")        if downpath == 'n':          downpath = None        # ex) ./temp, ./temp\_text      pdftext = pdfread(pdfpath,downpath)      if pdftext is not None:          with io.open(os.path.join("읽지못한pdf"+".txt"),'w',encoding='utf8') as f:              for pt in pdftext:                  f.write(pt)                  f.write("\n")              f.close()    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

# 각 text들을 하나로 합침 (combine\_txt.py)

1. txt파일을 엑셀파일에 함친다

합치면서 단어만 존재하거나 공백만 존재하는 행을 삭제한다(표데이터라고 생각)

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | ## pdf를 읽어서 txt파일로 변환한 파일들을 엑셀파일로 합치는 코드    def read\_text(dirname):      # 디렉토리에 있는 파일이름 읽음      text\_list = os.listdir(dirname)      pdf\_txt = []        for text in text\_list:          # txt파일만          if '.txt' == os.path.splitext(text)[-1]:              with io.open(os.path.join(dirname,text),'r',encoding='utf8') as f:                  text\_data = ''                  # txt파일 read                  while(1):                      line = f.readline()                      if line:                          # 단어만 존재 or 공백만 존재하면 제외                          if len(re.findall("[\S]",line)) == 0:                              pass                          if len(re.findall("[\ ]",line)) < 2:                              pass                          else:                              text\_data = text\_data + line                      else:                          break                pdfname = os.path.splitext(text)[0]              pdf\_txt.append([pdfname,text\_data])        # 데이터프레임으로 변환      pdf\_txt = DataFrame(pdf\_txt)      pdf\_txt.columns = ["pdf명","text"]        return pdf\_txt  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. 함수호출

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | if \_\_name\_\_ == "\_\_main\_\_":      dirname = input("txt폴더 : ")      excel\_path = input("데이터를 합칠 excel파일 :")        process\_textdata = read\_text(dirname)        excel\_data = pd.read\_excel(excel\_path)      excel\_data['pdf명'] = [i.split("/")[-1] for i in excel\_data['pdf']]      excel\_data        data = pd.merge(excel\_data,process\_textdata,how = 'left')      writer = pd.ExcelWriter(os.path.splitext(excel\_path)[0]+"\_text.xlsx")      data.to\_excel(writer,'Sheet1')      writer.save()  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

# 분석가 찾기(find\_analyst.py)

1. txt파일을 읽기

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | # analyst 를 찾기위해 전부 읽어들이기  # 이때는 전처리를 하지않고 읽기  def analyst\_read\_text(dirname):      # 디렉토리에 있는 파일이름 읽음      text\_list = os.listdir(dirname)      pdf\_txt = []      for text in text\_list:          # txt파일만          if '.txt' == os.path.splitext(text)[-1]:              with io.open(os.path.join(dirname,text),'r',encoding='utf8') as f:                  text\_data = ''                  # txt파일 read                  while(1):                      line = f.readline()                      if line:                          text\_data = text\_data + line                      else:                          break              pdfname = os.path.splitext(text)[0]              pdf\_txt.append([pdfname,text\_data])      # 데이터프레임으로 변환      pdf\_txt = DataFrame(pdf\_txt)      pdf\_txt.columns = ["pdf명","text"]      return pdf\_txt  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. analyst를 중심으로 분석가명 찾기

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | #analyst라고 명시되어 있는 분삭가의 찾기  def analyst\_find(textdata):      analysts = [None for \_ in range(len(textdata))]      analysts\_check= [None for \_ in range(len(textdata))]      # text파일에서 이름이 포함된 부분을 추출      for i,pdf in enumerate(textdata["text"]):          # Analyst와 analyst 같이 대 소문자가 다를 수 있기 때문에          pdf = str(pdf).lower() #소문자변환          pdf = sent\_tokenize(pdf) #문장단위 토큰화          name = ''          for sent in pdf:              # analyst라는 단어를 찾으면 그 문장을 저장              if sent.find("analyst") >=0:                  name = sent                  break          #name을 못찾은 경우          if name is None:              print("없음")              continue          else:              name = name.split('\n')          L = []          for j,n in enumerate(name):              # analyst가 있는 주변을 추출              if n.find('analyst') >=0:                  if len(name) == j+1:                      L = [n,'']                  else:                      L = [n,name[j+1]]          analysts[i] = L      # 이름이 아닌 부분을 제거      pre\_analysts = [None for \_ in range(len(textdata))]      for i, analyst in enumerate(analysts):          if analyst is None:              continue          # 한글빼고 제거          analyst = [re.sub("[^가-힣]",'',a) for a in analyst]          for a in analyst:              if a is not "":                  pre\_analysts[i] =  a                  if len(pre\_analysts[i]) == 6:                      #사람이름+사람이름                      analysts\_check[i] = 'check'                      pre\_analysts[i] =  pre\_analysts[i][:3]+','+pre\_analysts[i][-3:]                      continue                  if len(pre\_analysts[i]) > 6:                      # 회사명+사람이름                      analysts\_check[i] = 'check'                      pre\_analysts[i] = pre\_analysts[i][-3:]      # 데이터프레임 생성      pre\_analysts = DataFrame(pre\_analysts)      pre\_analysts.columns = ["분석가"]      analysts\_check = DataFrame(analysts\_check)      analysts\_check.columns = ["확인해 볼 분석가"]    return pd.merge(pre\_analysts,analysts\_check,left\_index=True,right\_index=True) | [cs](http://colorscripter.com/info#e) |

1. 이메일을 중심으로 분석가명 찾기

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | ## email형식을 찾아서 분석가 찾기  def analyst\_email\_find(textdata,analysts\_df):      # 정규표현식으로 이메일찾기      p = re.compile('\w+[@]\w+[.]')      for i,text in enumerate(textdata['text']):          if analysts\_df['분석가'][i] is None:              a\_email = p.search(str(text))              if a\_email is not None:                  # 이메일이 적힌 위치 앞                  name = text[a\_email.start()-30:a\_email.start()]                  analysts\_df['분석가'][i] = re.sub("[^가-힣]",'',name)                  if len(analysts\_df['분석가'][i]) > 3:                      # 3글자 이상이면 회사명+사람이름 이라 생각하고 전처리                      analysts\_df["확인해 볼 분석가"][i] = 'check'                      analysts\_df['분석가'][i] = analysts\_df['분석가'][i][-3:]      return analysts\_df  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. 함수실행 및 데이터 합쳐서 엑셀저장

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | if \_\_name\_\_ == "\_\_main\_\_":      dirname = input("txt폴더 : ")      excel\_path = input("데이터를 합칠 excel파일 :")      text\_data = analyst\_read\_text(dirname)      df\_analysts = analyst\_find(text\_data)      df\_analysts = analyst\_email\_find(text\_data,df\_analysts)      # 데이터프레임을 하나로 합친다      df\_analyst\_pdf = pd.merge(text\_data,df\_analysts,left\_index=True,right\_index=True)      # 전처리 되지 않은 text는 저장할 필요가 없어서 제거      del df\_analyst\_pdf['text']      excel = pd.read\_excel(excel\_path)      # pdf명을 기준으로 merge      data = pd.merge(excel,df\_analyst\_pdf)      # 엑셀저장      writer = pd.ExcelWriter(os.path.splitext(excel\_path)[0]+"\_analyst.xlsx")      data.to\_excel(writer,'Sheet1')      writer.save()    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

# 긍부정 사전 만들기(make\_dict.py)

이 함수는 딱 한번만 실행하고 반복 실행할 필요가 없음(불용어를 수정해야될때만 확인)

Json은 python의 dictionary를 형태 그대로 저장할 수 있기 때문에 사용함

1. 긍부정 txt파일을 읽고 사전을 만들어 반환

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | tagger = Twitter() # Twitter 태깅 함수  # 불용어 리스트 (본인의 판단에 따라 더 추가 가능) : 텍스트 분석에 필요가 없거나 사업보고서 특성상 당연히 많이 나와 분석에 의미 없는 단어들  stop\_words = ['사항','제기','및','년','사업','관','그','등','것','및','부','수','위','나','대하','개월','원']    def read\_text(text,flag,pos\_neg\_dict):      # 파일 읽고 dictionary로 만듦      f = open(text, 'r', encoding = 'UTF-8')      dict\_text = f.read()      f.close()        dict\_text = dict\_text.split('\n')      dict\_text[0] = re.sub("\ufeff",'',dict\_text[0])        words = [tagger.nouns(doc) for doc in dict\_text if len(tagger.nouns(doc))!=0]        for word in words:          for w in word:              w = re.findall(r'[가-힣]{2,10}', w)              for p in w:                  if p is None: continue                  if p not in stop\_words:                      pos\_neg\_dict[p] = flag      return pos\_neg\_dict    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. 함수호출 및 json 저장

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | if \_\_name\_\_ == "\_\_main\_\_":      print("pos\_pol\_word, neg\_pol\_word 파일명이어야합니다.")      textpath = input("textfile위치(n - 현재폴더) : ")        if textpath == 'n':          textpath = './'        pos\_neg\_dict = {}      pos\_neg\_dict = read\_text(os.path.join(textpath,"pos\_pol\_word.txt"),'pos',pos\_neg\_dict)      pos\_neg\_dict = read\_text(os.path.join(textpath,"neg\_pol\_word.txt"),'neg',pos\_neg\_dict)        json = json.dumps(pos\_neg\_dict)      f = open("dict.json","w")      f.write(json)      f.close()        print("dictionary가 json으로 저장되었습니다")  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

# 텍스트 전처리 및 단어수,긍정 부정단어 수 세기(text\_pos\_neg.py)

1. 단어 수 세기

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | ## 단어 수 세기  # - textdata : excel에 있는 text데이터  # - dir\_path : pdf를 저장할 경로  def count\_word(textdata):      print("----------------------")      print("단어 수 세는 중")      process\_text = [None for \_ in range(len(textdata))]        for idx in range(len(textdata)):          # 읽지 못한 pdf파일은 건너뛴다          if type(textdata[idx]) == str:              sent\_text = textdata[idx].lower()              sent\_text = sent\_tokenize(sent\_text)              sent\_text = [re.sub("\n",'',text) for text in sent\_text]                # 숫자 및 특수문자 제거              #sent\_text = [re.sub("[^(가-힣ㄱ-ㅎㅏ-ㅣa-zA-Z\ )]",' ',text) for text in sent\_text]              # 한글빼고 다 제거              sent\_text = [re.sub("[^(가-힣)]",' ',text) for text in sent\_text]              sent\_text = [re.sub("[(]",' ',text) for text in sent\_text]              sent\_text = [re.sub("[)]",' ',text) for text in sent\_text]              sent\_text = [re.sub("\ +",' ',text) for text in sent\_text]              # 공백이 2개 이상일때              sent\_text = [re.sub("  +",' ',text) for text in sent\_text]                process\_text[idx] = sent\_text        word\_cnt = []      for idx in range(len(process\_text)):          tmp = process\_text[idx]          if tmp is not None:              text\_token = [word\_tokenize(text) for text in tmp]              text\_token = [y for x in text\_token for y in x]              word\_cnt.append(len(text\_token))          else:              word\_cnt.append(0)        return process\_text,word\_cnt      ## word\_cnt는 list -> return 받으면 dataframe으로 저장해야된다  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. 긍부정 단어 수 세기

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | ## 긍부정 단어 수  # - textdata : 전처리한 text데이터  # - pn\_dict : 긍부정 사전  def count\_pos\_neg(textdata,pn\_dict=None):        # 긍부정 사전 open      if pn\_dict is None:          pn\_dict = './dict.json'        json1\_file = open(pn\_dict)      json1\_str = json1\_file.read()      pos\_neg\_dict = json.loads(json1\_str)        # text데이터에서 명사 추출(오래걸림)      start = datetime.datetime.now()      text\_tag = []        for sent\_text in process\_text:          text\_token = []            if sent\_text is None:              text\_tag.append(text\_token)              continue          # 문장단위 토큰화에서 명사 추출          for text in sent\_text:              # 여기서 비교의 용이를 위해 2음절 이상의 단어만 가져옴              if len(text) >= 2:                  text\_token += tagger.nouns(text)            text\_tag.append(text\_token)          progressBar(len(text\_tag),len(process\_text))        end = datetime.datetime.now()      print("\n걸린시간 : ",end-start)        # 긍부정 단어 세기      pos\_cnt = []      neg\_cnt = []        for text in text\_tag:          pos,neg = 0,0          for word in text:              if word in pos\_neg\_dict:                  np = pos\_neg\_dict.get(word)                  if np == "pos":                      pos += 1                  elif np == "neg":                      neg += 1          pos\_cnt.append(pos)          neg\_cnt.append(neg)        return pos\_cnt, neg\_cnt | [cs](http://colorscripter.com/info#e) |

1. 함수실행 및 엑셀 저장

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | if \_\_name\_\_ == "\_\_main\_\_":      excelfile = input("excel파일명 : ")      data\_flag = input("dart? naver? (d/n) :")      pn\_dict = input("긍부정사전( n - ./dict.json): ")      ## excel파일 열기      excel\_data = pd.read\_excel(excelfile)        if pn\_dict == 'n':          pn\_dict = None        if data\_flag == "d":          process\_text,word\_cnt = count\_word(excel\_data["텍스트"])      else:          process\_text,word\_cnt = count\_word(excel\_data["text"])        pos,neg = count\_pos\_neg(process\_text,pn\_dict)        # 데이터 합치기      df = DataFrame(word\_cnt)      pos\_df = DataFrame(pos)      neg\_df = DataFrame(neg)        data2 = pd.concat([df, pos\_df, neg\_df], axis=1)      data2.columns = ["단어수","긍정단어수","부정단어수"]        data = pd.merge(excel\_data,data2,right\_index=True,left\_index=True)        writer = pd.ExcelWriter(os.path.splitext(excelfile)[0]+'\_wordcnt.xlsx')      data.to\_excel(writer,'Sheet1')      writer.save()        print("파일이 저장되었습니다")  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |