

Presented by: KYC

# 킹땡땡과 IDLE

---

고원희 / 심재용 / 유영창 / 임진하



# Outline



## Preparing Corpus

뉴스 크롤링/ 채권보고서 크롤링/ 의사록 크롤링



## Pre-processing

데이터프레임 형식으로 데이터 준비  
(columns = ['date', 'tokens', 'Ngrams', 'labeling'])



## Feature Selection

Token 열 불용어 처리/ 15개 미만 빈도 Ngrams 삭제



## Polarity Classification

Polarity Score 구하기 (positive/ negative 리스트로 분류)



## Sentiment Measurement

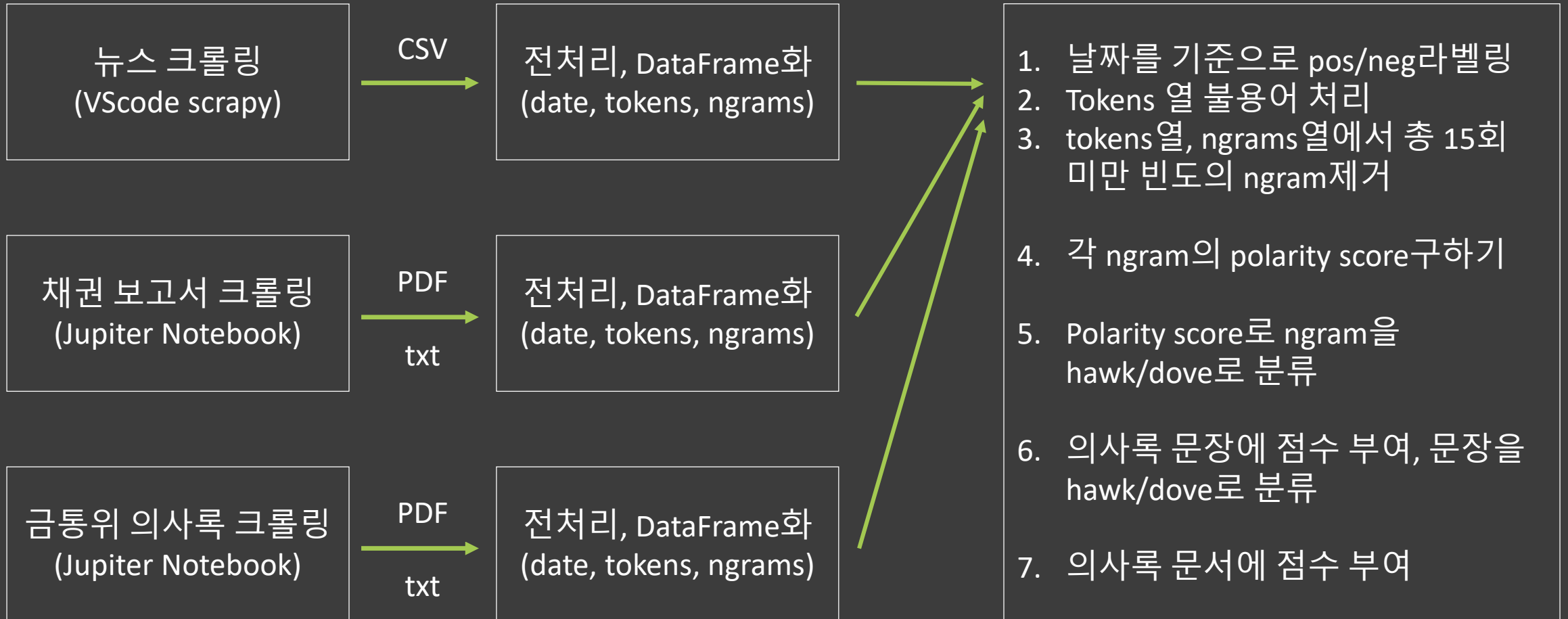
의사록 문장 별 점수 부여 / 의사록 점수 부여



## Result

의사록 점수와 기준금리와의 동향 비교 (matplotlib)

# Outline



킹땡땡과 IDLE

# 크롤링 이슈

# 뉴스 크롤링 이슈

이슈

원하는 언론사 3사만을 필터링



해결 방안

scrapy request 시 쿠키를 포함하여 request,  
연합뉴스, 연합뉴스, 이데일리의 결과만을  
response받아서 처리

Name	Value	Domain	Path	Expires / Max-Age
ASID	afd3a16600000166531b068e0000004a	.naver.com	/	2069-07-06T05:38:...
BMR	s=1564032819491&r=https%3A%2F%2Fpost.naver.com%2Fviewer%2Fpost...	.naver.com	/	Session
NFS	1	.naver.com	/	2028-12-09T09:35:...
NID_AUT	KpJ3YvKoC+tdUk76j7UAJ7alqSECJbYw3XeRmJVrNX/cqRuNtk+EPj2nrXZ/xw7	.naver.com	/	2021-06-11T00:36:...
NID_JKL	kwS3amjs5/9vPXlrIvIkQqm/zClKp5baHz/Xv7xV1vc=	.naver.com	/	2019-08-11T00:36:...
NID_SES	AAABuJT25k0yH7gLn2SyG9ov7bidz44XmX8AhUdil+rgvEsxqhe24zUB5iuAnw...	.naver.com	/	2019-08-24T11:04:...
NNB	4AQ4GIKAZCOFW	.naver.com	/	2050-01-01T09:00:...
_ga	GA1.2.742364616.1541749392	.naver.com	/	2021-03-09T04:45:...
_naver_usersession_	sesvZXsgn54BOVEkifKB/a/N	.naver.com	/	2019-07-25T11:26:...
news_office_checked	1001,1018,2227	.search.naver.com	/	Session
nx_open_so	1	.search.naver.com	/	2068-09-15T09:02:...
nx_ssl	2	.naver.com	/	2019-08-13T03:21:...
page_uid	UgELvdp0JywssaXGUbkSsssss5w-163426	.naver.com	/	Session

```
def start_requests(self):
    for url in self.start_urls:
        yield scrapy.Request(url=url,
                               #연합뉴스, 연합뉴스, 이데일리를 지정하는 쿠키
                               cookies={'news_office_checked': '1001,1018,2227'},
                               callback=self.parse)
```

# 뉴스 크롤링 이슈

## 이슈

뉴스사마다 다른 형식의 웹페이지 사용



## 연관 이슈

1. 오래된 뉴스들은 전부 네이버 뉴스 형식을 따른다.

2. 이데일리는 특정 연도에서 redirecting 에러가 있다.

3. 연합뉴스는 삭제된 link가 있다.

```
# news office에 따라 yield를 지정
# 연합뉴스
if i.css('dd.txt_inline a::attr(href)').get()=='#':
    yield response.follow(href, self.yhif_news)
# 네이버뉴스에서의 이데일리, 연합뉴스
else:
    naver_href = i.css('dd.txt_inline a::attr(href)').get()
    yield response.follow(naver_href, self.naver_news)
```

## 연관 이슈 1 해결 방안

→ 네이버 형식이 없는 연합뉴스를 제외한 이데일리, 연합뉴스는 네이버 뉴스 형식으로만 크롤링

# 뉴스 크롤링 이슈

## 이슈

뉴스사마다 다른 형식의 웹페이지 사용



## 연관 이슈

1. 오래된 뉴스들은 전부 네이버 뉴스 형식을 따른다.
2. 이데일리는 특정 연도에서 **redirecting** 에러가 있다.
3. 연합뉴스포맥스는 삭제된 link가 있다.



## 연관 이슈 2 해결 방안

1. Href에 있는 뉴스 id만 추출, 이데일리 홈 url에 붙여 크롤링
2. 네이버 뉴스 형식으로 크롤링  
(연관 이슈1과 맞물려, 본 팀은 이 방법을 사용하여 진행하였음)

```
def parse(self, response):
```

```
    for item in response.css('ul.type01 li'):
```

```
        url = item.css('a::attr(href)').get()
```

```
        if 'yna' in url:
```

```
            yield response.follow(url, self.parse_yna)
```

```
        if 'curType=read' in url:
```

```
            query_str = parse.parse_qs(parse.urlsplit(url).query)
```

```
            url = 'https://www.edaily.co.kr/news/read?newsId='+query_str['newsid'][0]
```

```
            yield response.follow(url, self.parse_edaily)
```

```
        if 'edaily' in url:
```

```
            yield response.follow(url, self.parse_edaily)
```

# 뉴스 크롤링 이슈

## 이슈

뉴스사마다 다른 형식의 웹페이지 사용



## 연관 이슈

1. 오래된 뉴스들은 전부 네이버 뉴스 형식을 따른다.
2. 이데일리는 특정 연도에서 redirecting 에러가 있다.
3. 연합인포맥스는 삭제된 link가 있다.

```
#delete NaN column, NaN row
frame['date'] = pd.to_numeric(frame['date'], errors='coerce')
frame = frame.dropna()
frame['date'] = frame['date'].astype(int)
dateList = list(frame['date'])
```

## 연관 이슈 3 해결 방안

→ ngram, token을 구할 때 text가 “연합인포맥스 에러”인 column은 다른 date, title, href가 NaN이기 때문에 date이 NaN이면 drop을 해서 걸러냄



# 채권보고서 크롤링 이슈

이슈1

다운로드



해결 방안

pdf 링크를 통해 바로 읽어오는 방식을  
이용해 다운로드

제한적 강세 전망

저물가를 우려하는 각국 중앙은행의 움직임

주사위는 던져졌다

적기에 실행된 금리인하, 당분간 추가 인하는 어려울 ..

설마 했던 한은의 선제적 금리인하

하나금융투자



키움증권



하나금융투자



미래에셋대우



이베스트증권



```
1 def download_file(download_url,name,place):  
2     response = urlopen(download_url)  
3     file = open(place + "{}.pdf".format(name), 'wb')  
4     file.write(response.read())  
5     file.close()
```

# 채권보고서 크롤링 이슈

## 이슈2

채권보고서 제목의 특수문자를  
파일이름에 쓰지 못해 에러 발생



## 해결 방안


처음에 replace를 반복해 사용했으나,  
정규표현식을 이용해 간단하게 정리.

```
def cleanText(readData):  
    #텍스트에 포함되어 있는 특수 문자 제거  
    text = re.sub('[ -+=, #/₩?:^$.@*₩" ※~&%·! 「」 ₩₩ '₩(₩)₩[₩]₩<₩>`₩'...> ]', '', readData)  
    return text
```

```
title = soup.find('table', class_='type_1').find_all('tr')[i+2].find('td').find('a').text  
title = cleanText(title)
```

```
pdf_href = soup.select('div.box_type_m table.type_1')[0].find_all('tr')[i+2].find('td', c  
    try:  
download_file(pdf_href, date+'_'+corp+'_'+title, pdf_folder)  
    except:  
        print(date+'_'+corp+'_'+title)  
        error_list.append(date+'_'+corp+'_'+title)  
        pass
```

# 채권보고서 크롤링 이슈

그 외 이슈	해결 방안
<ul style="list-style-type: none"><li>1. 10.02.18 채권보고서는 다운로드가 되지 않는 문제가 발생.</li><li>2. pdf를 txt파일로 저장 시 encoding문제를 해결해야 했음.</li><li>3. tika에서 에러</li></ul>	 <ul style="list-style-type: none"><li>1. 처음 try,except를 사용할 때 와일드카드를 사용해야 했음.(pdfminer가 느리기 때문)</li><li>2. with open()내에서 encoding='utf8'을 매번 적어야 함.</li><li>3. tika에서 에러가 난 리스트를 반환해 pdfminer로 다시 돌리는 과정을 짰음.</li></ul>

# 의사록 크롤링 이슈

이슈1

클릭 시 바로 다운



해결 방안

html에서 링크를 가져와 도메인에  
연결하는 방법을 이용해 pdf를 다운로드함.

```
base_pdf = 'http://www.bok.or.kr'

for title, date, link in names_urls:
    name = title+'_'+date
    url = base_pdf + link
    downloadURLResource(url, name, pdf_folder)
    print(name, '저장 완료')
```

# 의사록 크롤링 이슈

## 이슈2

pdf와 hwp의 순서가 항상 달라지며,  
2005년 하반기부터 pdf형식을 지원



## 해결 방안

순서를 고정해서 링크를 가져오면  
에러발생.  
따라서 pdf 형식일 때만 가져오고,  
hwp가 등장하는 순간 코드를 브레이크

```
# 더 이상 pdf형식을 지원하지 않을 경우 탈출
if table.find_all('li')[i+3].find('div', class_='fileGoupBox') == None:
    print('0페이지 0번째 파일부터 pdf형식을 제공하지 않습니다.'.format(page, i))
    stop_signal = 1
    break

pdf_or_hwp = table.find_all('li')[i+3].find('div', class_='fileGoupBox').find('ul')
if pdf_or_hwp.find_all('li')[0].find('a').text.split('.')[1][:3] == 'pdf':
    link = pdf_or_hwp.find_all('li')[0].find('a')['href']
else:
    link = pdf_or_hwp.find_all('li')[1].find('a')['href']

title_list.append(title)
date_list.append(date)
link_list.append(link)
```

```
sections = ['Economic Situation', 'Foreign Currency', 'Financial Markets',
            'Monetary Policy', 'Participants' Views', 'Government' s View']
section_texts = (section2, section3)

return section_texts
```

킹땡땡과 IDLE

# 전처리 이슈

# 뉴스 전처리 이슈

이슈

## 뉴스 corpus에서 무의미한 text 삭제



## 해결 방안

1. 정규표현식으로 (~기자, ~특파원, ~부, (끝))등 내용과 관계없는 부분 제거

(ngram의 퀄리티가 더 좋아지는지 실험했으나, 큰 차이는 없었음.)

2. 에러 발생 시 yield로 "~뉴스에러"를 저장해, 에러가 발생한 행들을 Drop함. Drop한 행들은 약 2355건 정도이며, 결과적으로 244,495건의 뉴스를 전처리함.

```
import re
```

```
def doFiltering(inputStr):
```

```
filtering = re.sub(r"(\.? 끝 .?\)|\
```

(.?안방에서 만나는 가장.\*)|\

(\(.??.?연합뉴스.?\\)(.?(기자|특파원).\*=.?)?|\\

(\.[0,5}(edaily|이데일리).[0,20}(특파원|기자|증권부|채권외환팀|국제부|.?.럼니스트|경제부장|보도제작부|시장부

```
(.{0,4}(기자|. {0,5}?특파원).?()).{0,10}@\\)|\\
```

(\((\{0,2\}(\text{각 기관별|전문가별|이 인터뷰는|이 기사는|더 자세한|보다}).+)?이데일리 유료.+다.\{0,2\}\))\|

(당사의 기사를 사전 동의.\*)|\

(자신만만 재테크.+)\

```
(<?저작권자.*)",',str(inputStr))-
```

```
chkStr = re.compile(r'(<span class\=.*>')
```

```
if chkStr.search(filtering):
```

```
filtering = filtering.split(chkStr.search(filtering).group())[0].strip()
```

```
return filtering
```

# 의사록 전처리 이슈

이슈

의사록의 섹션 분리(섹션2와 섹션3만 사용)



해결 방안

섹션분리 및 각 섹션을 반환하는 함수를 수정. 섹션 분리의 기준이 되는 상투적 문장을 찾는 정규표현식은 시간문제 상 건드리지 않고 그대로 사용함.

292개의 05년도~19년도 의사록들 중 섹션이 분리되어 2,3 섹션에 내용이 존재하는 의사록은 총 147건이었음.

```
# 외환.국제금융 동향
bos = s2
eos = s3 if s3 >= 0 else s4
section = minutes[bos:eos] if bos >= 0 or eos >= 0 else ''
pos = re.search(r'(일부|대부분의) 위원들은', section, re.MULTILINE)
bos = pos.start() if pos else -1
section = section[bos:] if bos >= 0 else section
section2, section2_txt = tidy_sentences(section)
#print(section)
```

```
# 금융시장 동향
bos = s3
eos = s4
section = minutes[bos:eos] if bos >= 0 or eos >= 0 else ''
pos = re.search(r'(일부|대부분의) 위원들은', section, re.MULTILINE)
bos = pos.start() if pos else -1
section = section[bos:] if bos >= 0 else section
section3, section3_txt = tidy_sentences(section)
```

```
# 외환.국제금융 동향
pos = re.search(
    r'(.?외환.?국제금융?s?동향.?과 관련하여,.*\#\#\#) 외환.국제금융?s?(및 금융시장)?\s?동향)\n?\s*(일부 위원은|대부분의 위원들은)',
    minutes,
    re.MULTILINE)
s2 = pos.start() if pos else -1

# 금융시장 동향
pos = re.search(
    r'(.?금융시장?s?동향.?과 관련하여,.*\#\#\#) 금융시장?s?(및 일부 위원들은)',
    minutes,
    re.MULTILINE)
s3 = pos.start() if pos else -1
```



# 의사록 전처리 이슈

이슈

tidy\_sentences 함수 수정



해결 방안

tidy\_sentences 함수에서 사용한 정규표현식이 문장분리를 잘 못하기 때문에 수정

수정하다가 한계를 느껴 pdf를 txt로 바꾸는 과정에서 '\n'을 "로 처리하지 않고 ' '로 바꾸도록 수정. 문장 분리 성공.

```
#주로 에러가 발생하는 코드
try:
    parsedPDF = re.sub('\n', ' ', parser.from_file(pdf_tmp_filepath)["content"])
    #문!*****
except:
    print(pdf)
    error_list.append(pdf)
    pass
```

```
def tidy_sentences(section):
    #sentence_enders = re.compile(r"((?<=[함음됨임봄짐움])(\s*\n|\s|,|:|(?<=[\s])\s)|\s*)") <- 원본
    sentence_enders = re.compile(r"((?<=[함음됨임봄짐움])(\s+|\s|,|:|(?<=[\s])\s)|\s*\s*(+))")
    splits = list((m.start(), m.end()) for m in re.finditer(sentence_enders, section))
    starts = [0] + [i[1] for i in splits]
    ends = [i[0] for i in splits]
    sentences = [section[start:end] for start, end in zip(starts[:-1], ends)]
    for i, s in enumerate(sentences):
        sentences[i] = (s.replace('\n', ' ').replace(' ', ' ')) + ' ' #두번째 replace된지???
    #쓰지도 않는 부분
    text = '\n'.join(sentences) if len(sentences) > 0 else ''

    return sentences, text
```

# 의사록 전처리 이슈

이슈

문장별 스코어 부여 준비



해결 방안

섹션이 분리되는 모든 문서에서 섹션2,3에 포함되는 문장들 갯수만큼의 행을 갖는 데이터프레임 생성.

```
path = output_folder
columns = ['date', 'title', 'content', 'tokens', 'ngrams']
data = []

for title in os.listdir(path)[1:]:
    date = title.split("(")[1].split('_')[0]
    name = title.split("_")[0]
    file = path + title
    with open(file, 'r', encoding = 'utf8') as f:
        content = f.read()
        content_tgsections = [y for x in preprocess_minutes(content) for y in x]
        for sentence in content_tgsections:
            tokens = text2tokens(sentence)
            ngrams = text2ngram(sentence)
            #print(content)
            data.append([date, name, sentence, tokens, ngrams])

print(title, '추가 완료')
```

```
df = pd.DataFrame(data, columns = columns)
```

킹땡땡과 IDLE

# 결과 산출

---

# 결과 산출 단계 이슈

이슈1

불용어 함수의 작업 속도 개선



해결 방안

불용어 함수로 통합 데이터프레임의 tokens열을 전부 불러와서 한번에 처리함. 각 셀의 데이터가 리스트가 아니라 str인 점을 이용해 정규표현식의 re.sub을 이용해 불용어 토큰들을 지우는 데에 성공함.

```
def stopPos(df):
    for idx in range(len(df['tokens'])):
        if df['tokens'][idx] == '[]':
            continue
        else:
            df['tokens'][idx] = re.sub(r'#[S+]/(SC|SY|SF|SE|SS|SP|SO|SW|SSC|
|JKS|JKC|JKG|JKO|JKB|JKV|JKQ|JX|JC|
|EF|EC|ETN|ETM)#[,]?#s?', '', df['tokens'][idx])

    if idx%1000==0:
        print(idx)
    return df
```

# 결과 산출 단계 이슈

## 이슈2

Feature selection 함수 속도 개선



## 해결 방안

15회 미만 출현빈도 ngram들을 삭제할 때 value\_counts 데이터 프레임의 인덱싱을 활용.

```
def selectFeat(df):
    allTokens=[]
    allNgrams=[]

    num=0
    for item in df['tokens']:
        li = eval(item)
        allTokens.extend(li)
        num+=1
        if num%1000==0:
            print('tokens {}번째까지 병합완료'.format(num))
    tokens15 = list(pd.Series(allTokens).value_counts()[pd.Series(allTokens).value_counts() >= 15].index)
    print('tokens에서 15회 이상 엔그램 추출완료')

    num=0
    for item in df['ngrams']:
        li = eval(item)
        allNgrams.extend(li)
        num+=1
        if num%1000==0:
            print('ngrams {}번째까지 병합완료'.format(num))
    ngrams15 = list(pd.Series(allNgrams).value_counts()[pd.Series(allNgrams).value_counts() >= 15].index)
    print('ngrams에서 15회 이상 엔그램 추출완료')

    return tokens15+ngrams15
```

# 결과 산출 단계 이슈

## 이슈3

Polarity score 계산 함수 속도 개선



## 해결 방안

처음에 pos및 neg라벨링이 붙은 데이터프레임의 모든 ngram들을 불러와 한 리스트에 저장 후, polarity score를 계산하고 싶은 ngram을 해당 리스트에서 카운트하는 방식을 이용했으나 너무 느림.  
pos\_ngram리스트와 neg\_ngram리스트를 각 시리즈로 만들어 value\_counts에서 인덱싱하는 방법으로 속도 개선.

```
def polarScore(df, lst):
    posDf = df[df['label']=='pos']
    negDf = df[df['label']=='neg']
    print('라벨에 따른 의사록 데이터프레임 분류완료')
```

```
# polarity score 산출
pos_list=[]
neg_list=[]
```

```
allPos_valuecounts = pd.Series(allPos).value_counts()
allNeg_valuecounts = pd.Series(allNeg).value_counts()
```

```
print('중간
```

```
i=0
for ngram in lst:

    if ngram not in allNeg_valuecounts.index and ngram not in allPos_valuecounts.index:
        score = 1
    elif ngram not in allNeg_valuecounts.index:
        score = (allPos_valuecounts[ngram]/pos_cnt) / (1/neg_cnt)
    elif ngram not in allPos_valuecounts.index:
        score = 0
    else:
        score = (allPos_valuecounts[ngram]/pos_cnt) / (allNeg_valuecounts[ngram]/neg_cnt)

    if score >= 1.3:
        pos_list.append(ngram)
    elif score <= (1/1.3):
        neg_list.append(ngram)

    i+=1
    if i%1000==0:
        print('0번째 진행완료 / 총 0개'.format(i, len(lst)))

return pos_list, neg_list
```

# 결과 산출 단계 이슈

이슈4

시각화

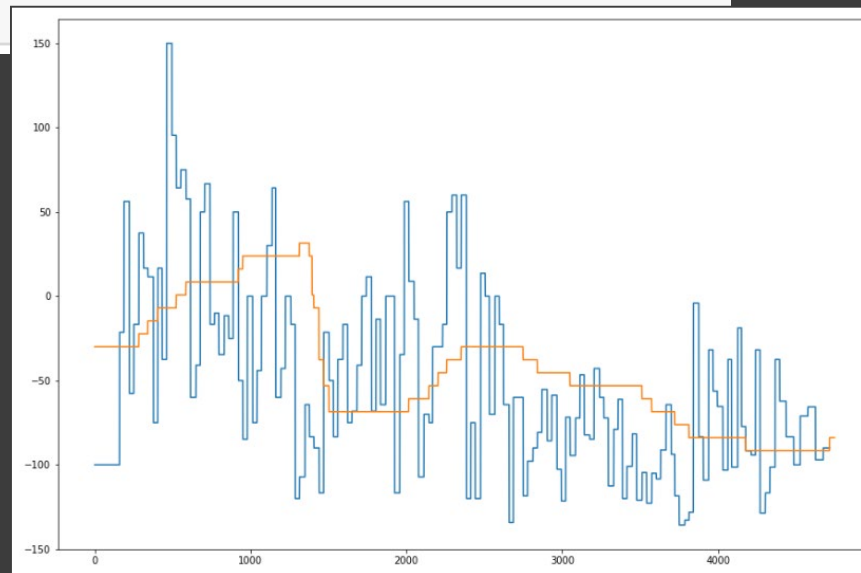


해결 방안

의사록 문장과 의사록 문서 스코어 계산 시에는 논문에 나와있는 대로 수식을 이용

기준금리와 비교시각화에서는 스케일링을 통해 한 그래프에 나타냈음.

```
plt.figure(figsize=(15,10))
plt.plot(result2['톤 스코어']/result2['톤 스코어'][0]*(-100))
plt.plot(base_rate['baseRate']/base_rate['baseRate'][0]*100-130)
# plt.legend(['톤', '기준금리'])
# plt.ylim(-2,2)
```

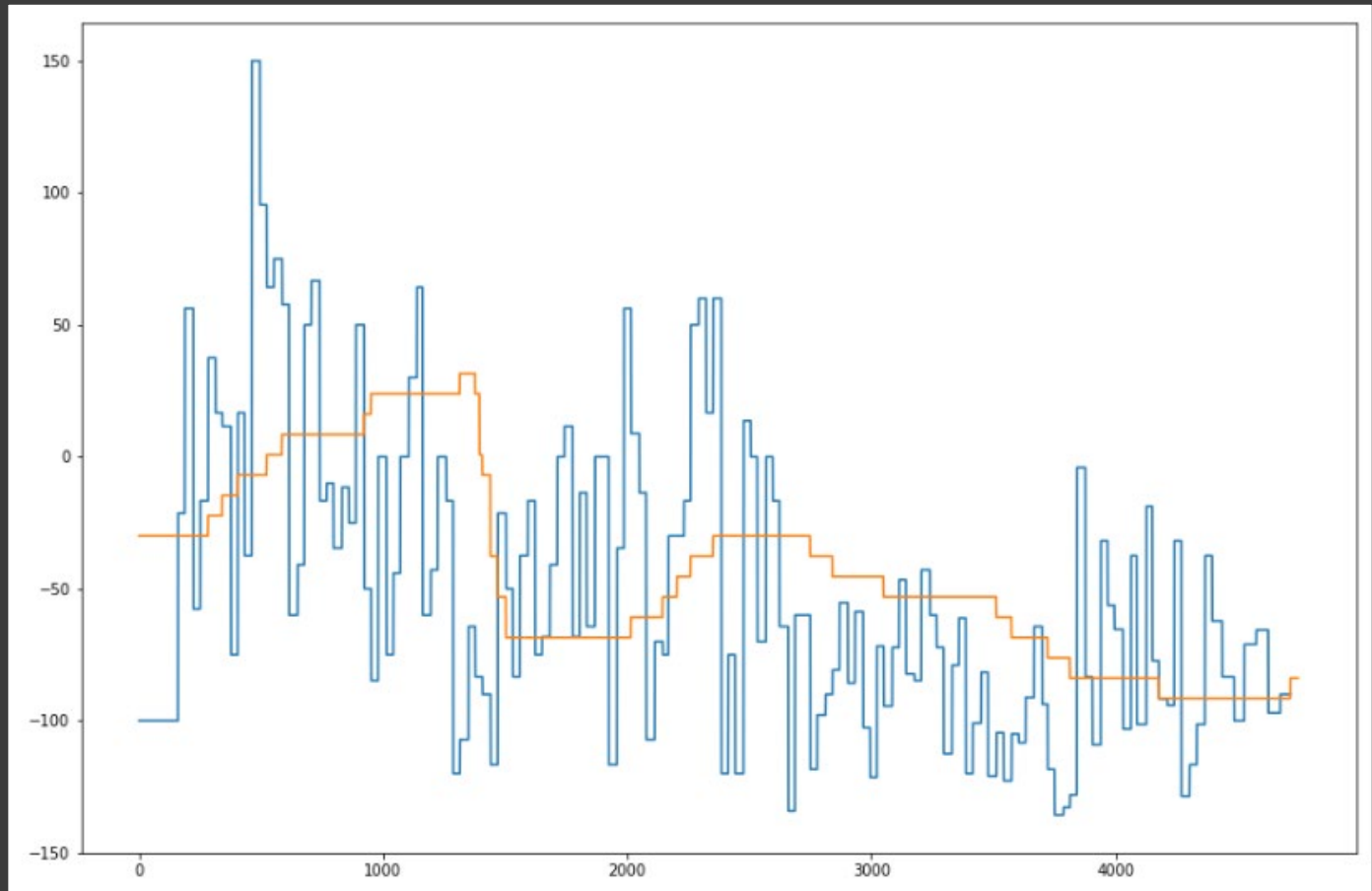


킹땡땡과 IDLE

# Result & 아쉬웠던 점

```
df.corr('pearson')
```

	톤스코어	baseRate
톤스코어	1.000000	0.415041
baseRate	0.415041	1.000000





# 우리가 한 일은요



고원희

“

- 뉴스 크롤링
- 뉴스 전처리
- 콜 금리 크롤링
- 콜 금리 전처리
- 의사록, 채권보고서, 뉴스의 data 통합 후 label 붙이기
- 함수화를 포함한 마무리 작업
  - Ppt 제작



심재용

“

- 뉴스 크롤링
- 뉴스 전처리
- 기준 금리 크롤링
- 기준 금리 전처리
- 뉴스 Ngram 추출



유영창

“

- 의사록 크롤링
- 의사록 전처리
- 의사록 ngram, tokens 추출
- 채권보고서 크롤링, 전처리
- 채권보고서 ngram, tokens 추출
- 콜금리 전처리
- 의사록, 채권보고서, 뉴스의 data 통합 후 label 붙이기
- 함수화를 포함한 마무리 작업, 기준금리와 비교 시각화



임진하

“

- 뉴스 크롤링 도움
- 뉴스 전처리 도움
- 의사록 크롤링
- 채권보고서 크롤링
- 뉴스 tokens 추출

킹땡땡과 IDLE

# 우리가 사용한 인프라는요

Branch: master ▼ KYCwithIDLE / BOK 논문 리뷰 / 4\_FeatureSelection(n-grams) / Create

ddori5338 190726\_유영창\_기준금리와의 상관계수 산출

..

190726_유영창_기준금리와의 상관계수 산출	190726_유영창_기준금리와의 상관계수 산출
190726_유영창_기준금리와의 상관계수 산출	190726_유영창_기준금리와의 상관계수 산출
190724_유영창_모든데이터병합부터 의사록	190724_유영창_모든데이터병합부터 의사록

폴더 이름 ↑

기준금리	의사록섹션분리,문장분리이슈
콜금리	bond_report_crawling_3459
doctorrock_crawling_292	news crawling
ngram	ppt팜플렛

# Thank you

---



# Q&A

---

