

Term Project

보고서 및 논문 윤리 서약

1. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다.
2. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다.
3. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다.
4. 나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁받지도 않겠습니다.

나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.

학 과 : 융합소프트웨어학부 데이터테크놀로지전공

과 목 : 빅데이터 프로그래밍

담당교수 : 권동섭 교수님

강좌 번호: 5951

학 번 : 60191692

이 름 : 주지현

(서명)

1. 문제 정의

KBO 리그는 대한민국의 프로 야구 리그이며 6개 구단으로 1982년에 처음 시작되었다. 우리나라에서 야구는 2008년 베이징 올림픽 금메달을 따며 '베이징 뉴비'라는 말이 생길 만큼 인기가 크게 늘었고 2017년에는 총 관중 수 8,400,688명이라는 수를 기록하며 현재 국내 모든 스포츠 리그를 통틀어 가장 인기가 많은 스포츠라고 볼 수 있다.

프로야구에 대한 관심이 많고 이목이 집중되는 만큼 팀과 선수들에게 향하는 기대와 더불어 비난의 양도 비례한다. 당일 경기의 결과가 좋지 않다면 가차없이 바로 비판의 기사가 양산되고 팬들의 반응 또한 싸늘하다.

이번 2021년 KBO 정규시즌은 코로나로 인한 리그 중단, 선수의 방역 규칙 위반 음주 사태, 실망스러운 도쿄올림픽 메달 획득 실패, 그리고 총재 비리 등 크고 작은 이슈들이 아주 많았다. 또한 순위 변동도 많았던 시즌이었는데 정규시즌 마지막 경기까지도 1등부터 3등까지의 세 팀의 순위를 알 수 없었고, 4등부터 6등까지 세 팀의 순위도 알 수 없었다.

이렇게 야구 외부적인 이슈와 함께 리그 내적으로도 순위 변동 뿐만 아니라 '양석환'-'함덕주' 트레이드, '서건창'-'정찬헌' 트레이드 등 선수 간의 변화도 많아 시끄러웠던 시즌이었다. 특히 10개 구단 중 두산 베어스는 8월 정규시즌 공동 7등에서 최종 4위까지 올라갔을 만큼 월별 성적 차이가 컸다. 또한 두산 베어스는 화제가 되었던 트레이드의 중심이었으며 코로나와 총재 비리 관련된 이슈에도 관련이 있는 구단이었다.

한 시즌 동안 말이 많았던 팀인만큼 과연 "두산 베어스"라는 키워드가 들어간 기사를 모아 놓고 봤을 때 월별로 기사에 어떤 단어와 표현이 있는지, 어떠한 단어들이 있을 때 성적이 좋았고 혹은 반대로 그렇지 않았는지 궁금해졌다. 따라서 월별로 본문과 내용에 "두산 베어스"라는 키워드가 포함된 기사들을 시즌 개막 이후 월별로 수집하여 특정 구단에 대한 언론의 관심과, 해당 기간 팀의 성적과의 관계를 분석해보고자 본 프로젝트를 시작하게 되었다.

2. 시스템 아키텍처

1) 데이터 수집 - colab, python, csv

데이터 수집은 구글에서 python을 편리하게 사용할 수 있도록 제공하는 서비스인 colab(laboratory(이후 줄여서 colab이라고 하겠다))을 사용하였다. 딥러닝과 같이 GPU를 이용한 병렬 작업이 필요하진 않았지만 로컬 환경이 불안정하여 colab에서 코드를 실행시켰다. 데이터는 네이버 기사를 크롤링하여 수집하였다. 크롤링을 진행할 페이지 수, 검색어, 검색 방식, 날짜를 입력하여 기사가 발행된 날짜와 기사 제목, 신문사, 기사 요약본, 링크 데이터를 가져왔다.

2) 데이터 저장 및 전처리 - csv

크롤링을 진행한 후 가져온 데이터의 포맷은 csv이다. 크롤링으로 가져온 csv 파일을 MS Excel로 열면 한글 인코딩이 깨질 수 있지만 맥북에서 Numbers를 사용하여 csv 파일을 열었기 때문에 추가적인 인코딩 혹은 xlsx 파일을 사용하지 않고 csv만 사용하였다.

크롤링한 데이터를 보면 컬럼이 날짜, 제목, 신문사, 내용, 링크 이렇게 5개로 이루어져 있다. 본 프로젝트에서는 Spark를 통해 word count를 진행하는데 이 때 링크와 신문사 컬럼에 있는 내용은 중복되는 내용도 많고 본 프로젝트에 필요한 부분이 아니기 때문에 미리 컬럼을 삭제하고 날짜, 제목, 내용 3가지의 컬럼만 남겨두었다.

3) 데이터 분석 - spark, python

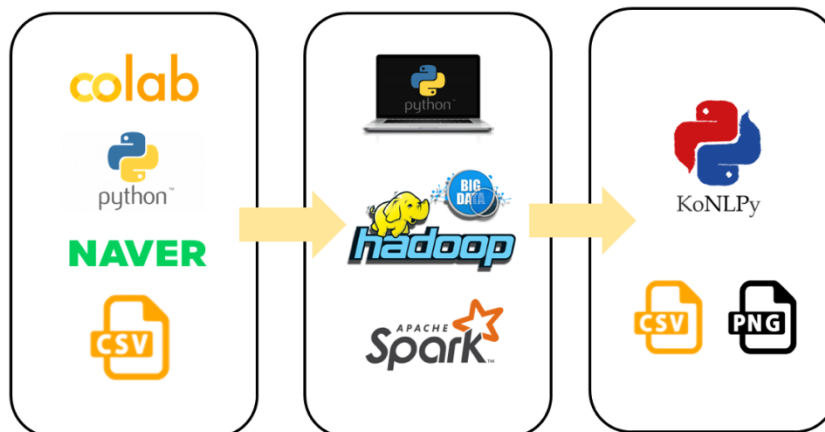
최종적으로 나온 데이터인 csv 파일을 HDFS에 업로드하고 ssh@localhost에 접속해 spark-submit으로 Spark를 통해 분석을 진행한다. Spark를 사용한 이유는 수업을 듣고 과제를 할 때 그냥 Map-Reduce를 사용하는 것보다 Spark를 통해 분석을 진행하는 것이 더 쉽고 속도가 훨씬 빠르게 진행되었기 때문이다. 하둡은 여러 단계를 거쳐 처리가 되는 반면에 Sparks는 클러스터에서 데이터를 읽고, 수행 및 결과값 입력과 같은 모든 과정이 동시에 진행되는 과정이 두 단계로 처리가 되기 때문에 속도면에서 월등히 빠르다는 장점이 있다.

Spark로 분석을 진행할 때 사용할 코드는 python으로 작성하였고, 분석을 통해 나온 결과(word count)는 다시 csv형태로 만들어 output.csv가 생성되게 하였다.

4) 데이터 시각화 - python konlpy

Spark를 통해 word count를 진행하여 나온 결과를 csv에 저장하였는데, csv 파일로 단어를 하나하나 보는 것은 불편하고 똑똑한 방법이 아니기 때문에 시각적으로 한 눈에 볼 수 있게 만들고 싶었다. 따라서 역시 colab에서 python으로 작성한 코드에 konlpy와 wordcloud 라이브러리를 사용하여 데이터 시각화를 하였다.

프로젝트의 전체적인 시스템 흐름은 아래 그림과 같다.



3. 데이터 수집 방법

1) 크롤링 코드

```
[ ] !pip install beautifulsoup
    !pip install urllib
    !pip install pandas
```

```
[ ] # 필요한 라이브러리 import
    from bs4 import BeautifulSoup
    import requests
    import pandas as pd
    import re
```

```
[ ] # 크롤링 결과를 저장할 리스트
    date_list = []
    title_list = []
    press_list = []
    contents_list = []
    link_list = []
    data_dict = {}
```

우선 필요한 라이브러리를 임포트하고, 크롤링을 하며 저장할 내용인 날짜, 기사 제목, 언론사, 기사 내용, 링크를 담은 리스트와 해당 리스트들을 딕셔너리 형태로 만들 변수를 선언해준다.

```
def main():
    maxpage = input("최대 페이지 수: ")
    keyword = input("키워드: ")
    sort = input("관련도순=0 최신순=1 오래된순=2: ")
    start = input("시작 날짜 ex.(2021.04.03):")
    end = input("마지막 날짜 ex.(2021.11.18):")

    crawling(maxpage, keyword, sort, start, end)

main()
```

메인함수에서 크롤링을 진행할 최대 페이지 수, 기사에 포함되어야 할 키워드, 정렬 방법, 시작 날짜, 마지막 날짜를 입력받고 해당 변수들을 인자로 넣어 crawling함수를 호출한다.

```
def crawling(maxpage, keyword, sort, start, end):
    page = 1
    max = (int(maxpage) - 1) * 10 + 1
    d_from = start.replace(".", "")
    d_to = end.replace(".", "")

    while page <= max:
        url = "https://search.naver.com/search.naver?where=news&query=" + keyword + "&sort=" + sort + "&ds=" + start + "&de=" + end + "&nso=s0h3Arh2CpK3Afrom" + d_from + "to" + d_to + "%2Ca3A6start=" + str(page)
        response = requests.get(url)
        html = response.text
        soup = BeautifulSoup(html, 'html.parser')

        # <a> 태그에서 제목과 링크를 추출한다.
        a_tag = soup.select('.news_tit')
        for i in a_tag:
            # 기사 제목 추출
            title_list.append(i.text)
            # 기사 링크 추출
            link_list.append(i['href'])

        # 신문사를 추출한다.
        press_lists = soup.select('.info_group > .press')
        for i in press_lists:
            press_list.append(i.text)

        # 기사 날짜를 추출한다.
        date_lists = soup.select('.info_group > span.info')
        for i in date_lists:
            # n번 n단에서 '번'을 제거하여 숫자만을 추출한다.
            if i.text.find("년") == -1:
                date_list.append(i.text)

        # 기사 요약본을 추출한다.
        contents_lists = soup.select('.news_dsc')
        for i in contents_lists:
            # 기사 요약본을 정제화할 함수를 호출한다.
            contents_cleansing(i)

        # 리스트 형태로 저장된 data, title, press, contents, link를 data_dict에 딕셔너리 형태로 넣어준다.
        data_dict = {"date": date_list, "title": title_list, "press": press_list, "contents": contents_list, "link": link_list }
        print(page)

        # 딕셔너리로 저장된 data_dict를 데이터프레임으로 변환해준다.
        df = pd.DataFrame(data_dict)
        page += 10
        df

    # to_csv를 사용하여 dataframe을 csv으로 생성
    outputFileName = 'doosanbears_11'
    df.to_csv("doosanbears_11.csv")
```

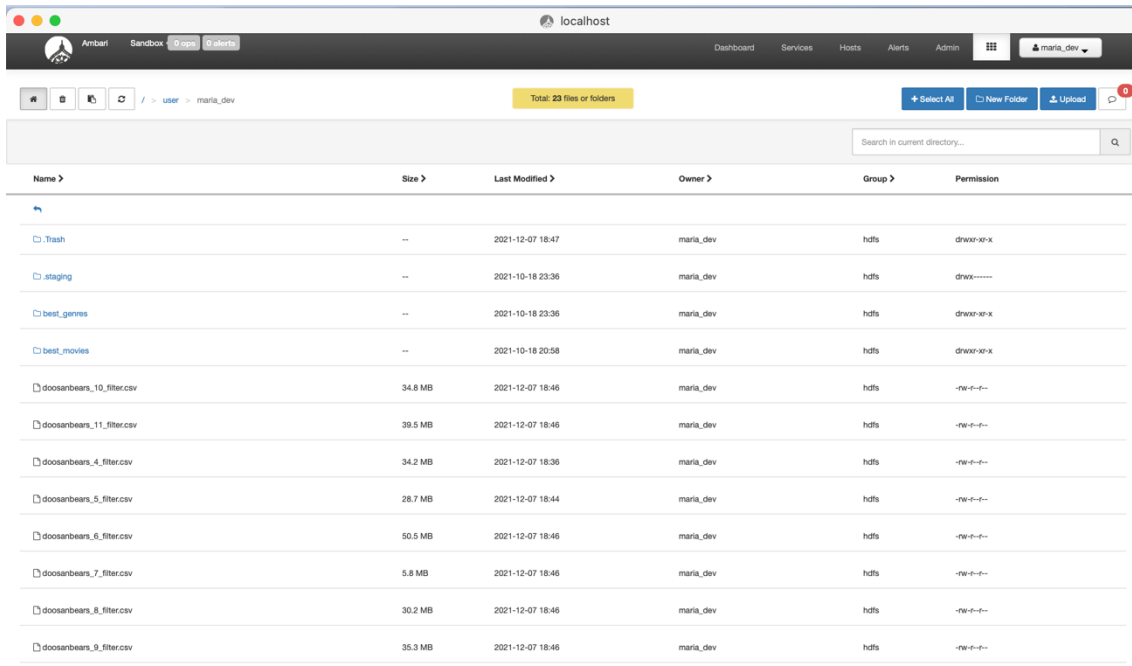
현재 페이지가 입력 받은 최대 페이지보다 작을 때까지 반복하여 기사의 제목, 링크, 신문사, 날짜, 요약본을 추출한다. 기사 요약본은 정제화가 필요하기 때문에 따로 정제화하는 함수를 호출해준다.

위에서 선언한 리스트에 각각의 데이터를 넣어주고 최종적으로 이 데이터들을 data_dict 라는 변수에 딕셔너리 형태로 넣는다. 후에 이 딕셔너리 형태의 데이터를 df를 사용하여 데이터프레임으로 변환한 뒤 최종적으로 csv로 저장한다.

```
# 기사의 내용을 정제화하는 함수
def contents_cleansing(contents):
    # 앞에 있는 내용 중 필요없는 부분을 제거한다
    first_step = re.sub('<dl>.*?</div> </dd> <dd>', '', str(contents)).strip()
    # 뒤에 있는 내용 중 필요없는 부분을 제거한다
    second_step = re.sub('<ul class="relation_lst">.*?</dd>', '', first_step).strip()
    third_step = re.sub('<.+?>', '', second_step).strip()
    # 정제화를 마친 내용을 기사 내용 담는 list에 append해준다.
    contents_list.append(third_step)
```

기사 요약본을 정제화하는 함수이다. Contents 값을 가져와 <dl>부터 <div> </dd> <dd> 까지의 내용을 공백처리하여 필요한 내용만을 추출한다.

2) 수집된 데이터



Name	Size	Last Modified	Owner	Group	Permission
.Trash	--	2021-12-07 18:47	aria_dev	hdfs	drwx-r-x
.staging	--	2021-10-18 23:36	aria_dev	hdfs	drwx-----
.best_genres	--	2021-10-18 23:36	aria_dev	hdfs	drwx-r-x
.best_movies	--	2021-10-18 20:58	aria_dev	hdfs	drwx-r-x
doosanbears_10_filter.csv	34.8 MB	2021-12-07 18:46	aria_dev	hdfs	-rw-r--r--
doosanbears_11_filter.csv	39.6 MB	2021-12-07 18:46	aria_dev	hdfs	-rw-r--r--
doosanbears_4_filter.csv	34.2 MB	2021-12-07 18:36	aria_dev	hdfs	-rw-r--r--
doosanbears_5_filter.csv	28.7 MB	2021-12-07 18:44	aria_dev	hdfs	-rw-r--r--
doosanbears_6_filter.csv	50.6 MB	2021-12-07 18:46	aria_dev	hdfs	-rw-r--r--
doosanbears_7_filter.csv	5.8 MB	2021-12-07 18:46	aria_dev	hdfs	-rw-r--r--
doosanbears_8_filter.csv	30.2 MB	2021-12-07 18:46	aria_dev	hdfs	-rw-r--r--
doosanbears_9_filter.csv	35.3 MB	2021-12-07 18:46	aria_dev	hdfs	-rw-r--r--

수집한 데이터는 Ambari File View에서 /user/aria_dev에 업로드한다.

4. 데이터 분석 방법:

1) Spark code

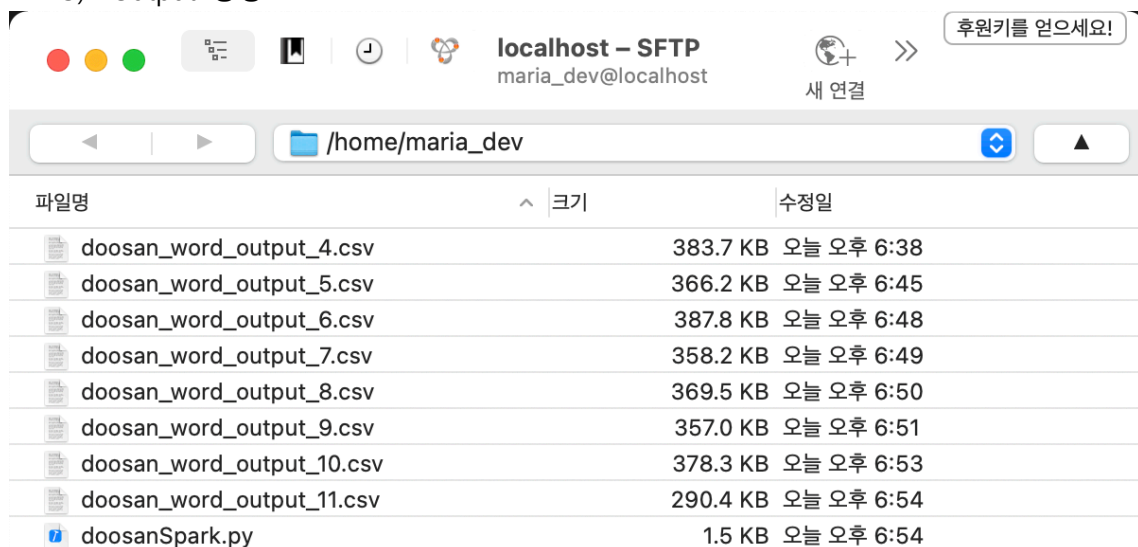
```
doosanSpark.py 2 x
doosanSpark.py > ...
1 import sys
2 import csv
3
4 from pyspark import SparkContext, SparkConf
5
6 # Encoding to UTF-8
7 reload(sys)
8 sys.setdefaultencoding('utf-8')
9
10 # Create Spark Context
11 conf = SparkConf().setAppName("Doosan Bears Word Count")
12 sc = SparkContext(conf = conf)
13
14 # Create RDD and Split words
15 tokens = sc.textFile("hdfs:///user/aria_dev/doosanbears_11_filter.csv").flatMap(lambda line: line.split(" "))
16
17 # Count words
18 word_counts = tokens.map(lambda word: (word, 1)).reduceByKey(lambda v1, v2: v1 + v2)
19
20 # Collect counted words to keywords
21 keywords = word_counts.collect()
22
23 # Save keywords to CSV
24 fp = open("doosan_word_output_11.csv", "w")
25 writer = csv.writer(fp, dialect="excel")
26 writer.writerows(keywords)
27 fp.close()
```

2) 스파크 실행 과정

```
judy — maria_dev@sandbox-hdp:~ — ssh maria_dev@localhost -p 2222 — 96x20
[judy@JudyMacBookPro ~ % ssh maria_dev@localhost -p 2222
maria_dev@localhost's password:
Last login: Mon Dec 6 23:31:26 2021 from 172.18.0.2
[[maria_dev@sandbox-hdp ~]$ spark-submit doosanSpark.py
SPARK_MAJOR_VERSION is set to 2, using Spark2
```

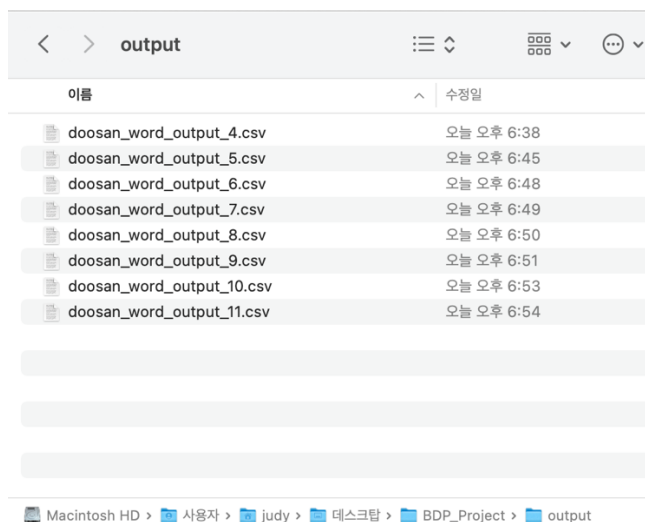
Cyberduck의 /home/maria_dev에 위의 doosanSpark.py를 업로드하고 터미널에서 ssh maria_dev@localhost:2222에 접속하여 해당 파일을 실행시킨다.

3) Output 생성



파일명	크기	수정일
doosan_word_output_4.csv	383.7 KB	오늘 오후 6:38
doosan_word_output_5.csv	366.2 KB	오늘 오후 6:45
doosan_word_output_6.csv	387.8 KB	오늘 오후 6:48
doosan_word_output_7.csv	358.2 KB	오늘 오후 6:49
doosan_word_output_8.csv	369.5 KB	오늘 오후 6:50
doosan_word_output_9.csv	357.0 KB	오늘 오후 6:51
doosan_word_output_10.csv	378.3 KB	오늘 오후 6:53
doosan_word_output_11.csv	290.4 KB	오늘 오후 6:54
doosanSpark.py	1.5 KB	오늘 오후 6:54

doosanSpark.py 코드에서 fp=open()을 사용하여 csv 파일을 생성하였기 때문에 cyberduck을 보면 /home/maria_dev에 output csv들이 잘 생성된 것을 확인할 수 있다.



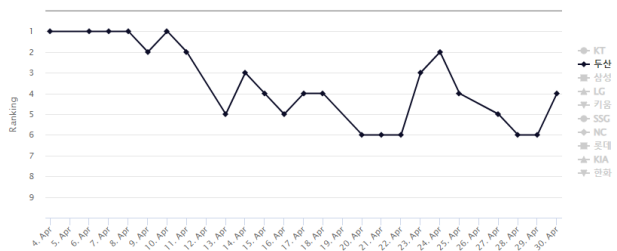
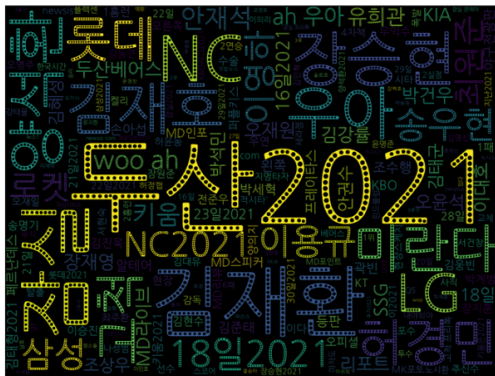
이름	수정일
doosan_word_output_4.csv	오늘 오후 6:38
doosan_word_output_5.csv	오늘 오후 6:45
doosan_word_output_6.csv	오늘 오후 6:48
doosan_word_output_7.csv	오늘 오후 6:49
doosan_word_output_8.csv	오늘 오후 6:50
doosan_word_output_9.csv	오늘 오후 6:51
doosan_word_output_10.csv	오늘 오후 6:53
doosan_word_output_11.csv	오늘 오후 6:54

해당 파일들을 로컬에 다운로드하였다.

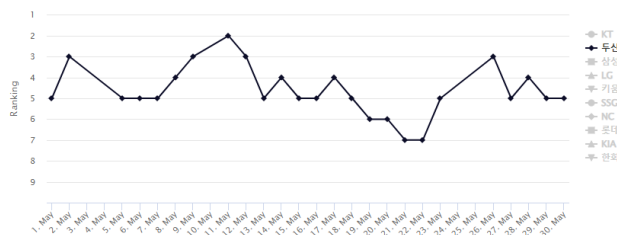
5. 데이터 분석 결과

전체적으로 다른 구단들의 이름과 선수들의 이름이 가장 많이 나왔다. 기사는 정보 전달의 성격을 가지고 있기 때문에 내가 주제를 선정하며 처음에 도출하고자 했던 언론과 대중들의 긍정적 혹은 부정적 반응과 같은 결과를 얻을 수 없었다. 또한 한 개 구단의 기사만 보았기 때문에 비교군이 없어 기사 키워드와 팀의 월별 성적 간의 유의미한 결과를 내지는 못했다.

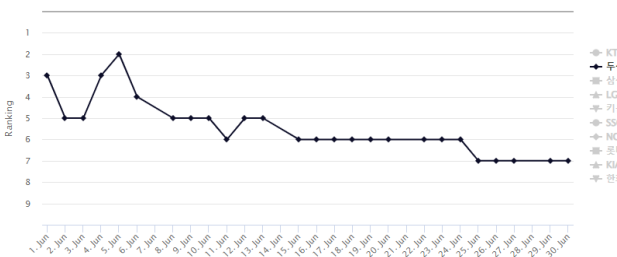
그럼에도 불구하고 코로나로 특히 논란이 되었던 7월의 워크클라우드에는 '코로나'가 있는 것을 확인할 수 있었고, 리그 MVP를 받은 '미란다' 선수가 10월 11월에는 크게 나타난 것을 확인함으로써 데이터의 수집과 분석이 잘 이루어진 것을 알 수 있었다.



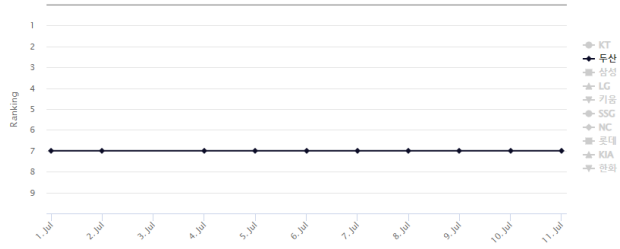
4월



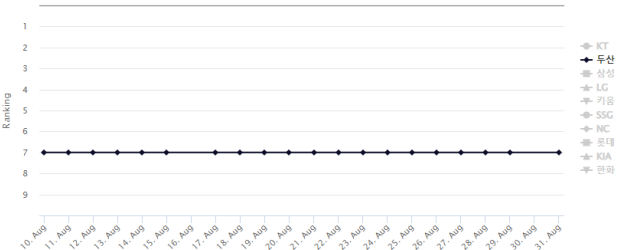
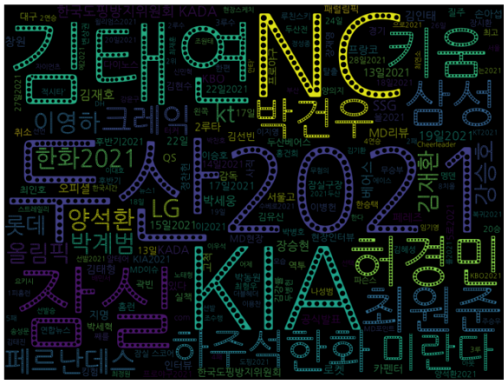
5월



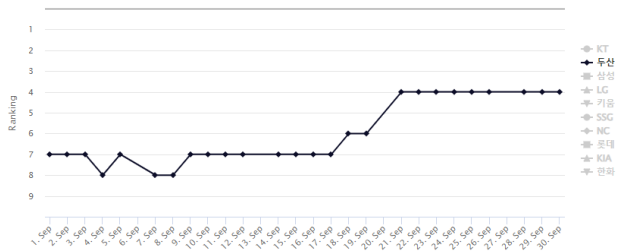
6월



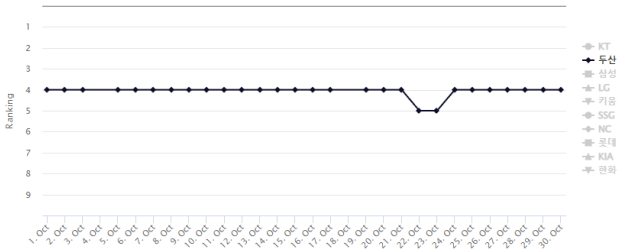
7월



8월



9월



10월



11월 (정규시즌이 끝난 이후이므로 월별 순위 변동 그래프는 없다.)

6. 추가적인 확장 가능성: 추가적인 확장 아이디어나 가치

본 프로젝트는 내가 응원하는 한 가지 구단만을 가지고 분석을 진행하였다. 추후에 프로젝트를 확장한다면 한 개의 구단이 아닌 KBO 전체 10개 구단을 대상으로 진행할 수 있을 것 같다. 각 구단마다 월별로 기사를 크롤링하여 워드 클라우드를 생성한다. 그리고 실제로 단어의 표현과 당월 리그 순위에 관계가 있는지, 긍정적인 단어가 많을수록 구단의 성적이 좋고 부정적인 단어가 많으면 성적이 좋지 않은지에 대해 분석해볼 수 있다.

또한 본 프로젝트는 word count를 하기 위해 데이터를 수집할 때 네이버 기사만을 가져와 분석을 했다는 한계가 있다. 기사는 정해진 형식이 있고 표준어만을 사용해야 하기 때문에 야구에 대한 기자들의 반응이 자유롭지 못한 경우가 많다. 조금 더 야구팬, 나아가 사람들의 현실적인 반응을 보려면 기사 뿐만 아니라 야구와 관련된 여러 커뮤니티의 글들을 크롤링하여 데이터를 수집하면 더욱 효과적이고 객관적인 결과가 도출될 것으로 기대된다.

7. 시행착오

처음에 시도한 프로젝트의 주제는 FA 계약 선수들의 계약 전후 성적 비교였다. 그런데 FA 계약을 진행한 선수만 따로 지정하여 데이터를 수집할 수 있는 방법을 찾지 못하였다. 전체 선수의 성적을 수집하고 FA 계약을 진행한 선수만 직접 한 명 한 명 지정하여 데이터를 추출해야 했는데, 해당 방법은 데이터의 용량이 아주 작을 뿐더러 빅데이터를 가지고 프로젝트를 진행하는 목적과 부합하지 않아 주제를 수정하게 되었다.

8. Github 주소

<https://github.com/JuJiHyun/bigdata-programming>

참고문헌

https://ko.wikipedia.org/wiki/KBO_%EB%A6%AC%EA%B7%B8 위키백과 KBO 리그

<https://www.koreabaseball.com/TeamRank/GraphDaily.aspx> KBO 구단별 순위 변동 그래프

<https://devanix.tistory.com/296> 파이썬 정규 표현식