캡스톤디자인 제작과정 보고서

일시:2019/06/26
팀명: 15조 MBS

형태소 분석기 비교

악의적인 댓글인지 아닌지 판단하기 위해 형태소 분석기를 이용하여 문장을 분석해야 한다.
많은 한글 형태소 분석기가 존재하는데 그 중에 우리 프로젝트에 맞고 성능이 좋은 형태소 분석기를 고르기 위해 6가지의 형태소 분석기를 가지고 실험을 진행 하였다 6가지 형태소 분석기는 konlpy에서 제공하는 한나눔 꼬꼬마 코모란 okt mecab와 요번에 새로 나온 khaiii를 가지고 진행하였다.

실험 환경

ubuntu 18.04
python 3.6.8

형태소 분석기 정확도 비교

형태소 분석기는 정확도가 무엇보다 중요하다고 생각해서 정확도를 먼저 분석해보았다 분석유형은 4가지이며 첫 번째로 긴 문장 문장이 길기 때문에 어떻게 이 문장에 대해 반응하는지 보기 위해 선택하였고 악플 방지를 피하기 위해 중간에 숫자나 알파벳을 넣는 경우도 있으므로 그때는 어떻게 반응하는지 보기 위해서 단어 중간에 숫자 있을 때 또 띄어쓰기가 틀린 경우도 많기 때문에 띄어쓰기가 없을 때 어떻게 되는지 마지막으로 저번에 웹크롤링을 실습할 때 마블과 같이 형태소가 고유명사에서 많이 사용해서 명사로 바뀐 경우가 있는데 이럴 때 잘 구별하는지를 보았다

1.긴 문장

2. 중간에 숫자나 알파벳이 있을 경우

3.띄어쓰기가 없을 경우

4.고유 명사가 명사로 바뀐 경우

1긴 문장

문장 : 내년도 최저임금을 기존 방식대로 전체 업종에 동일하게 적용하기로 결정했다.
최저임금의 업종별 차등 적용을 요구해온 사용자위원들은 이에 반발해 전원회의에서 퇴장했다.

1.한나눔

```
('내년', 'N')
('도', 'J')
('최저임금', 'N')
('을', 'J')
('기존', 'N')
('방식', 'N')
('대로', 'J')
('전체', 'N')
('업종', 'N')
('에', 'J')
('동일', 'N')
('하', 'X')
('게', 'E')
('적용', 'N')
('하', 'X')
('기', 'E')
('로', 'J')
('결정', 'N')
('하', 'X')
('었다', 'E')
('.', 'S')
('최저임금', 'N')
('의', 'J')
('업종별', 'N')
('차등', 'N')
('적용', 'N')
('을', 'J')
('요구', 'N')
('하', 'X')
('어', 'E')
('오', 'P')
('ㄴ', 'E')
('사용자위원들', 'N')
('은', 'J')
('이', 'N')
('에', 'J')
('반발', 'N')
('하', 'X')
('어', 'E')
('전원회의', 'N')
('에서', 'J')
('퇴장', 'N')
('하', 'X')
('었다', 'E')
('.', 'S')
```

2.꼬꼬마

```
('내년도', 'NNG')
('최저', 'NNG')
('임금', 'NNG')
('을', 'JKO')
('기존', 'NNG')
('방식', 'NNG')
('대로', 'JX')
('전체', 'NNG')
('업종', 'NNG')
('에', 'JKM')
('동일', 'NNG')
('하', 'XSV')
('게', 'ECD')
('적용', 'NNG')
('하', 'XSV')
('기로', 'ECD')
('결정', 'NNG')
('하', 'XSV')
('었', 'EPT')
('다', 'EFN')
('.', 'SF')
('최저', 'NNG')
('임금', 'NNG')
('의', 'JKG')
('업종별', 'NNG')
('차등', 'NNG')
('적용', 'NNG')
('을', 'JKO')
('요구', 'NNG')
('하', 'XSV')
('어', 'ECS')
('오', 'VV')
('ㄴ', 'ETD')
('사용자', 'NNG')
('위원', 'NNG')
('들', 'XSN')
('은', 'JX')
('이에', 'MAG')
('반발', 'NNG')
('하', 'XSV')
('어', 'ECS')
('전원', 'NNG')
('회의', 'NNG')
('에서', 'JKM')
('퇴장', 'NNG')
('하', 'XSV')
('었', 'EPT')
('다', 'EFN')
('.', 'SF')
```

3.komoran

```
('내년', 'NNG')
('도', 'JX')
('최저임금', 'NNP')
('을', 'JKO')
('기존', 'NNG')
('방식', 'NNG')
('대로', 'JX')
('전체', 'NNG')
('업종', 'NNG')
('에', 'JKB')
('동일', 'XR')
('하', 'XSA')
('게', 'EC')
('적용', 'NNG')
('하', 'XSV')
('기', 'ETN')
('로', 'JKB')
('결정', 'NNG')
('하', 'XSV')
('았', 'EP')
('다', 'EF')
('.', 'SF')
('최저임금', 'NNP')
('의', 'JKG')
('업종', 'NNG')
('별', 'XSN')
('차등', 'NNG')
('적용', 'NNG')
('을', 'JKO')
('요구', 'NNG')
('하', 'XSV')
('아', 'EC')
('오', 'VX')
('ㄴ', 'ETM')
('사용자', 'NNP')
('위원', 'NNP')
('들', 'XSN')
('은', 'JX')
('이', 'NP')
('에', 'JKB')
('반발', 'NNG')
('하', 'XSV')
('아', 'EC')
('전원', 'NNG')
('회의', 'NNG')
('에서', 'JKB')
('퇴장', 'NNG')
('하', 'XSV')
('았', 'EP')
('다', 'EF')
('.', 'SF')
```

4.okt

```
('내년', 'Noun')
('도', 'Josa')
('최저임금', 'Noun')
('을', 'Josa')
('기존', 'Noun')
('방식', 'Noun')
('대로', 'Josa')
('전체', 'Noun')
('업종', 'Noun')
('에', 'Josa')
('동일하게', 'Adjective')
('적용', 'Noun')
('하기로', 'Verb')
('결정', 'Noun')
('했다', 'Verb')
('.', 'Punctuation')
('최저임금', 'Noun')
('의', 'Josa')
('업종', 'Noun')
('별', 'Noun')
('차등', 'Noun')
('적용', 'Noun')
('을', 'Josa')
('요구', 'Noun')
('해온', 'Verb')
('사용자', 'Noun')
('위원', 'Noun')
('들', 'Suffix')
('은', 'Josa')
('이', 'Noun')
('에', 'Josa')
('반발', 'Noun')
('해', 'Verb')
('전', 'Modifier')
('원', 'Modifier')
('회의', 'Noun')
('에서', 'Josa')
('퇴장', 'Noun')
('했다', 'Verb')
('.', 'Punctuation')
```

5.mecab

```
('내년도', 'NNG')
('최저', 'NNG')
('임금', 'NNG')
('을', 'JKO')
('기존', 'NNG')
('방식', 'NNG')
('대로', 'JX')
('전체', 'NNG')
('업종', 'NNG')
('에', 'JKB')
('동일', 'NNG')
('하', 'XSV')
('게', 'EC')
('적용', 'NNG')
('하', 'XSV')
('기', 'ETN')
('로', 'JKB')
('결정', 'NNG')
('했', 'XSV+EP')
('다', 'EF')
('.', 'SF')
('최저임금', 'NNP')
('의', 'JKG')
('업종', 'NNG')
('별', 'XSN')
('차등', 'NNG')
('적용', 'NNG')
('을', 'JKO')
('요구', 'NNG')
('해', 'XSV+EC')
('온', 'VX+ETM')
('사용', 'NNG')
('자', 'XSN')
('들', 'XSN')
('은', 'JX')
('이', 'NP')
('에', 'JKB')
('반발', 'NNG')
('해', 'XSV+EC')
('전원', 'NNG')
('회의', 'NNG')
('에서', 'JKB')
('퇴장', 'NNG')
('했', 'XSV+EP')
('다', 'EF')
('.', 'SF')
```

6.khaiii

```
내년도  내년/NNG + 도/JX
최저임금을      최저/NNG + 임금/NNG + 을/JKO
기존    기존/NNG
방식대로        방식/NNG + 대로/JX
전체    전체/NNG
업종에  업종/NNG + 에/JKB
동일하게        동일/NNG + 하/XSA + 게/EC
적용하기로      적용/NNG + 하/XSV + 기/ETN + 로/JKB
결정했다.최저임금의     결정/NNG + 하/XSV + 였/EP + 다/EF + ./SF + 최저임금/NNG + 의/JKG
업종별  업종/NNG + 별/XSN
차등    차등/NNG
적용을  적용/NNG + 을/JKO
요구해온        요구/NNG + 하/XSV + 여/EC + 오/VX + ㄴ/ETM
사용자위원들은  사용자/NNG + 위원/NNG + 들/XSN + 은/JX
이에    이/NP + 에/JKB
반발해  반발/NNG + 하/XSV + 여/EC
전원회의에서    전원/NNG + 회의/NNG + 에서/JKB
퇴장했다.       퇴장/NNG + 하/XSV + 였/EP + 다/EF + ./SF
```

긴 문장을 분석할 경우 다른 것에 비해 좀 걸리네 라고 느낄 정도의 시간의 차이는 있었으나 대체로 형태소 분석이 잘된 것을 확인 할 수 있었다.

2. 중간에 숫자나 알파벳이 있을 경우

문장: 씨1발아 병a신 같은 놈 씨발 병신

순서 한나눔, khaiii, 꼬꼬마, komoran, mecab, okt

```
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/hannanum.py"
('씨1발', 'N')
('아', 'J')
('병a신', 'N')
('같', 'P')
('은', 'E')
('놈', 'N')
('이', 'J')
('씨발', 'N')
('병신', 'N')
time: 2.380690813064575
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/khaii.py"
씨1발아 씨/NNG + 1/SN + 발/NNB + 아/NNG
병a신   병/NNG + a/SL + 신/NNG
같은    같/VA + 은/ETM
놈이    놈/NNB + 이/JKS
씨발    씨발/NNG
병신    병신/NNG
time: 0.1069998741149902
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/kkma.py"
('씨', 'NNB')
('1', 'NR')
('발아', 'NNG')
('병', 'NNG')
('a', 'OL')
('신', 'NNG')
('같', 'VA')
('은', 'ETD')
('놈', 'NNB')
('이', 'JKS')
('씨', 'NNG')
('발', 'NNG')
('병신', 'NNG')
time: 10.49838924407959
```

```
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/komoran.py"
('씨', 'NNB')
('1', 'SN')
('발아', 'NNP')
('병', 'NNG')
('a', 'SL')
('신', 'NNP')
('같', 'VA')
('은', 'ETM')
('놈', 'NNB')
('이', 'JKS')
('씨발', 'IC')
('병신', 'NNP')
time: 4.272691965103149
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/mecab.py"
[('씨', 'NNG'), ('1', 'SN'), ('발아', 'NNG'), ('병', 'NNG'), ('a', 'SL'), ('신', 'NNG'), ('같', 'VA'), ('은', 'ETM'), ('놈', 'NNB'), ('이', 'JKS'), ('씨발', 'IC'),
 ('병신', 'NNG')]
time: 0.10049033164978027
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/twitter.py"
('씨', 'Noun')
('1', 'Number')
('발아', 'Noun')
('병', 'Noun')
('a', 'Alpha')
('신', 'Noun')
('같은', 'Adjective')
('놈', 'Noun')
('이', 'Josa')
('씨발', 'Noun')
('병신', 'Noun')
time: 6.6721155643463135
```

우리 프로젝트의 가장 중요한 욕설 중간에 숫자나 알파벳이 있을 때 과연 형태소 분석이 잘되는지 진행해 봤는데 전혀 되지 않았다 숫자나 알파벳을 기준으로 나뉘어서 따로 분석 되는 모습을 볼 수 있었다.

3.띄어쓰기가 없을 경우

문장: 집에가고싶다이제쉬고싶다
순서 한나눔, khaiii, 꼬꼬마, komoran, mecab, okt,

```
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/hannanum.py"
('집에가고싶다이제쉬', 'N')
('이', 'J')
('고', 'E')
('싶', 'P')
('다', 'E')
time: 1.9595725536346436
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/khaii.py"
집에가고싶다이제쉬고싶다        집/NNG + 에/JKB + 가/VV + 고/EC + 싶/VX + 다/EC + 이/MAG + 제쉬/VV + 고/EC + 싶/VX + 다/EC
time: 2.2533512115478516
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/kkma.py"
('집', 'NNG')
('에', 'JKM')
('가', 'VV')
('고', 'ECE')
('싶', 'VXA')
('다', 'ECS')
('이제', 'MAG')
('쉬', 'VV')
('고', 'ECE')
('싶', 'VXA')
('다', 'EFN')
time: 12.157114028930664
```

```
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/komoran.py"
('집', 'NNG')
('에', 'JKB')
('가', 'VV')
('고', 'EC')
('싶', 'VX')
('다', 'EC')
('이제', 'MAG')
('쉬', 'VV')
('고', 'EC')
('싶', 'VX')
('다', 'EC')
time: 4.376290321350098
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/mecab.py"
[('집', 'NNG'), ('에', 'JKB'), ('가', 'VV'), ('고', 'EC'), ('싶', 'VX'), ('다', 'EF'), ('이제', 'MAG'), ('쉬', 'VV'), ('고', 'EC'), ('싶', 'VX'), ('다', 'EC')]
time: 0.6167721748352051
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/twitter.py"
('집', 'Noun')
('에', 'Josa')
('가고싶다', 'Verb')
('이제', 'Noun')
('쉬고싶다', 'Verb')
time: 7.993001461029053
```

한나눔과 khaiii 제외하고는 형태소 분석이 대체로 잘되고 있다는 것을 알 수 있다 한나눔은 '집에가고싶다쉬'를 명사로 보았고 khaiii는 '이제 쉬고 싶다'에서 제쉬 라고 분석해버렸다

4.고유 명사가 명사로 바뀐 경우

문장: 마블 3000만큼 사랑합니다.

순서 한나눔, khaiii, 꼬꼬마, komoran, mecab, okt,

```
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/hannanum.py"
('마블', 'N')
('3000', 'N')
('만큼', 'J')                                   ('사랑', 'N')
('하', 'X')
('ㅂ니다', 'E')
time: 6.260161876678467
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/khaii.py"
마블    마/NNP + 블/NNG
3000만큼        3000/SN + 만큼/NNB
사랑합니다      사랑/NNG + 하/XSV + ㅂ니다/EC
time: 0.3556852340698242                        bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments
-Block-System/morphological analysis/kkma.py"
/usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/komoran.py"
('마블', 'NNG')
('3000', 'NR')
('만큼', 'NNM')
('사랑', 'NNG')
('하', 'XSV')
('ㅂ니다', 'EFN')
time: 16.399147033691406
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/komoran.py"
('마블', 'NNP')
('3000', 'SN')
('만큼', 'NNB')
('사랑합니다', 'NNP')
time: 4.249666452407837
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/mecab.py"
[('마블', 'NNG'), ('3000', 'SN'), ('만큼', 'JKB'), ('사랑', 'NNG'), ('합니다', 'XSV+EC')]
time: 0.9971177577972412
```

```
bong@ubuntu:~/Downloads/Malicious-comments-Block-System$ /usr/bin/python3 "/home/bong/Downloads/Malicious-comments-Block-System/morphological analysis/twitter.py"
('마블', 'Noun')
('3000만', 'Number')
('큼', 'Foreign')
('사랑', 'Noun')
('합니다', 'Verb')
time: 6.739722013473511
```

komoran은 마블을 NNP로 고유명사로 구별 했지만 그 외에 다른 khaiii 꼬꼬마 mecab는 NNG 일반명사로 구별이 잘 되었다 다른 okt와 한나눔은 명사라고 구별하고 그 이상 구별하지 않았다.

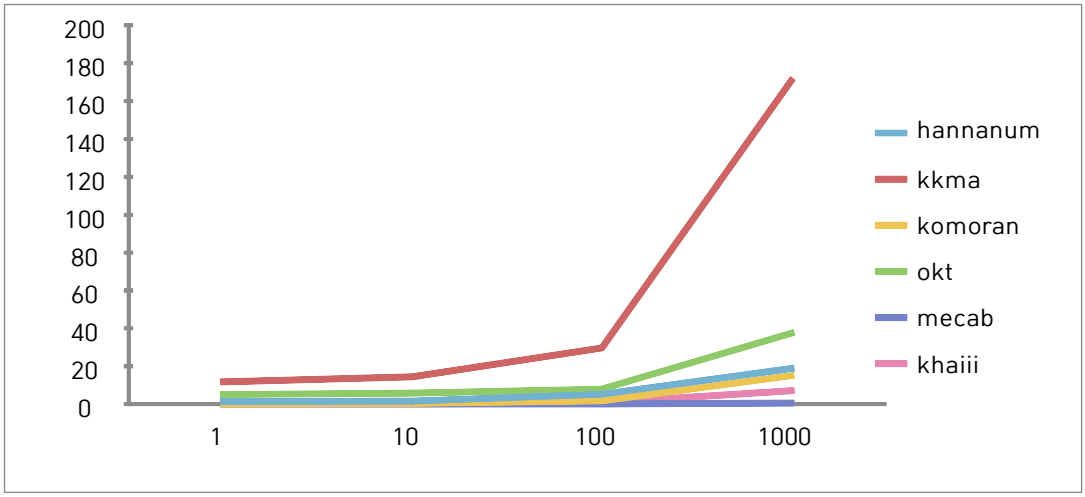
형태소 분석기 시간 분석 결과

좀 긴 문장을 해서 시간의 차이가 좀 더 잘 보일 수 있도록 분석 해보았다 이때 사용한 문장은 기사에 실린 긴 문장을 가져와서 사용해 보았다

‘ 내년도 최저임금을 기존 방식대로 전체 업종에 동일하게 적용하기로 결정했다. 최저임금의 업종별 차등 적용을 요구해온 사용자위원들은 이에 반발해 전원회의에서 퇴장했다. 최저임금위원회 사용자위원들은 이날 오후 정부세종청사에서 열린 최저임금위원회 제5차 전원회의 도중 퇴장해 기자들과 만나 "금일 최저임금위원회는 최저임금 고시에 월 환산액을 병기하고 2020년 최저임금을 모든 업종에 동일하게 적용하기로 결정했다"고 밝혔다.'

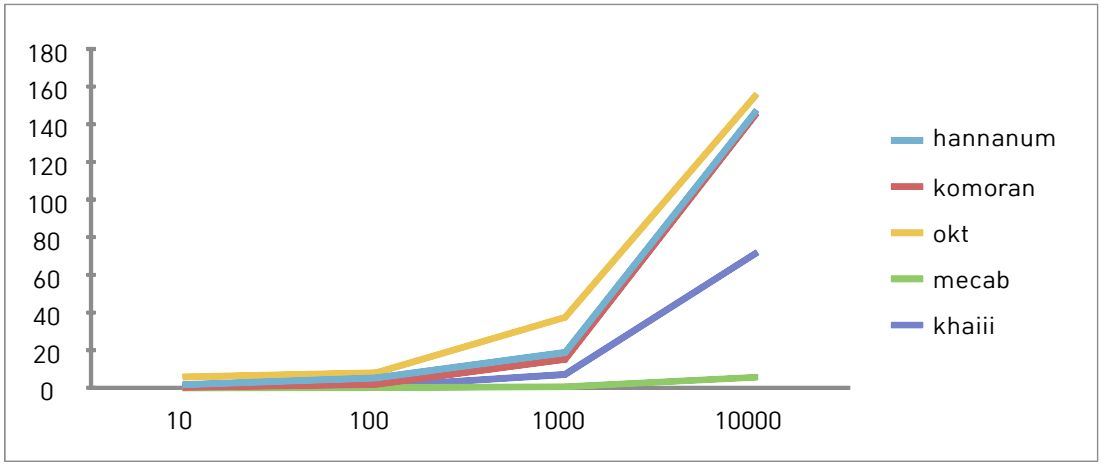
이 문장을 1,10,100,1000 반복해 1,10,100,1000문장일 때 형태소 분석 시간을 계산해보았다 이때 시간은 5회 반복해서 제일 낮은 시간을 측정했다

	1	10	100	1000
hannanum	1.44507	1.66621	5.25361	18.77751
kkma	11.80016	14.43480	29.63884	170.76234
komoran	0.08969	0.33671	1.83922	15.05882
okt	5.09221	5.79224	7.93883	37.38386
mecab	0.00335	0.01212	0.10388	0.57210
khaiii	0.00569	0.11792	0.72923	7.11792



꼬꼬마 형태소 분석기가 문장이 늘어날수록 기하급수적으로 시간이 늘어나 다른 형태소 분석기와 차이가 많이 난다는 것을 알 수 있다 따라서 꼬꼬마 형태소를 제외하고 만부터 다시 시간을 측정하였다

	10	100	1000	10000
hannanum	1.66621	5.25361	18.77751	146.22653
komoran	0.33671	1.83922	15.05882	144.60418
okt	5.79224	7.93883	37.38386	154.84640
mecab	0.01212	0.10388	0.57210	5.58000
khaiii	0.11792	0.72923	7.11792	71.19120



10만 문장부터는 시간이 너무 오래 걸리거나 파이썬이 강제로 죽는 경우가 발생하여 만 문장까지 측정하였다 전체적으로 봤을 때 mecab가 가장 빠르게 분석을 하였고 그다음이 khaiii 나머지는 비슷한 시간대를 보여주었다.

결론

전체적으로 봤을 때 한나눔과 khaiii는 띄어쓰기를 잘 지키지 않는 문장을 형태소 분석할 때 오류가 있다는 것을 알 수 있었다

시간적으로 봤을 때 mecab가 가장 빨랐으며 그다음이 khaiii였다 mecab는 연산속도가 중요할 때 사용하면 좋을 거 같고 심지어 분석능력 또한 좋았다

okt, 한나눔은 다른 형태소 분석기와 달리 명사를 고유명사 일반명사 같이 세세하게 들어가지 않고 그냥 명사로 구별하고 있었다. 세세한 분석이 필요하면 한나눔과 okt 대신 다른 형태소 분석기가 적합할거 같다.

우리 프로젝트는 많은 댓글이 달렸을 때 사용할 수도 있기 때문에 연산속도가 중요하다고 생각한다. 또한 댓글을 다는 사람이 띄어쓰기를 잘 지키지 않을 수도 있기 때문에 띄어쓰기가 없었던 문장 분석을 할 때 약했던 khaiii와 한나눔은 적합하지 않았던 거 같다 따라서 처음 생각한 khaiii 대신 mecab를 사용하는 것이 프로젝트에 더 적합하다고 느껴져 mecab 형태소 분석기를 사용하기로 하였다. 중간에 숫자와 영문이 들어갔을 때 형태소 분석이 잘되지 않았는데 이 부분을 해결하는 방법이 필요하다고 생각한다.

참고 문헌

khaiii 깃헙 https://github.com/kakao/khaiii,
konlpy 홈페이지 https://konlpy-ko.readthedocs.io/ko/v0.4.3/