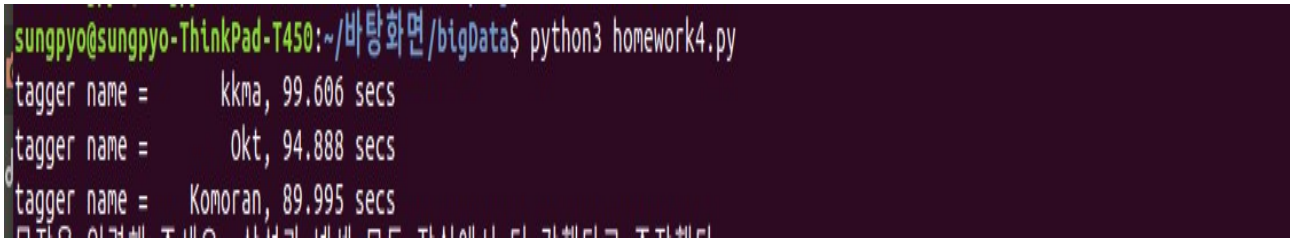
빅데이터 최신기술 과제4 보고서

20152900 홍성표

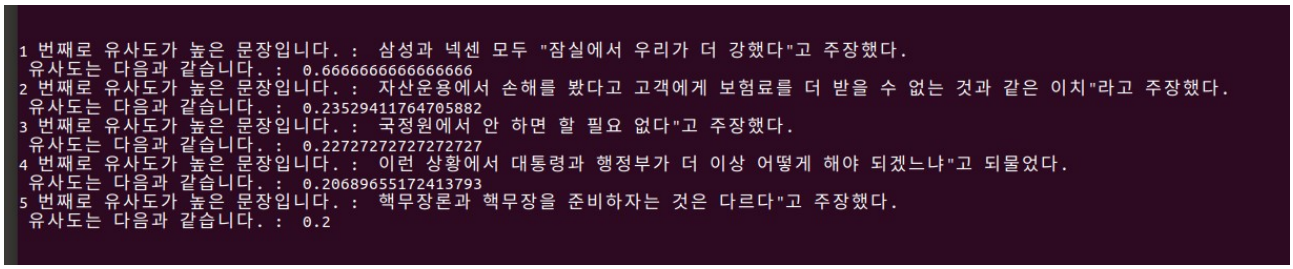
형태소 분석기를 통하여 분석해본 결과입니다. 형태소 분석은 Kkma, Okt, Komoran 을 3개를 사용하여 정확도 및 실행 시간을 비교한 결과입니다.

먼저 결과화면입니다.

데이터는 5000줄 라인의 kcc28을 5000줄 정도로 수정한후 실행하였습니다.



1. Kkma 실행시간은 약 100초 정도로 3개 분석기중에 가장 오래 걸립니다. 밑에서 보면 알겠지만 정확한 품사나 정보가 필요할때 사용하면 좋습니다.



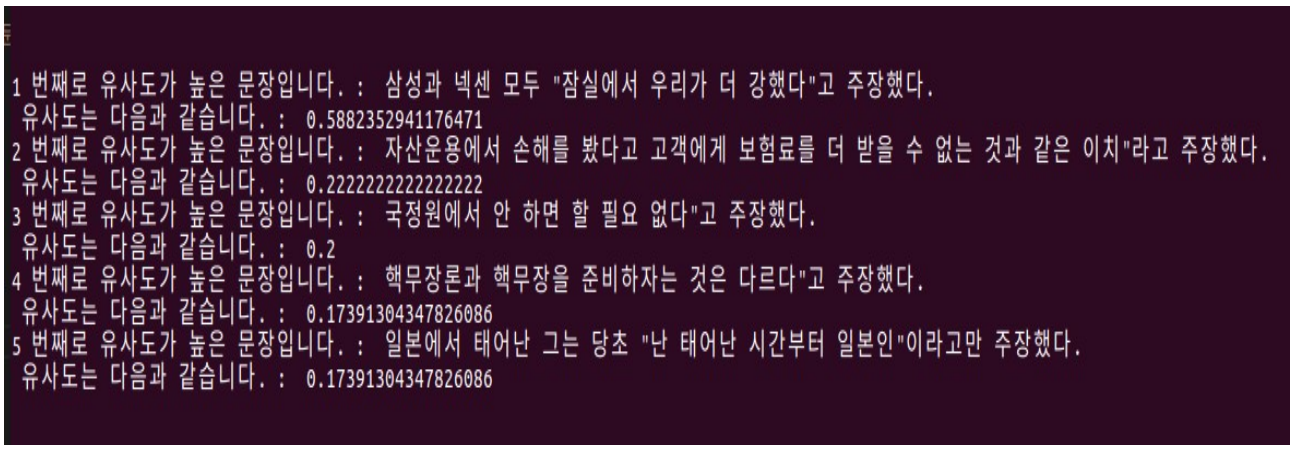
정확도 높은 순서대로 kkma 문장 입니다.
입력문장 : “잠실과 넥센 모두 잠실에서 더 강했다고 주장했다.” 입니다

가장 유사한 문장은 0.66 정도의 높은 유사도를 보이며 5개 문장을 출력하였습니다.

1.삼성과 넥센 모두 “잠실에서 우리가 더 강했다”고 주장했다.
2. 자산운용에서 손해를 봤다고 고객에게 보험료를 더 받을 수 없는 것과 같은 이치”라고 주장했다.
3.국정원에서 안하면 할 필요 없다”고 주장했다.
4. 이런 상황에서 대통령과 행정부가 더 이상 어떻게 해야 되겠느냐”고 되물었다.
5. 핵무장론과 핵무장을 준비하자는 것은 다르다”고 주장했다.

다음은 Okt 형태소 분석기 입니다.

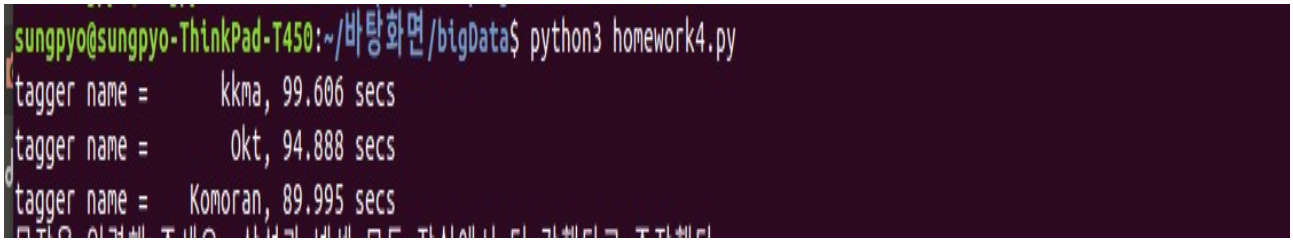
실행시간은 약 94초 정도로 측정되었습니다.
상대적으로 빠른 분석에 많이 사용합니다. 자신의 상황이 빠른 분석을 요구한다면 이러한 이 형태소 분석기를 사용하는게 가장 좋은거 같습니다.



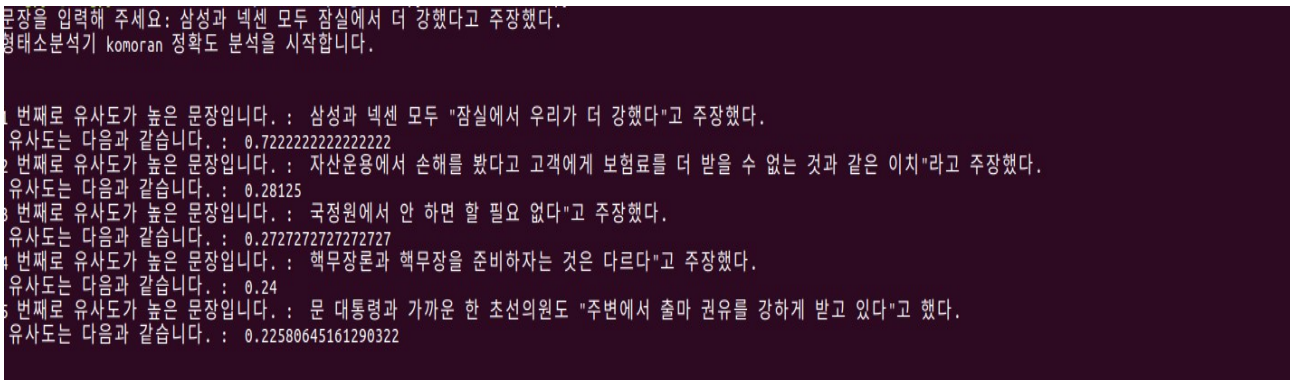
유사도 분석 결과는 다음과 같습니다.
1번째로 유사도가 높은 문장은 약 0.588 의 정확도를 가집니다. 정확도는 확실히 이전 kma 보다는 떨어지는게 눈에 보입니다.

1. 삼성과 넥센 모두 잠실에서 우리가 더 강했다고 주장했다.
2 자산운용에서 손해를 봤다고 고객에게 보험료를 더 받을 수 없는 것과 같은 이치” 라고 주장했다.
3. 국정원에서 안 하면 할 필요 없다”고 주장했다.
4. 핵무장론과 핵무장을 준비하자는 것은 다르다고 주장했다.
5. 일본에서 태어난 그는 당초 난 태어난 시간부터 일본인 이라고만 주장했다.

마지막 형태소 분석기 komoran 분석기 입니다.



분석시간은 약 89초이며 가장 정확도 측면에서는 뛰어난다고 볼 수 있습니다.



약 0.72의 정확도를 보입니다. 세 형태소 분석기중에 성능이 가장 좋다고 할 수 있습니다.

1.삼성과 넥센 모두 잠실에서 우리가 더 강했다고 주장했다.
2. 자산운용에서 손해를 봤다고 고객에게 보험료를 더 받을 수 없는 것과 같은 이치라고 주장했다.
3. 국정원에서 안하면 할 필요 없다고 주장했다
4. 핵무론과 핵무장을 준비하자는 것은 다르다고 주장했다
5. 문대통령과 가까운 한 초선의원도 주변에서 출마 권유를 강하게 받고 있다고 했다.

다음은 코드에 대한 설명입니다.

```
def RunningSpeed():
    pos_taggers = [('kkma', Kkma()), ('Okt', Okt()), ('Komoran', Komoran())]
    results = []
    for name, tagger in pos_taggers:
        tokens = []
        process_time = time.time()
        texts = open('input3.txt')
        for text in texts:
            testLine = texts.readline()  #형태소를 라인별로 받아서
            n = km.morphs(testLine)   # morphs 함수를 사용하여 형태소를 분석합니다.
        process_time = time.time() - process_time ㅎ
        print('tagger name = %10s, %.3f secs' % (name, process_time))
        results.append(tokens)
```

시간 측정 함수 입니다. 여기서 형태소를 라인별로 받아 morphs 함수를 써서 차례대로 kkma , okt, komoran 형태소 분석기 시간측정을 합니다.

```
def Jaccard_similarty(text1,text2):
    res = set(text1).union(set(text2))
    intersection = set(text1).intersection(set(text2))
    similarty.append(len(intersection)/len(res))
    #print(similarty)
```

자카드 유사도를 사용한 함수입니다. 두개의 형태소 분석한 리스트를 합쳐서 교집합을 계산해 줍니다. 유사도를 측정하기 위한 매우 중요한 함수입니다.

```
def accuracy(name):
    reslut = []
    if name == 'kma':
        mode = Kkma()
    elif name == 'okt':
        mode = Okt()
    elif name == 'komoran':
        mode = Komoran()
    else :
        return 0
```

```
    mylin = input ("문장을 입력해 주세요: " )

    print("형태소분석기",name,"정확도 분석을 시작합니다. ")
    print('\n')
    acc = mode.morphs(mylin) # 입력문장 형태소 분석
    for sentence in texts:
        arr.append(sentence)
        sp_text = mode.morphs(sentence) # 한줄씩 문장별로 잘라서 형태소 분석
        Jaccard_similarty(acc,sp_text) # 자칼드 유사도로 유사도 계산

    n = 5
    Sortsimilarty = sorted(range(len(similarty)),key=lambda i: similarty[i], reverse=True)[:n]
# 결과를 sort 해줍니다.

    k = 0
    for i in Sortsimilarty:
        k = k+1
        print( k ,"번째로 유사도가 높은 문장입니다. : ",arr[i],"유사도는 다음과 같습니다. : ",
similarty[i] )

    print('\n')
```

정확도를 계산하는 함수입니다. 라인별로 형태소 분석기를 사용하고 입력받은 문장과 자칼드 유사도를 통해서 merge 해주는 함수입니다. 이 함수를 통해 최종 유사도 값을 나타내줍니다.