악성댓글 자동수집

빅데이터 캡스톤디자인 / 고소미

20157131 김재석(3.5)
20155134 신다연(2.0)
20155135 심수빈(2.0)
20175337 정연선(2.5)

차례

프로젝트 목표

15,043

악성댓글 발견 수

85%

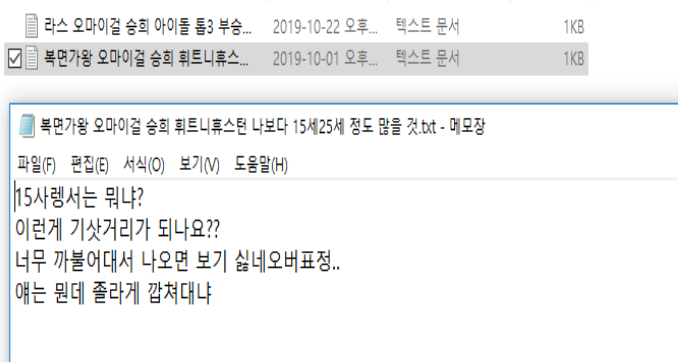
악성댓글 분류 학습률

10m

악성댓글 수집시간

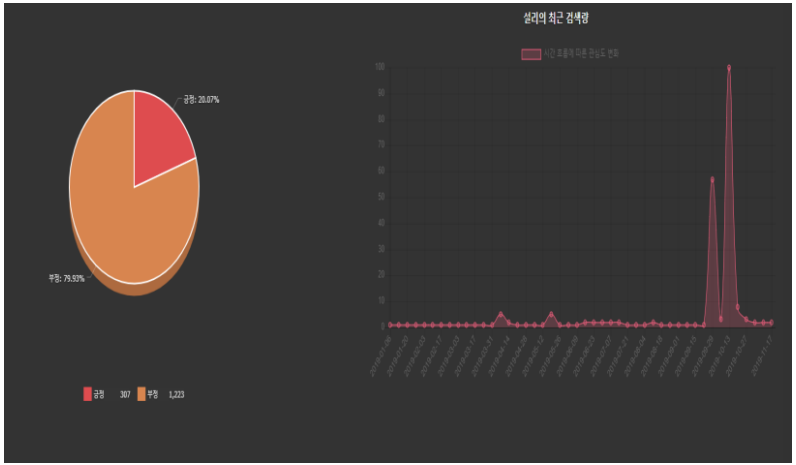
프로젝트 목표

- 기사의 댓글 중 악성댓글을 판별하여 수집
- 사용자에게 보다 효과적으로 전달하기 위해 데이터를 여러 종류로 시각화



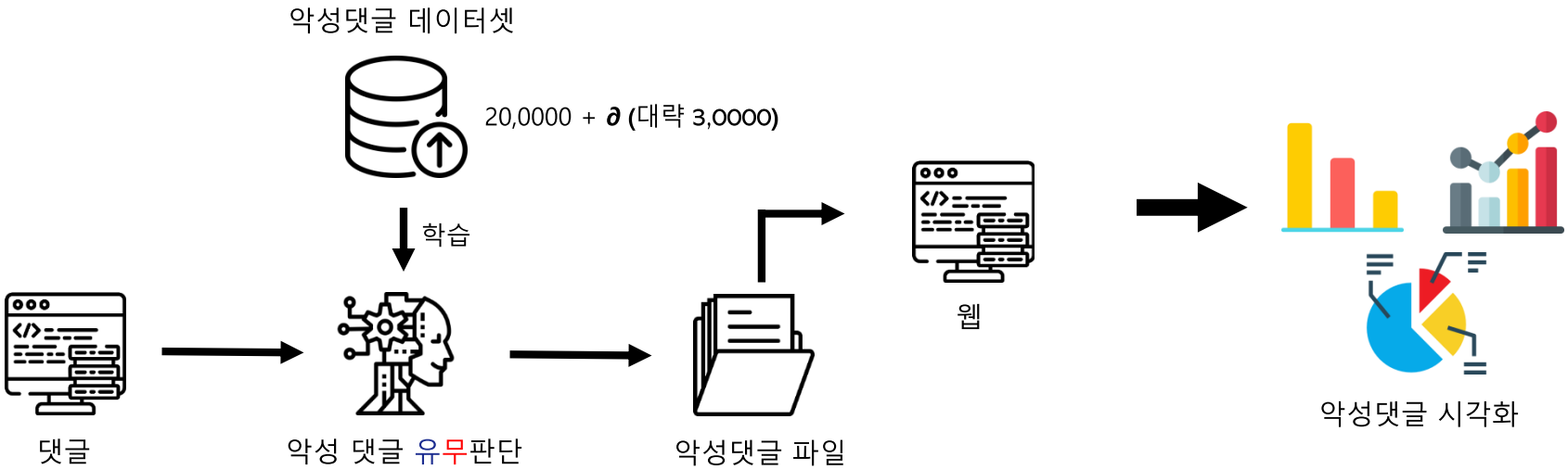
프로젝트 목표

- 기사의 댓글 중 악성댓글을 판별하여 수집
- 사용자에게 보다 효과적으로 전달하기 위해 데이터를 여러 종류로 시각화



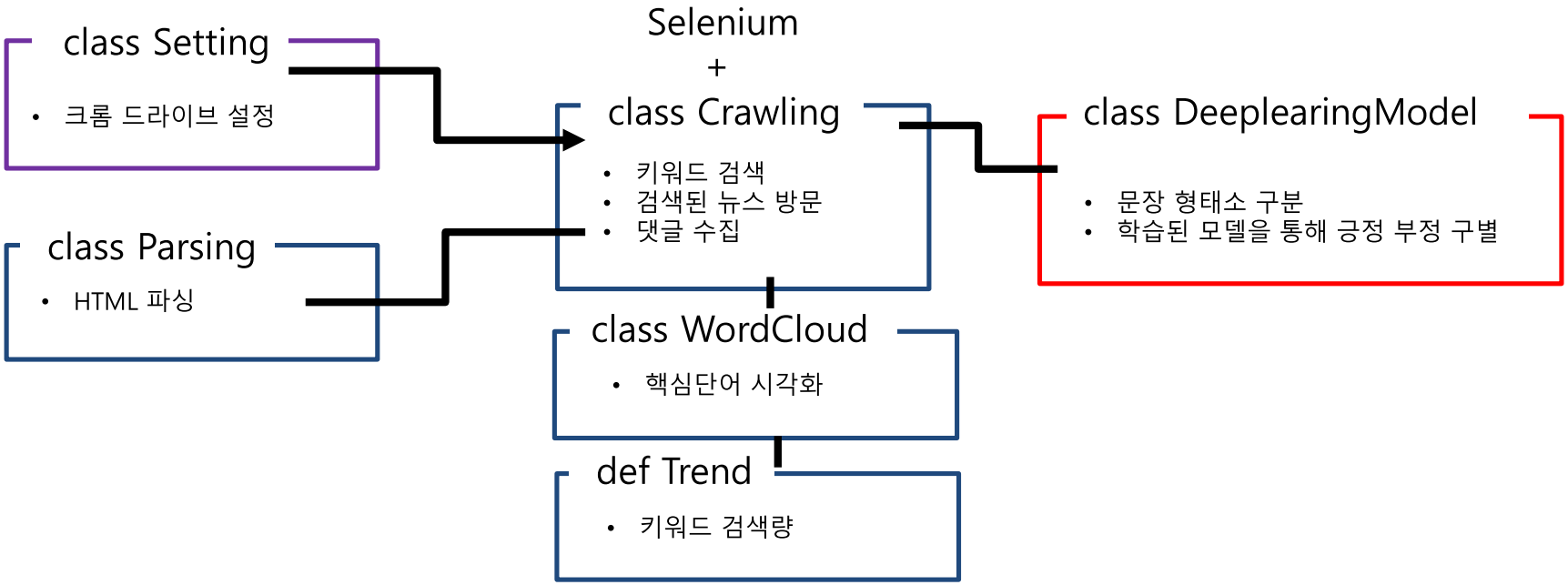
프로젝트 내용

- 시스템 구성도



프로젝트 내용

- 시스템 구성도



프로젝트 내용

- 시스템 기술
  - 데이터셋 (https://github.com/e9t/nsmc/)

    20,0000+ ***∂*** **(**대략3,0000**)**

    (부정 1, 긍정0)
  - Konlpy를 이용하여 형태소를 구별
  - Nltk를 이용하여 문자열을 작은 단위로 나누는 토큰화 작업

```
def tokenize(doc):
    return ['/'.join(t) for t in okt.pos(doc, norm=True, stem=True)]
```

형태소 구별

```
import nltk
text = nltk.Text(tokens, name='NMSC')
```

토큰화

프로젝트 내용

- 학습 모델
  - 예측 값이 부정과 긍정 둘 중 하나인 형식이므로 binary_crossentropy사용
  - 10번 학습 진행
  - 85%의 성능 도출

```
#모델 학습과정 설정.
model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss=losses.binary_crossentropy,
              metrics=[metrics.binary_accuracy])

# 모델 학습하기.
model.fit(x_train, y_train,validation_data=(x_test, y_test), epochs=10, batch_size=512)

# 모델 평가 하기.
results = model.evaluate(x_test, y_test)
```

```
(150000, 10000)
WARNING:tensorflow:From C:\Python\Anaconda\lib\site-packages\tensorflow\python\keras\initializers.py:119: calling RandomUniform.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\Python\Anaconda\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\Python\Anaconda\lib\site-packages\tensorflow\python\ops\nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 150000 samples, validate on 50000 samples
Epoch 1/10
150000/150000 [==============================] - 9s 61us/sample - loss: 0.3931 - binary_accuracy: 0.8304 - val_loss: 0.3550
- val_binary_accuracy: 0.8468
Epoch 2/10
150000/150000 [==============================] - 8s 52us/sample - loss: 0.3171 - binary_accuracy: 0.8651 - val_loss: 0.3468
- val_binary_accuracy: 0.8516
Epoch 3/10
150000/150000 [==============================] - 8s 51us/sample - loss: 0.2910 - binary_accuracy: 0.8782 - val_loss: 0.3448
- val_binary_accuracy: 0.8543
Epoch 4/10
150000/150000 [==============================] - 8s 52us/sample - loss: 0.2731 - binary_accuracy: 0.8868 - val_loss: 0.3507
- val_binary_accuracy: 0.8553
Epoch 5/10
150000/150000 [==============================] - 7s 50us/sample - loss: 0.2563 - binary_accuracy: 0.8980 - val_loss: 0.3517
- val_binary_accuracy: 0.8558
Epoch 6/10
150000/150000 [==============================] - 8s 50us/sample - loss: 0.2388 - binary_accuracy: 0.9061 - val_loss: 0.3540
- val_binary_accuracy: 0.8560
Epoch 7/10
150000/150000 [==============================] - 8s 53us/sample - loss: 0.2211 - binary_accuracy: 0.9142 - val_loss: 0.3624
- val_binary_accuracy: 0.8558
Epoch 8/10
150000/150000 [==============================] - 8s 53us/sample - loss: 0.2031 - binary_accuracy: 0.9214 - val_loss: 0.3782
- val_binary_accuracy: 0.8538
Epoch 9/10
150000/150000 [==============================] - 7s 50us/sample - loss: 0.1862 - binary_accuracy: 0.9293 - val_loss: 0.3906
- val_binary_accuracy: 0.8538
Epoch 10/10
150000/150000 [==============================] - 7s 50us/sample - loss: 0.1704 - binary_accuracy: 0.9361 - val_loss: 0.4111
- val_binary_accuracy: 0.8513
50000/50000 [==============================] - 3s 68us/sample - loss: 0.4111 - binary_accuracy: 0.8513
Out[15]: [0.41109749904394915, 0.85132]
```

프로젝트 내용

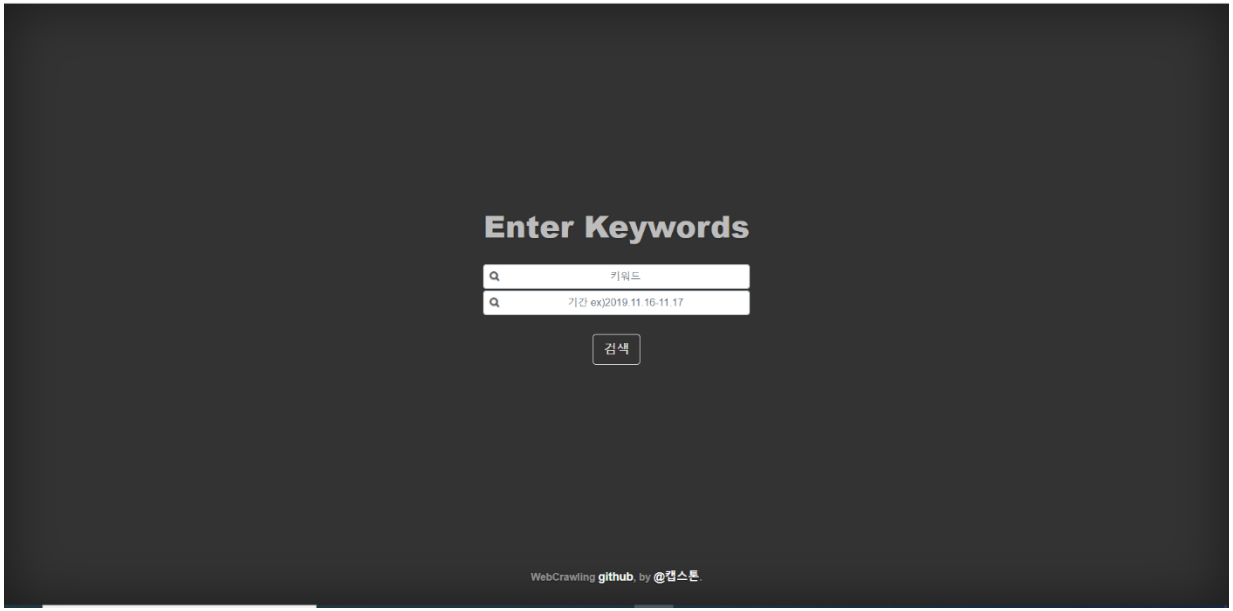
- 네이버 키워드 검색

```
"https://search.naver.com/search.naver?where=news&query=" + keyword \
+ "&sort=" + str(sort) + "&ds=" + self.start_date + "&de=" + self.end_date \
+ "&nso=so%3Ar%2Cp%3Afrom" + s_from + "to" + e_to + "%2Ca%3A&start=" + str(page)
```

1. Keyword : 검색단어.
2. Sort : 뉴스 정렬 순서(0-> 관련도 순서, 1 -> 최신 순서 , 2 -> 오래된 순서)
3. Start_date : 시작 날짜. (ex 2019.10.21)
4. End_date : 마지막날짜. (ex 2019.10.22)
5. S_from : 시작날짜 -> 20191021
6. e_to : 마지막날짜 -> 20191022
7. Page : 페이지 (1 페이지 -> 1, 2 페이지 -> 11, 3 페이지 -> 21)
   * 패턴 => (page-1) * 10 +1

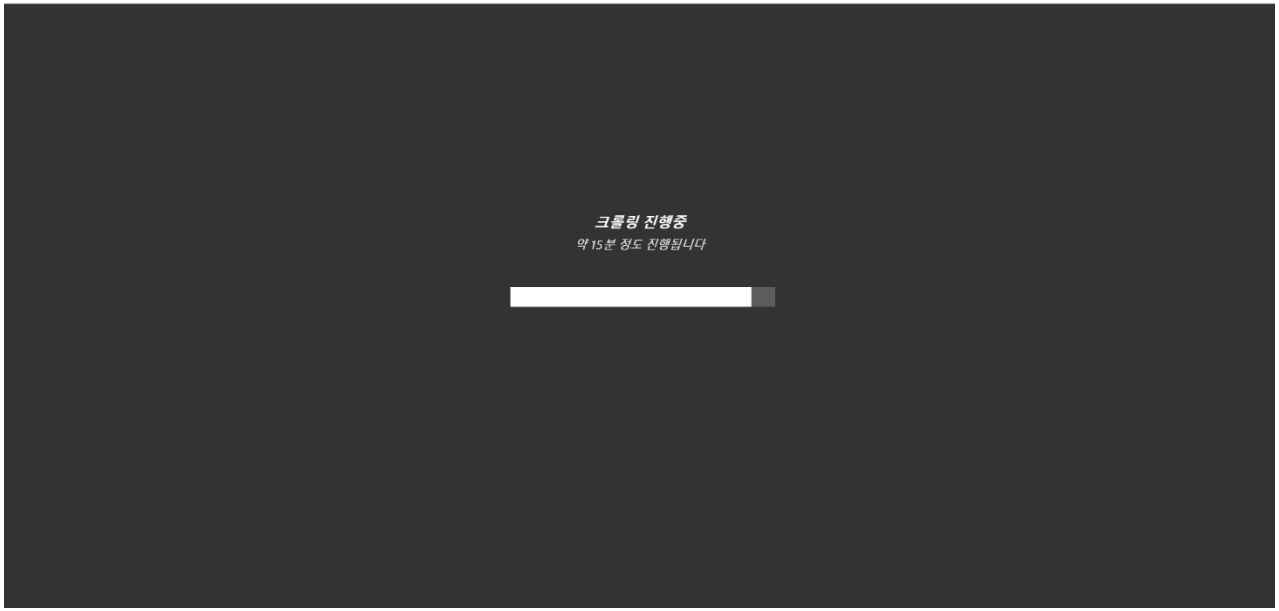
프로젝트 결과

- 네이버 키워드 검색 - index



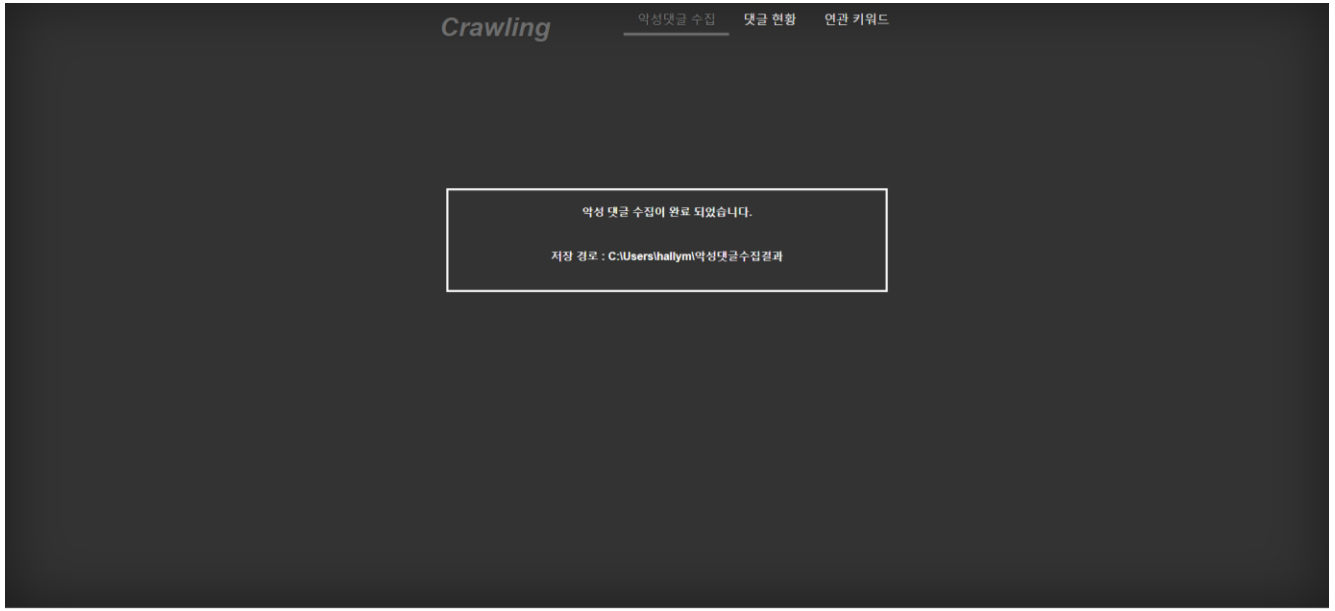
프로젝트 결과

- 크롤링 진행 창 - crawling



프로젝트 결과

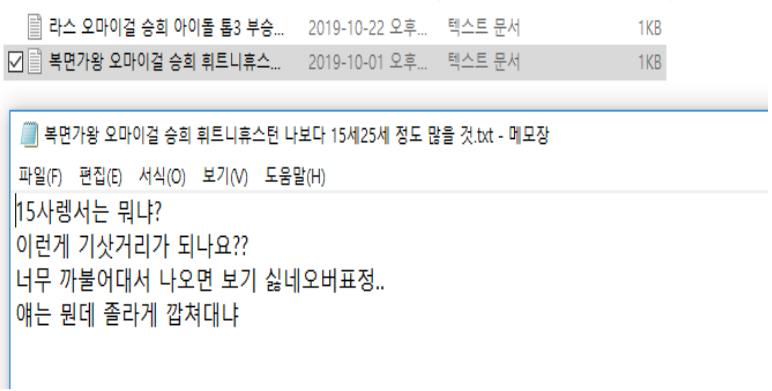
- 악성댓글 수집 경로 – menu1



프로젝트 결과

- 악성댓글 수집 경로 – menu1

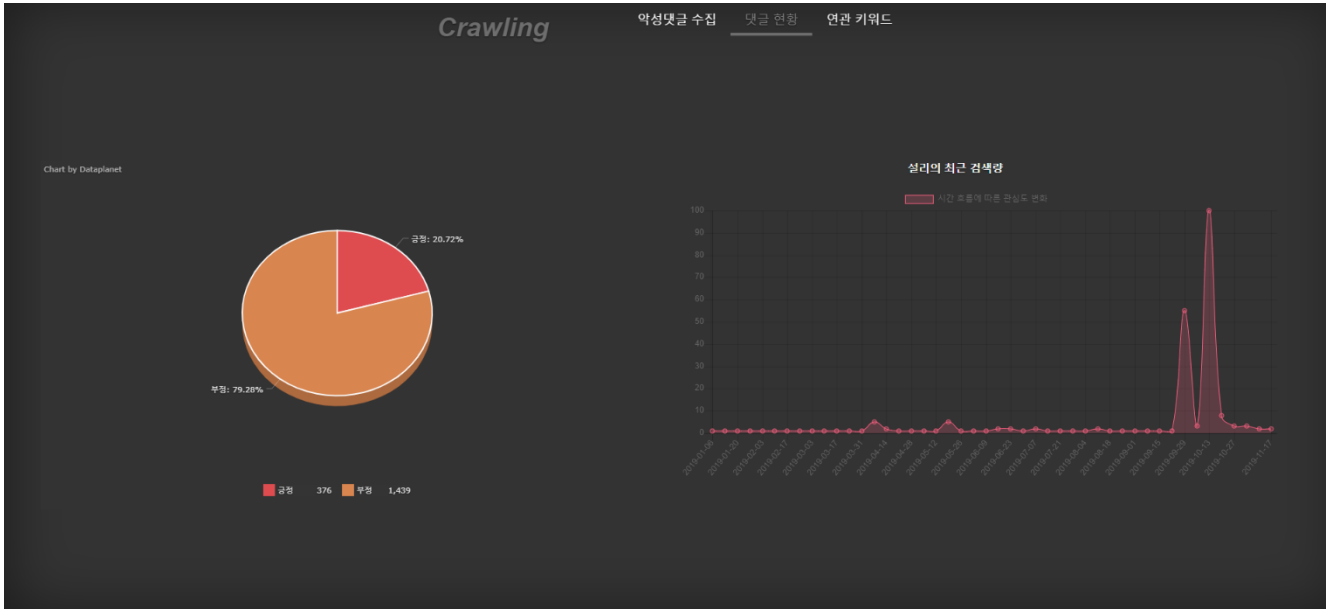
악성댓글 분류 폴더



악성댓글 내용

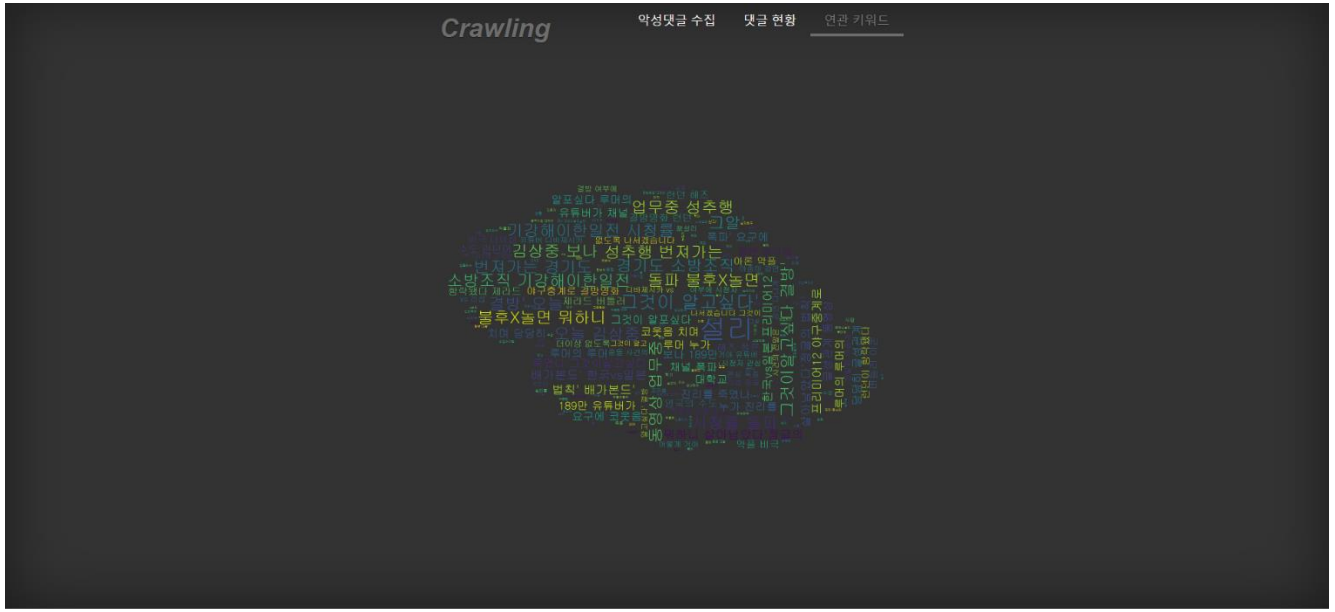
프로젝트 결과

- 키워드 시각화 (그래프, 차트) – menu2



프로젝트 결과

- 키워드 시각화 (워드 클라우드) – menu3



활용방안 및 기대효과

- 툴을 이용해 수집한 악성댓글을 법적제출용 증거자료로 활용
- 해당 연예인이나 기업의 시간에 따른 이미지 변화 분석 가능
- 키워드와 관련된 키워드를 워드 클라우드를 통해 쉽게 파악
- 악성댓글의 실질적인 심각성을 몸소 느낄 수 있음

활용방안 및 기대효과

- 일반인이 악성댓글의 심각성을 직접 경험함으로써 악성댓글의 문제를 해결하는데 이바지
- 기업에서 출시한 제품에 대한 시장조사를 할 때 설문조사의 절차없이 반응을 알 수 있음
- 이전의 기록들을 보다 쉽고 편리하게 검색하고 찾아낼 수 있음

THANK YOU