



## 게임리뷰 데이터를 이용한 감성 분석 및 맞춤 감정 사전 구축

조원: 18510068 문주영  
18512083 김지효



## 목차

- 1.연구배경
- 2.선행 연구
- 3.연구 의의 및 연구 방법
- 4.연구 프레임 워크
- 5.연구 결과
- 6.결론

# Background



## 연구 배경

국내 게임 시장 뿐만 아니라 세계 게임 시장 규모 또한 증가세.

도표 25. 국내 게임 시장 규모 및 전망

| 구 분     | 2014   |        | 2015    |        | 2016(P) |       | 2017(E) |       | 2018(E) |       |
|---------|--------|--------|---------|--------|---------|-------|---------|-------|---------|-------|
|         | 매출액    | 성장률    | 매출액     | 성장률    | 매출액     | 성장률   | 매출액     | 성장률   | 매출액     | 성장률   |
| 온라인게임   | 55,425 | 1.7%   | 52,804  | -4.7%  | 52,390  | -0.8% | 53,480  | 2.1%  | 54,490  | 1.9%  |
| 모바일게임   | 29,136 | 25.2%  | 34,844  | 19.6%  | 38,905  | 11.7% | 55,320  | 42.2% | 61,000  | 10.3% |
| 비디오게임   | 1,598  | 70.7%  | 1,661   | 3.9%   | 1,670   | 0.5%  | 1,698   | 1.7%  | 1,672   | -1.5% |
| PC게임    | 337    | -11.3% | 379     | 12.5%  | 390     | 2.9%  | 401     | 2.8%  | 412     | 2.7%  |
| 아케이드게임  | 528    | -35.9% | 474     | -10.3% | 476     | 0.4%  | 482     | 1.3%  | 490     | 1.7%  |
| PC방     | 12,277 | -26.1% | 16,604  | 35.2%  | 18,901  | 13.8% | 17,609  | -6.8% | 16,789  | -4.7% |
| 아케이드게임장 | 405    | -36.6% | 457     | 13.0%  | 462     | 1.1%  | 479     | 1.7%  | 472     | 0.4%  |
| 합계      | 99,706 | 2.6%   | 107,223 | 7.5%   | 113,194 | 5.6%  | 129,469 | 14.4% | 135,325 | 4.5%  |

자료 : 게임백서2016, 신영증권 리서치센터

자료: 게임백서2016, 신영증권 리서치센터



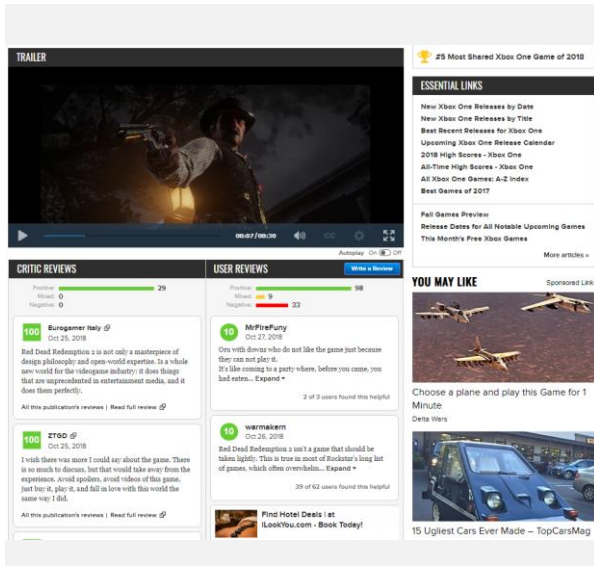
자료: NEWZOO  
(2018년 4월 Global Game Market Report)

# Background

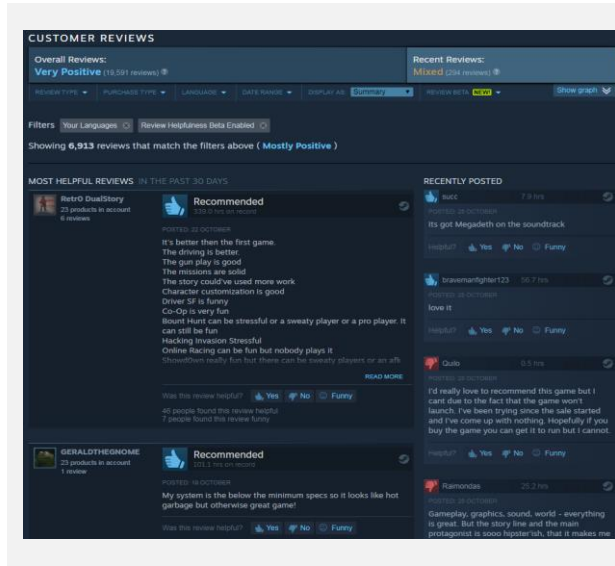


## 연구 배경

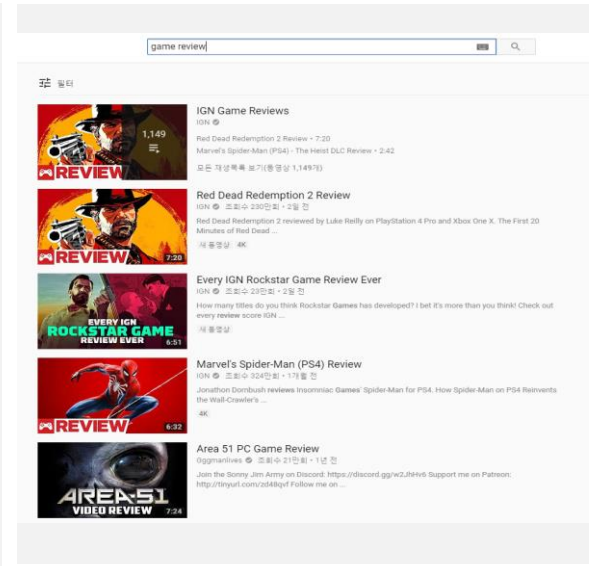
그에 따라 여러 매체를 이용한 게임 리뷰 등장 및 게임 리뷰를 목적으로 하는 사이트 출현.



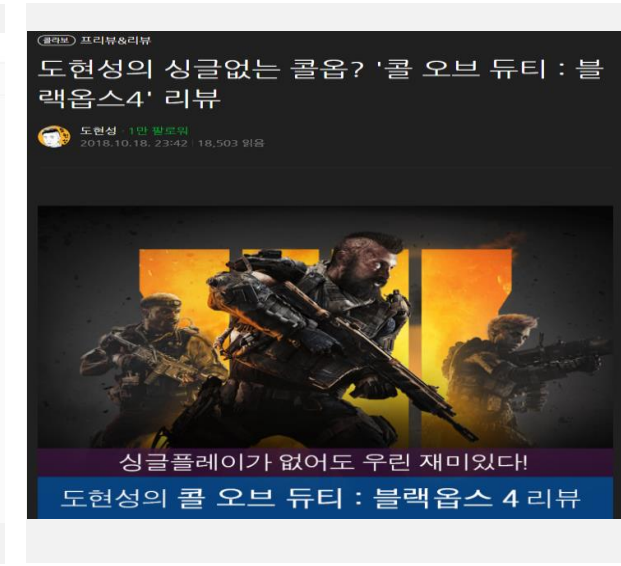
www.metacritic.com  
(accessed: October 2018)



store.steampowered.com  
(accessed: October 2018)



www.youtube.com  
(accessed: October 2018)



post.naver.com/my.nhn?member  
No=21542240 (accessed: October  
2018)

# Background



## 선행 연구

search keyword: steam platform game reviews

### An empirical study of **game reviews** on the **Steam platform**

D Lin, CP Bezemer, Y Zou, AE Hassan - Empirical Software Engineering, 2018 - Springer

The steadily increasing popularity of computer games has led to the rise of a multi-billion dollar industry. Due to the scale of the computer **game** industry, developing a successful **game** is challenging. In addition, prior studies show that gamers are extremely hard to ...

☆ 99 관련 학술자료 전체 3개의 버전 》》

### An empirical study of early access games on the **Steam platform**

D Lin, CP Bezemer, AE Hassan - Empirical Software Engineering, 2018 - Springer

... the **Steam platform** in the early access stage of a **game**. In addition, we expect that developers post more updates for an EAG, as they are improving the **game** (for example, based on the feedback that they acquire from user **reviews**). Approach. We compare the average **review** ...

☆ 99 1회 인용 관련 학술자료 전체 4개의 버전 》》

### [HTML] Mining personality traits from social messages for **game** recommender systems

HC Yang, ZR Huang - Knowledge-Based Systems, 2018 - Elsevier

... **Steam** is a major gaming **platform** which provides access to various types of games ... The **game**-related data were retrieved using **Steam** Web API [47] and import.io Web scraper service ... For **game**-centric approach, the personality traits of a **game** G i were calculated by the ...

☆ 99 전체 2개의 버전

### How do Sales Responses to Various User-generated-content?

Y Zhang, J Zhang - 2018 - aisel.aisnet.org

... distribution **platform**- **Steam**, and **game**-related user generated video content data from YouTube and Twitch ... For content provider perspective: (1) A **game**'s popularity on live stream video **platform**, popularity on pre-recorded video **platform** and its **review** have limited ...

☆ 99

### Sentiment Analysis of **Steam Review** Datasets using Naive Bayes and Decision Tree Classifier

Z Zuo - 2018 - ideals.illinois.edu

... All of them are free third-party tools designed to give better insight into the applications and packages. It provides more comprehensive information than **Steam** official API ... Like Youtube, Twitch is a large Page 2. online **game** steaming **platform** ...

☆ 99 》》

- 게임 별 특성과 매일 갱신되는 리뷰 수의 변동 사이의 관계
- 게임 리뷰의 유용한 정보들과 리뷰 작성 시 유저의 게임플레이시간과의 관계
- 모바일 앱의 리뷰와 스팀 리뷰와의 다른 양상을 알아봄.
- Future study 제시
  - Negative reviews: 버그 리포트 같이 게임을 개선할 수 있는 가치 있는 정보들을 포함.
  - 그렇다고 positive reviews에 담겨 있는 유용할 수 있는 리뷰들을 무시하면 안됨.

- 데이터 수집: steam official API가 아닌 steam의 데이터를 받아 쓰는 steamDB, steam spy라는 API를 통해 수집
- Positive와 negative를 분류하도록 학습시키는 sentiment analysis
- Decision tree와 Gaussian Naïve Bayes를 사용하여 classification model의 정확도 비교 연구 수행

# Background



## 선행 연구

search keyword: steam platform game reviews

### An empirical study of **game reviews** on the **Steam platform**

D Lin, CP Bezemer, Y Zou, AE Hassan - Empirical Software Engineering, 2018 - Springer

The steadily increasing popularity of computer games has led to the rise of a multi-billion dollar industry. Due to the scale of the computer **game** industry, developing a successful **game** is challenging. In addition, prior studies show that gamers are extremely hard to ...

☆ 99 관련 학술자료 전체 3개의 버전 》》

### An empirical study of early access games on the **Steam platform**

D Lin, CP Bezemer, AE Hassan - Empirical Software Engineering, 2018 - Springer

... the **Steam platform** in the early access stage of a **game**. In addition, we expect that developers post more updates for an EAG, as they are improving the **game** (for example, based on the feedback that they acquire from user **reviews**). Approach. We compare the average **review** ...

☆ 99 1회 인용 관련 학술자료 전체 4개의 버전 》》

### [HTML] Mining personality traits from social messages for **game** recommender systems

HC Yang, ZR Huang - Knowledge-Based Systems, 2018 - Elsevier

... **Steam** is a major gaming **platform** which provides access to various types of games ... The **game**-related data were retrieved using **Steam** Web API [47] and import.io Web scraper service ... For **game**-centric approach, the personality traits of a **game** G i were calculated by the ...

☆ 99 전체 2개의 버전

### How do Sales Responses to Various User-generated-content?

Y Zhang, J Zhang - 2018 - aisel.aisnet.org

... distribution **platform**- **Steam**, and **game**-related user generated video content data from YouTube and Twitch ... For content provider perspective: (1) A **game's** popularity on live stream video **platform**, popularity on pre-recorded video **platform** and its **review** have limited ...

☆ 99

### Sentiment Analysis of **Steam Review** Datasets using Naive Bayes and Decision Tree Classifier

Z Zuo - 2018 - ideals.illinois.edu

... All of them are free third-party tools designed to give better insight into the applications and packages. It provides more comprehensive information than **Steam** official API ... Like Youtube, Twitch is a large Page 2. online **game** steaming **platform** ...

☆ 99 》》

- 게임 별 특성과 매일 갱신되는 리뷰 수의 변동 사이의 관계
- 게임 리뷰의 유용한 정보들과 리뷰 작성 시 유저의 게임플레이시간과의 관계
- 모바일 앱의 리뷰와 스팀 리뷰와의 다른 양상을 알아봄.
- Future study 제시
  - Negative reviews: 버그 리포트 같이 게임을 개선할 수 있는 가치 있는 정보들을 포함.
  - 그렇다고 positive reviews에 담겨 있는 유용할 수 있는 리뷰들을 무시하면 안됨.



- Steam review: 내용과 상관없이 리뷰작성자들이 recommended와 not recommended를 선택하여 작성
  - 데이터 수집 과정에서 구분하여 수집
- LDA를 통해 topic modeling을 한 후, feature를 설명하는 단어 추출
  - Negative reviews: 게임 개선에 유용한 정보 추출
  - Positive reviews 에서도 유용할 수 있는 정보 추출



## 연구 의의

특정 게임의 긍정 및 부정 리뷰를 분석하여 가중치연산을 수행하여 감정 사전을 구축하고, 토픽에 적용하여 의미 있는 토픽을 찾음.

이를 통해 긍정과 부정 리뷰 모두 게임이 유지해야 할 방향과 개선해야 할 주요 이슈를 담고 있다는 것을 알아봄.

## 연구 방법

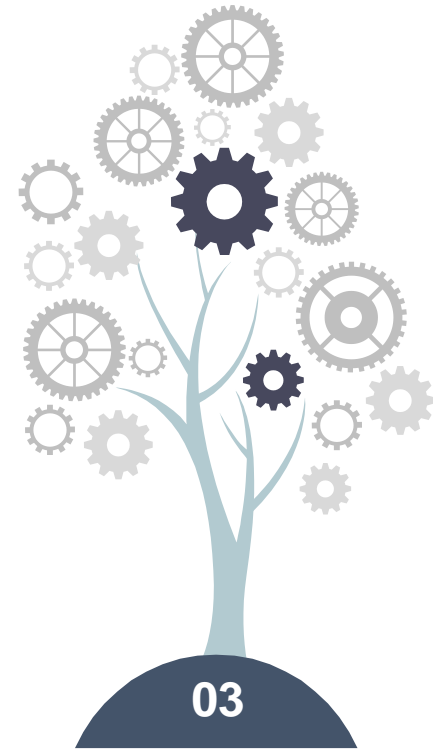
**01** 토픽 모델링을 통한 특정 게임의 주요 이슈 파악



**02** 가중치를 통한 감정 사전 구축



**03** 각 토픽에 감정 사전 적용하여 중요도 파악

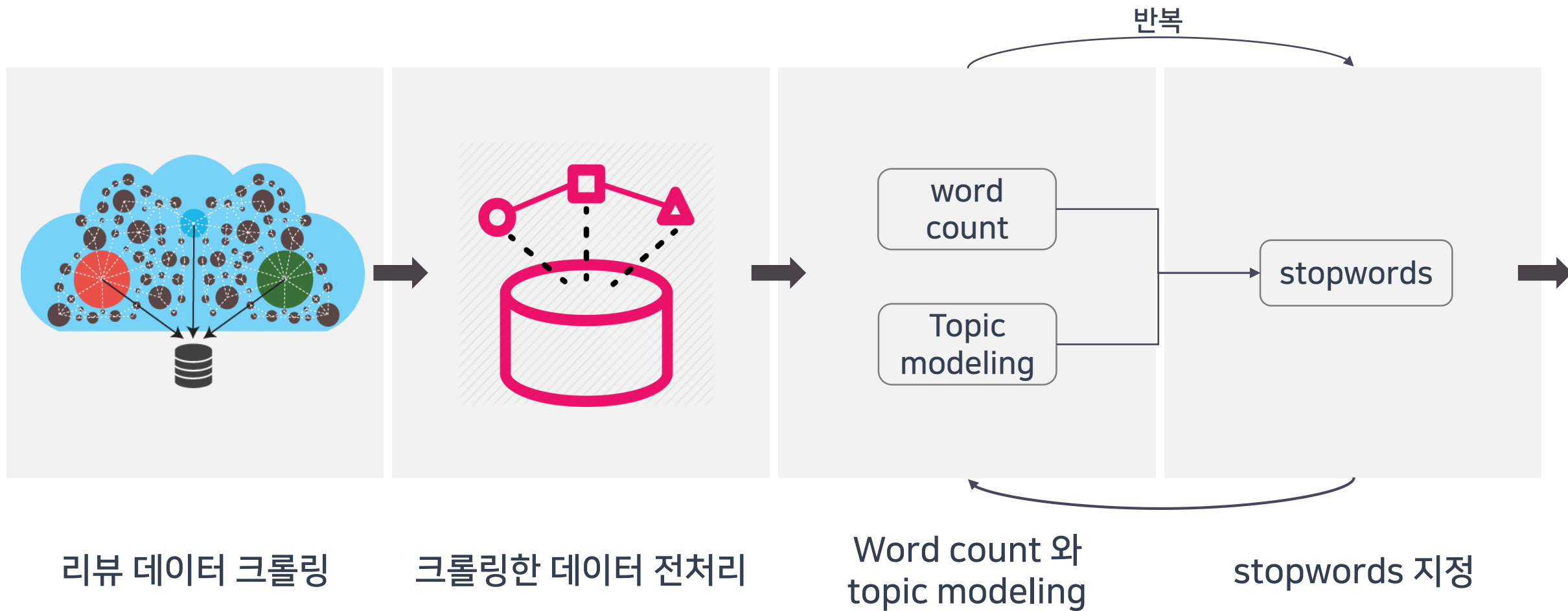




## Process



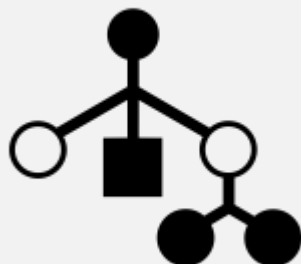
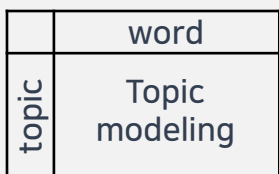
## 연구 프레임 워크







## 연구 프레임 워크



topic modeling 결과  
Topic별로 단어 구분

Naïve Bayes를 이용해  
단어의 가중치 계산

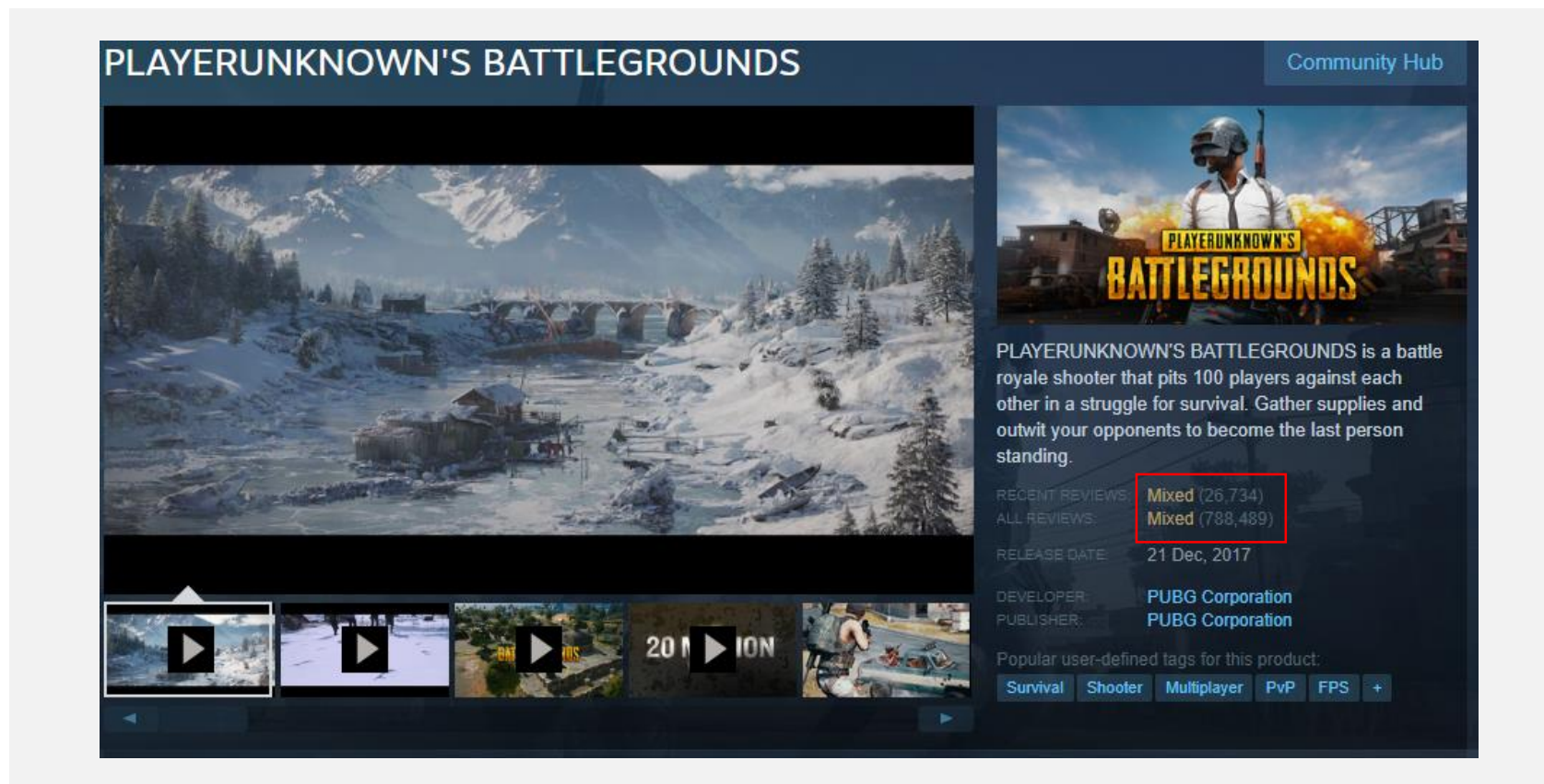
Naïve Bayes 가중치와  
topic modeling의 단어를  
이용하여 감성 사전 구축

topic별 topic score  
계산

Data



## 사용한 리뷰 데이터

A screenshot of the Steam store page for the game 'PLAYERUNKNOWN'S BATTLEGROUNDS'. The page features a large landscape image of a snowy mountain range with a bridge and a small building. To the right, there is a smaller image of a player character in a white uniform and helmet. Below these images, the game's description is provided: 'PLAYERUNKNOWN'S BATTLEGROUNDS is a battle royale shooter that pits 100 players against each other in a struggle for survival. Gather supplies and outwit your opponents to become the last person standing.' The review section shows 'RECENT REVIEWS: Mixed (26,734)' and 'ALL REVIEWS: Mixed (788,489)', with the word 'Mixed' highlighted in red. The release date is listed as '21 Dec, 2017', and the developer and publisher are both 'PUBG Corporation'. At the bottom, there are tags for 'Survival', 'Shooter', 'Multiplayer', 'PvP', 'FPS', and a plus sign for more tags. A 'Community Hub' link is visible in the top right corner.

PLAYERUNKNOWN'S BATTLEGROUNDS

Community Hub

PLAYERUNKNOWN'S BATTLEGROUNDS is a battle royale shooter that pits 100 players against each other in a struggle for survival. Gather supplies and outwit your opponents to become the last person standing.

RECENT REVIEWS: **Mixed** (26,734)  
ALL REVIEWS: **Mixed** (788,489)

RELEASE DATE: 21 Dec, 2017

DEVELOPER: PUBG Corporation  
PUBLISHER: PUBG Corporation

Popular user-defined tags for this product:  
Survival Shooter Multiplayer PvP FPS +

Steam의 'PLAYERUNKNOWN'S BATTLEGROUNDS'

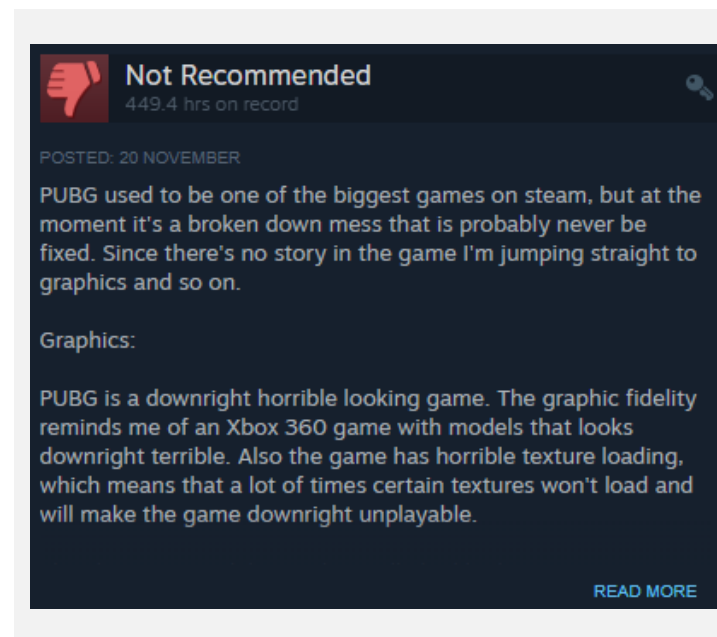
Data



## 사용한 리뷰 데이터



Positive



Negative

positive와 negative review 각각 10만개

# Programming



## 리뷰 데이터 크롤링

```
import requests
import re

from bs4 import BeautifulSoup

import csv
import time

j = 0
pos_review = []
for i in range(1, 10001):
    json_url = 'https://steamcommunity.com/app/578080/homecontent/?userreviewsoffset={0}'
    i += 1
    j += 10

    html = requests.get(json_url).text
    soup = BeautifulSoup(html, 'html.parser')

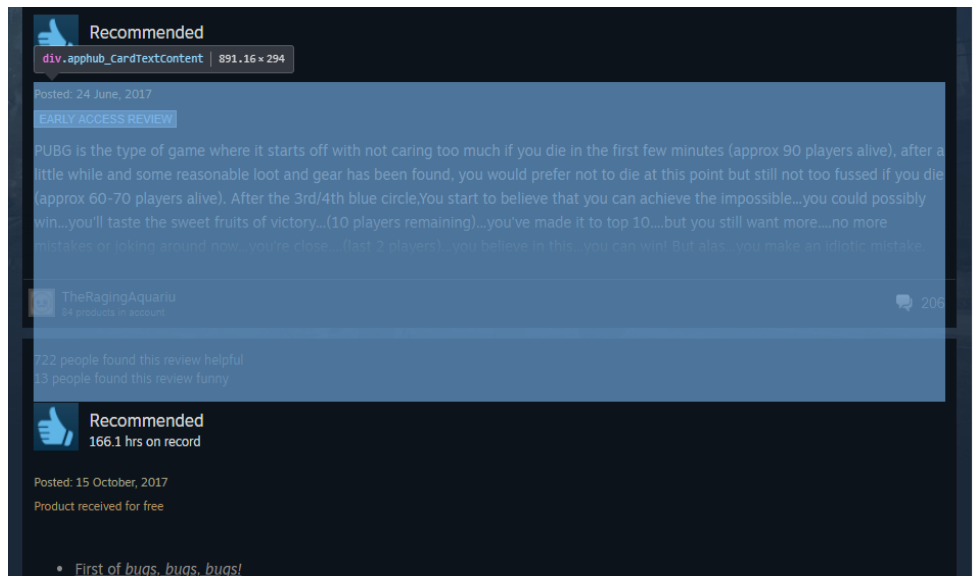
    for div in soup.find_all("div", {'class': "date_posted"}):
        div.decompose()

    for div in soup.find_all("div", {'class': "early_access_review"}):
        div.decompose()

    bus = soup.select('div[class=apphub_CardTextContent]')

    for text in bus:
        txt = text.get_text()
        pos = str(txt)
        pos = pos.strip()
        pos = pos.replace("<br>", '')
        pos = pos.replace("<div>", '')
        pos_review.append(pos)

    if j % 5 == 0:
        time.sleep(10)
    else:
        continue
```



```
left; width: 911.156px; height: 345px; opacity: 1;" data-modal-content-
url="https://steamcommunity.com/id/TheRagingAquariu/recommended/578080/"
data-modal-content-sizeToFit="false">
  <div class="apphub_CardContentMain" style="height: 287px;">
    <div class="apphub_UserReviewCardContent">
      <div class="vote_header"></div>
      <div class="found_helpful"></div>
      <div class="apphub_CardTextContent">
        <div class="date_posted">Posted: 24 June, 2017</div>
        <div class="early_access_review">Early Access Review</div>
        <br>
        "
        PUBG is the type of game where
        it starts off with not caring too much if you die in the first
        few minutes (approx 90 players alive), after a little while and
        some reasonable loot and gear has been found, you would prefer
        not to die at this point but still not too fussed if you die
        (approx 60-70 players alive). After the 3rd/4th blue circle,You
        start to believe that you can achieve the impossible...you could
        possibly win...you'll taste the sweet fruits of victory...(10
        players remaining)...you've made it to top 10....but you still
        want more....no more mistakes or joking around now...you're
        close....(last 2 players)...you believe in this...you can win!
        But alas...you make an idiotic mistake. You end as #2.....that
        moment will haunt you forever..."
        <br>
        <br>
        "This is what will happen every game, you'll feel rage, anger
        and above all serenity when you die because you know that the
        game is over....until you start the next one. "
        <br>
        <br>
        "Welcome to the battleground. "
      </div>
```

Review page의 review가 담겨있는  
<div class="apphub\_CardTextContent"> 부분을 크롤링



## 리뷰 데이터 전처리

```
list_par = []  
for i in data_list:  
    i = str(i)  
    text = re.sub('[^a-zA-Z0-9!?]', ' ', i).strip()  
    text = re.sub('[,]', '', text)  
    text = re.sub(' ', '', text)  
    if(text != ''):  
        if(text[0] != '?'):  
            list_par.append(text)
```

영어 및 숫자, 특수문자 ?, ! 만 남기고 제거

This game is like dating an alcoholic They're a lot of fun to be around and you legitimately like them But they have serious problems that ruin the whole relationship You keep thinking they'll change but they never will Instead they'll just try to sell you an Event Pass to see parts of their body

This game was killed by greedy and incompetent developers They have made A LOT of money out of us but they still refused to fix their game they kept releasing broken updates while turning a blind eye on actual problems such as rampant cheating They said there would be no micro transactions before the game left EA they lied They said the game left early access they lied it's still not as optimized as a game this big should be They are suing Epic Games because they are seeing a dip into their profits and they are too self-absorbed to realize their own incompetence is turning players away not a better competitor This company is interested in creating more paid content they don't care about fixing what they already have now they don't care about you

전처리 제거



## Word count와 topic modeling

```
word_count = collections.Counter(words)
result = word_count.most_common(500)
print(len(word_count))
print(result)
```

```
111223
[('even', 16685), ('still', 14936), ('money', 13926), ('time', 13719), ('many', 10335), ('really', 9785), ('bad', 9531), ('lag', 9360), ('be
tter', 8814), ('new', 8091), ('chinese', 8066), ('region', 7566), ('early', 7565), ('lock', 7558), ('server', 7180), ('devs', 6133), ('regio
nlockchina', 6026), ('bug', 5987), ('map', 5974), ('access', 5764), ('free', 5526), ('ever', 5425), ('fortnite', 5267), ('cheating', 5189),
('never', 5084), ('china', 5052), ('also', 4922), ('actually', 4574), ('back', 4505), ('update', 4461), ('not', 4356), ('loot', 4189), ('not
hing', 4153), ('play', 4112), ('fps', 4061), ('right', 4055), ('review', 3974), ('well', 3843), ('gameplay', 3710), ('instead', 3580), ('fi
x', 3562), ('problem', 3506), ('worst', 3424), ('worse', 3421), ('making', 3416), ('optimization', 3352), ('pc', 3342), ('system', 3303),
('trash', 3178), ('terrible', 3119), ('release', 3088), ('last', 3039), ('long', 3020), ('broken', 3004), ('running', 3000), ('point', 294
9), ('garbage', 2941), ('pretty', 2923), ('team', 2905), ('trying', 2902), ('reason', 2874), ('already', 2797), ('fixing', 2767), ('enough',
2746), ('almost', 2738), ('high', 2728), ('community', 2701), ('least', 2692), ('hackers', 2679), ('performance', 2672), ('hacker', 2657),
('always', 2595), ('experience', 2589), ('best', 2579), ('low', 2558), ('away', 2547), ('yet', 2486), ('match', 2467), ('far', 2457), ('chea
ter', 2426), ('buggy', 2396), ('shot', 2383), ('issue', 2372), ('real', 2361), ('ping', 2343), ('big', 2306), ('gun', 2297), ('lul', 2258),
('single', 2255), ('unplayable', 2215), ('too', 2202), ('buying', 2194), ('little', 2104), ('cheat', 2097), ('current', 2094), ('worth', 209
3), ('everything', 2088), ('buy', 2083), ('waste', 2079), ('too', 2066), ('content', 2051), ('idea', 2043), ('hard', 2037), ('end', 2028),
('please', 2016), ('poor', 2016), ('less', 1970), ('anyone', 1967), ('rather', 1960), ('laggy', 1959), ('literally', 1948), ('hacking', 193
8), ('shooting', 1919), ('ban', 1902), ('stupid', 1896), ('crash', 1879), ('maybe', 1872), ('cheaters', 1861), ('potential', 1858), ('work',
1842), ('probably', 1817), ('completely', 1804), ('random', 1797), ('poorly', 1784), ('sure', 1764), ('adding', 1738), ('boring', 1713), ('d
ecent', 1707), ('sometimes', 1698), ('developer', 1675), ('horrible', 1658), ('dev', 1626), ('concept', 1621), ('anti', 1620), ('crap', 160
4), ('desync', 1601), ('whole', 1590), ('using', 1587), ('huge', 1583), ('patch', 1576), ('else', 1567), ('squad', 1544), ('hack', 1538),
('honestly', 1538), ('extremely', 1528), ('just', 1528), ('however', 1508), ('crate', 1500), ('screen', 1499), ('absolutely', 1496), ('loadi
ng', 1479), ('care', 1473), ('anymore', 1472), ('working', 1467), ('car', 1464), ('hizl', 1452), ('version', 1447), ('death', 1441), ('weapo
```

word count

```
topic_vocab = c_vetorizer.get_feature_names()
topic_word = model.topic_word_
n_top_word = 30

for i, topic_dist in enumerate(topic_word):
    topic_words = np.array(topic_vocab)[np.argsort(topic_dist)][:-n_top_word:-1]
    print('Topic', i+1, topic_words)
```

```
Topic 1 ['still', 'even', 'time', 'money', 'many', 'really', 'better', 'lag', 'chinese',
'early', 'devs', 'ever', 'new', 'china', 'making', 'access', 'lock', 'free', 'bug',
'back', 'cheating', 'play', 'never', 'worst', 'nothing', 'terrible', 'system',
'community', 'not']
Topic 2 ['map', 'server', 'hackers', 'update', 'worse', 'trash', 'optimization', 'screen',
'loot', 'regionlockchina', 'review', 'poorly', 'team', 'everything', 'idea',
'best', 'instead', 'running', 'luck', 'awful', 'rather', 'simply', 'life',
'laggy', 'english', 'little', 'already', 'content', 'hard']
Topic 3 ['bad', 'region', 'trying', 'desync', 'absolutely', 'enough', 'sometimes',
'gameplay', 'anyone', 'low', 'cheater', 'wall', 'ban', 'friend', 'long',
'playable', 'fortnite', 'quality', 'absolute', 'performance', 'maybe', 'load',
'cover', 'match', 'shooter', 'random', 'unplayable', 'negative', 'plane']
Topic 4 ['loot', 'potential', 'worth', 'mode', 'wrong', 'different', 'death', 'soon',
'chance', 'refund', 'kind', 'world', 'waste', 'enemy', 'huge', 'lul', 'support',
'mess', 'gameplay', 'side', 'br', 'core', 'engine', 'old', 'biggest', 'driving',
'run', 'mobile', 'ping']
Topic 5 ['completely', 'horrible', 'buying', 'squad', 'issue', 'hell', 'using', 'else',
'looting', 'actual', 'sure', 'crashing', 'waste', 'sound', 'random', 'seriously',
'crash', 'bullet', 'now', 'account', 'impossible', 'care', 'zone', 'round',
'constantly', 'alpha', 'around', 'couple', 'anyone']
Topic 6 ['well', 'map', 'reason', 'gun', 'optimization', 'worse', 'literally', 'match',
'hacking', 'crap', 'probably', 'lul', 'hit', 'circle', 'however', 'running',
'damn', 'hizl', 'best', 'anymore', 'crate', 'building', 'developer', 'honestly',
'crash', 'rather', 'china', 'region', 'wait', 'long']
Topic 7 ['many', 'better', 'bad', 'regionlockchina', 'also', 'still', 'region', 'never',
'not', 'cheat', 'shot', 'single', 'just', 'low', 'top', 'cash', 'killing',
'current', 'adding', 'level', 'god', 'matter', 'boring', 'kill', 'content',
'major', 'everywhere', 'later', 'dead']
```

Topic modeling



## stopwords

```
stop_words = ['game', 'good', 'fun', 'pubg', 'ng', 'great',  
un_stopwords = np.unique(stop_words)  
un_stopwords = un_stopwords.tolist()
```

stopwords 지정

```
words = []  
for i in nnp_list:  
    # if(i!='?' and i!='!'):  
    text = str(i)  
    i = text.lower()  
    #words.append(i)  
    if(i not in un_stopwords):  
        words.append(i)  
  
print(words[0:10])  
len(words)
```

stopwords 제거





## Naïve Bayes

```
1 model = NaiveBayesClassifier()
2 model.train(nb_data)

1 print(len(model.word_probs))
176971

1 dict_p = {}
2
3 for i in model.word_probs:
4     dict_p[i[0]] = i[1]
5
6 dict_n = {}
7
8 for i in model.word_probs:
9     dict_n[i[0]] = i[2]
10
11 print(len(dict_p))
12 print(len(dict_n))
```

Naïve Bayes result  
Positive and negative words

```
1 import operator
2 sort_dict_p = sorted(dict_p.items(), key=operator.itemgetter(1), reverse=True)
3 sort_dict_n = sorted(dict_n.items(), key=operator.itemgetter(1), reverse=True)
4

1 for i in sort_dict_p[0:1700]:
2     print(i)

('good', 0.3220696321727971)
('fun', 0.27087048549190845)
('great', 0.1825052915784737)
('best', 0.1574557090715155)
('better', 0.13058949017516375)
('battle', 0.09660672322127964)
('product', 0.09049015849735838)
('nice', 0.07602373293778437)
('royale', 0.06630722821286313)
('hizl', 0.061757304044932584)
('pretty', 0.06142397626706222)
('bad', 0.060473992100131664)
('well', 0.059224012933117784)
('getting', 0.05795736737721038)
('map', 0.05737404376593724)
('amazing', 0.05382410293161781)
('fps', 0.0528241195980067)
('lag', 0.051140814319761335)
('never', 0.04344094265095582)
('right', 0.04287428542857619)
```

probability



## Sentiment Dictionary

```
# 긍정 토픽
ps_comment = ['good', 'fun', 'great', 'really', 'better', '
ps_charctor = ['battleroyale', 'competetive', 'real', 're
ps_style = ['battle', 'fps', 'royale', 'gameplay', 'combat
ps_feature = ['sniping', 'shot', 'map', 'bike', 'squad', 'i
ps_graphic = ['air', 'art', 'detail', 'night', 'weather',
ps_othergame = ['battlefield', 'dayz', 'hl2', 'hl', '
ps_another = ['twitch', 'wadu', 'youtube', 'streamer' ]
# 부정 토픽
ne_comment = ['bad', 'worst', 'waste', 'unreal', 'prob
ne_envoir = ['server', 'bug', 'lag', 'laggy', 'optimizatio
ne_another = ['chinese', 'region', 'regionlock', 'lock', '
# 토픽 모델링 단어에 대해 나이트베이지안 확률값 매핑
ps_comment_dic = {}
for topic in ps_comment:
    for word,value in dict_p.items():
        if(topic == word):
            ps_comment_dic[word] = value
```

Mapping Dictionary

```
1 print(ps_comment_dic)
2 print(ps_charctor_dic)
3 print(ps_style_dic)

{'good': 0.3220696321727971, 'fun': 0.27087048549190845,
155, 'pretty': 0.06142397626706222, 'amazing': 0.05382410
0.059224012933117784, 'goodgame': 0.0006749887501874968,
473458775687, 'wow': 0.0037249379177013717, 'favourite':
'exciting': 0.00737487708538191, 'favorite': 0.0107581540
'interesting': 0.00705821569640506, 'excellent': 0.006574
{'battleroyale': 0.003391610139831003, 'competetive': 0.0
'addictive': 0.017591373477108714, 'hardcore': 0.00310820
'faster': 0.005808236529391177, 'pvp': 0.007658205696571
{'battle': 0.09660672322127964, 'fps': 0.052824119598006
79032016134, 'war': 0.002108298195030083, 'hide': 0.00250
unting': 0.0016916384726921219, 'hitting': 0.002358294020
016308061532307794, 'aim': 0.005708238196030066, 'aiming
307746, 'picking': 0.003591606806553224, 'kill': 0.01649
```

Dictionary



## Sentiment Analysis

```
words = doc.split()
data_pos = nltk.pos_tag(words)
words_nnp = [word for word, pos in data_pos if pos in ['NN', 'NNP', 'VBG', 'JJ']]
words = [w for w in words_nnp if not w in stopwords.words('english')]
# 각 토픽별 점수 계산
for word in words:
    for _, i in ps_comment_dic.items():
        if (word == _):
            comment += i
    for _, i in ps_charctor_dic.items():
        if (word == _):
            charctor += i
    for _, i in ps_style_dic.items():
        if (word == _):
            style += i

total = comment + charctor + style + feature + graphic + othergame + antoher
print('-----리뷰 감성 분석-----')
print('-----토픽별 점수-----')
print('게임평 : '+str(comment)+' | 게임성격 : '+str(charctor)+' | 게임스타일 : '+str(style)+' | 게임요소')
print('총 합계 : '+str(total))
print('-----')
```

Sentiment Score

```
1 po = 'PUBG is the type of game where it starts off with not caring too much if you die in the first
2 positvie_review(po)
```

```
-----리뷰 감성 분석-----
-----토픽별 점수-----
게임평 : 0.7059882335294411 | 게임성격 : 0 | 게임스타일 : 0.032241129314511424 | 게임요소 : 0.0573740
: 0 | 그외 : 0
총 합계 : 0.7956034066098897
-----
```

Result Score

# Result



## Topic Modeling: Positive Review

|   | Topic | Words   |
|---|-------|---|
| 1 | 게임평   | good, fun, great, really, better, best, pretty, amazing, awesome, early, well, goodgame (gg), fantastic, enjoyable, wow, favourite (favorite), happy, love, exciting, hilarious, adrenaline, interesting, excellent |
| 2 | 게임성격  | battleroyale, competitive, real, realistic, addictive, hardcore, strategic, military, faster, pvp, massive  |
| 3 | 게임스타일 | battle, fps, royale, gameplay, combat, random, war, hide, wearing, eating, hunting, hitting, loot, shooter, aiming, survival, running, picking, killing   |
| 4 | 게임요소  | sniping, shot, map, bike, squad, inventory, dinner, chicken, scope, box, biggest, multiple, customization, weapon, winner, clothing, fpp, circle, rating, zone  |
| 5 | 그래픽   | air, art, detail, night, weather, graphics, graphic, character  |
| 6 | 타게임   | battlefield, dayz, h1z1, h1, overwatch, csgo, cs, arma, fortnite  |
| 7 | 그 외   | twitch, wadu, youtube   |

# Result



## Topic Modeling: Negative Reviews

|   | Topic   | Words   |
|---|---------|---|
| 1 | 게임평     | bad, worst, waste, unreal, problem, trash, terrible, wrong, stupid, crash, ridiculous, garbage, refund, serious, anymore, never, sad, bye, boring, horrible |
| 2 | 게임 환경   | server, bug, lag, laggy, optimization, unplayable, matchmaking, waiting   |
| 3 | 요구 개선사항 | Chinese, region, regionlock, lock, regionlockchina, china, cheat, cheating, hack, dev, development, fixing  |

# Result



## Naïve Bayes: Positive & Negative

| Positive   | Negative   |
|--|--|
| good, fun, great, best, better, battle, product, nice, royale, h1z1, pretty, bad, well, getting, map, amazing, fps, lag, never, right, awesome, player, devs, server, optimization, money, person, full, bit, little | money, good, fun, lag, bad, better, chinese, lock, region, server ,regionlockchina, devs, bug, getting, map, full, great ,never, battle, china, player, fortnite, update, loot, fps, right ,fix, cheating, well, problem |

## Conclusion



## 결론

|        | LDA | Naïve Bayes |
|--------|-----|-------------|
| 도메인 지식 | O   | X           |

Lda: 개인의 domain 지식을 적용해 토픽을 추출  
Naïve bayes: Domain 지식을 적용하지 않고 단어를 나열

각 방법에 따라 단어의 중요도에 약간의 차이가 있어 조금씩 다른 결과를 보여준다.

```
1 po = 'PLAYERUNKNOWN BATTLEGROUNDS My review for this game in early access. So this is a battleroyale game. A match consists a game that  
2 positvie_review(po)
```

```
-----리뷰 감성 분석-----  
-----토픽별 점수-----  
게임평 : 2.870035499408343 | 게임성격 : 0.020066332227796205 | 게임스타일 : 0.06275728737854369 | 게임요소 : 0.17148047532541125 | 그레  
픽 : 0 | 타게임 : 0 | 그외 : 0  
총 합계 : 3.124339594340094
```

```
1 ne = 'The developer is responsible for the massive amounts of cheating and racism that occurs on  
2 negative_review(ne)
```

```
-----리뷰 감성 분석-----  
-----토픽별 점수-----  
게임평 : 0.15325577907034882 | 게임환경 : 0.11705638239362677 | 그외 : 0.8321777970367161  
총 합계 : 1.1024899585006918
```

Topic modeling에 sentiment analysis를 적용시킨 결과

- positive: 게임스타일, 게임요소에 치중
- Negative: 게임환경, 요구개선사항에 치중



## Conclusion



## 결론

| positive   |   | negative                                 |
|--|---|--|
| Realistic, real                                  | ➡ | Optimization, money, unplayable          |
| Massive multiplayer, survival battleroyal, squad | ➡ | Server, waiting, lag, laggy, matchmaking |
|  |   | Chinese, region, lock, hack, cheating    |
| Customization: weapon, cloth, box, military      |   |  |
| fortnite   | ↔ | fortnite                                 |

- weapon, cloth 등을 customize 가능.
- 1등을 했을 때, chicken dinner 라는 캐치프라이즈가 '오늘은 치킨이닭'이라는 유행어가 되어 홍보효과 극대화.
- Biggest map 과 circle zone이라는 조합의 성공.
- 1인칭 및 3인칭 시점의 최초의 massive multiplayer (100명) 게임으로, survival battleroyal이라는 신선함의 성공.

- massive multiplayer 라는 장점에서 나오는 단점
  - 대규모인원으로 인한 server의 용량 문제
  - 다른 player들과 match하는데 긴 waiting 시간
- 게임이 유료이면서 최적화가 되지 않아 발생하는 단점
  - 게임을 구동시키기 위해 컴퓨터의 spec을 맞추는 데 money가 필요
  - 여기서 unplayable이라는 negative 단어가 나옴
- china의 game player들이 hack을 너무 많이 사용하여 gameplay에 cheating을 해 다른 player들에게 불편을 끼침
  - china region의 접속을 lock 해달라는 리뷰가 많이 발생

## Conclusion



## 결론

| positive   |   | negative                                 |
|--|---|--|
| Realistic, real                                  | ➡ | Optimizaton, money, unplayable           |
| Massive multiplayer, survival battleroyal, squad | ➡ | Server, waiting, lag, laggy, matchmaking |
|  |   | Chinese, region, lock, hack, cheating    |
| Customization: weapon, cloth, box, military      |   |  |
| fortnite   | ↔ | fortnite                                 |

- Battleground
  - 현실적인 디자인 및 게임 요소

- Battleground
  - 현실적인 디자인으로 인한 고성능 computer 필요
- fortnite
  - 최적화 잘 되어있음
  - 게임이 무료
  - 누구에게나 친숙하게 다가갈 수 있는 카툰형 식 디자인