



빈도분석

최 석 재

lingua@naver.com

빈도분석 사례

신한은행의 빅데이터 War Room

- 지역별 점포실적과 고객현황 등을 실시간으로 확인
- 870곳 점포, 2400만명의 고객 정보 활용
- 방문 고객 수와 대기 시간을 요일, 시간대, 날씨별로 파악
→ “A 점포는 ○요일, ○시간대가 붐비니 인근 B점포 이용”
- 입출금 메모로 소비트렌드 파악
→ 정수기 렌탈 → 렌탈비 포인트 환급
- P&G의 Business Sphere를 벤치마킹
→ 신제품 출시주기 ½, 시가총액 2배 상승



아모레의 빅데이터 고객 세분화

- 멤버십 카드의 고객 정보와 구매 내역을 통해 고객의 성향을 파악
- 고객의 특징에 따라 맞춤형 마케팅

고객 분류	특징
아모레 골드족	구매금액 가장 높음, 방문판매로 구매, 지방 거주, 설화수 선호
아모레 대표족	구매금액 차상위, 수도권 거주, 헤라·아모레퍼시픽 선호
아모레 갈대족	다양한 채널에서 구매
도시형 뷰티헌터	젊은 층, 온라인 구매, 다양한 브랜드 사용
도시형 알뜰족	기초제품 구매, 대도시 거주, 중저가 브랜드 선호
잇걸	젊은 층, 대도시 거주, 백화점 구매, 아모레 충성도 낮음
마이너리그	화장솜 같은 소도구나 할인 행사 제품 을 주로 구매
이니스프리족	이니스프리 외 다른 제품은 거의 쓰지 않음

Hertz의 고객불만 파악

- 자동차 렌탈업체 Hertz는 전 세계 8,000여 지점으로 수신되는 이메일, 콜센터 상담 내역, 홈페이지 게시물을 분석
- 불만 내용이 담긴 글에서 키워드를 추출하고,
- 지점 - 서비스 - 고객불만 의 연관성을 파악하여 서비스 개선
- 어떤 지역에서 불만 고조되는지 파악
 - 주된 불만 : 차량 반납에 걸리는 시간
 - 하루 중 어떤 시간에 지연이 발생하는지 파악
 - 해당 시간 직원 수 조정
 - 지점 매니저 재배치
 - 수익률 증가, 만족도 향상



빅데이터 기획연구 총서 17-03호

충북지역 대표 관광 콘텐츠 전략화 방안 제시를 위한 빅데이터 분석



2. 충북지역 대표 6개 관광지 방문 관광객 후기 텍스트 분석 결과

2.1 관광지별 주요 연관 키워드 분석 : 1) 단양팔경

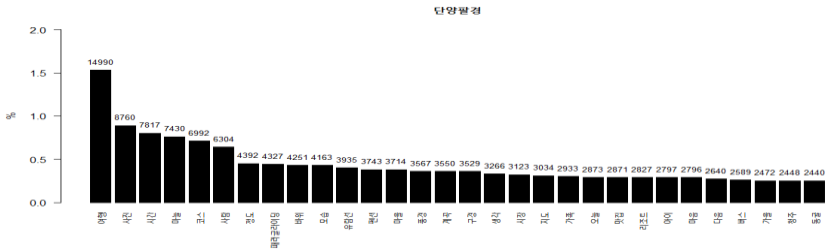
- 일반명사 분석



먹거리 - 마늘, 만두

즐길것 - 패러글라이딩, 유람선, 아쿠아리움

볼거리 - 계곡, 단풍



2. 충북지역 대표 6개 관광지 방문 관광객 후기 텍스트 분석 결과

2.1 관광지별 주요 연관 키워드 분석 : 정리

- 6개 관광지에서 모두 출현한 상위 빈도100 일반명사>
- 가을, 가족, 거리, 나무, 날씨, 느낌, 다음, 단풍, 마을, 마음, 모습, 문화, 사람, 사진, 생각, 시간, 식당, 아래, 아이, 아침, 여행, 오늘, 이름, 이번, 입구, 정도, 주변, 주차장, 지역, 처음, 체험, 친구, 코스, 풍경, 하늘, 호수

- | | |
|----------------------|--------|
| ▪ 가족, 아이, 친구 | 누구와 |
| ▪ 풍경, 나무, 단풍, 하늘, 호수 | 어떠한 것을 |
| ▪ 코스, 거리 | |
| ▪ 문화, 마을, 모습, 지역, 사람 | |
| ▪ 생각, 느낌, 마음 | |
| ▪ 체험 | |
| ▪ 식당 | 식사 |
| ▪ 주차장, 주변, 입구 | 도달방법 |

기타>
 다음, 이번, 정도, 처음
 아래
 이름
 가을, 날씨
 여행
 시간, 아침, 오늘
 사진

2. 충북지역 대표 6개 관광지 방문 관광객 후기 텍스트 분석 결과

2.1 관광지별 주요 연관 키워드 분석 : 정리

- 아름다운 자연
- 트레킹 및 드라이브
- 지역 특유의 먹거리
- 문화행사와 체험행사
- 편의 시설 (숙박, 주차장, 카페)

“편안한 **힐링** & 가벼운 **체험**”

텍스트마이닝기법을 활용한 현재 시놉시스 분석 및 개선안 제안

경희대학교

설문조사

총 2,000명 대상 조사

선호하는 시놉시스의 특징 파악 위험
전국 남녀 3사 IPTV 사용자
온라인 서베이 (엠브레인 패널) 활용

성별			연령			사용 중인 IPTV			거주 지역		
여성			10대			KT올레			서울		
남성			20대			SK브로드밴드			부산		
			30대						대구		
			40대						인천		
			50대 이상						광주		
									대전		
									울산		
									경기도		
									강원도		
									충청북도		
									충청남도		
									전라북도		
									전라남도		
									경상북도		
									경상남도		
									제주도		
									세종		

2. 영화 선정 기준

설문 문항	5점 척도
1) 영화 시청 전에 시놉시스를 신중히 검토 한다.	3.36*
2) 영화 관람객 수나 예약률을 많이 참고한다.	3.09
3) 주변의 평가에 상관없이 내가 좋아하는 영화를 본다.	3.71**
4) 전문가의 영화평을 많이 참고한다.	2.86
5) 일반인의 영화평이나 댓글을 많이 참고한다.	3.20
6) 최근 정치, 사회와 관련된 시사성 있는 영화를 본다.	2.66
7) 특정 장르, 배우, 감독 또는 영화 형식 등을 집중적으로 본다.	3.11

* 시놉시스는 영화평, 댓글, 여타 메타정보보다 더 중요한 영화선정 기준임

** 개인 미디어를 활용한 선호도 분석 및 큐레이션 (true personalization) 필요

3. 영화 선택 시 중요도

성별 및 연령대와 상관없이

1. 장르
2. 시놉시스
3. 출연배우의 순

항목	전체	남자	여자	10대	20대	30대	40대	50대 이상
영화제목	3.42	3.47	3.37	3.27	3.47	3.35	3.44	3.44
시놉시스	3.79	3.91	3.68	3.73	3.84	3.70	3.82	3.82
제작국가	2.83	2.86	2.81	2.76	2.82	2.80	2.85	2.89
장르	3.95	4.01	3.90	4.04	4.04	3.91	3.94	3.93
시청료	3.64	3.70	3.59	3.68	3.75	3.69	3.66	3.45
영화감독	3.04	3.03	3.06	2.60	2.91	3.03	3.13	3.17
출연배우	3.72	3.80	3.64	3.48	3.75	3.67	3.78	3.70
영화평가	3.40	3.46	3.34	3.27	3.42	3.39	3.49	3.30
최신성	3.16	3.09	3.23	2.79	3.01	3.13	3.31	3.22
시청연령대	2.76	2.73	2.79	2.69	2.55	2.71	2.93	2.83
국내개봉일	2.68	2.64	2.71	2.44	2.47	2.64	2.82	2.77

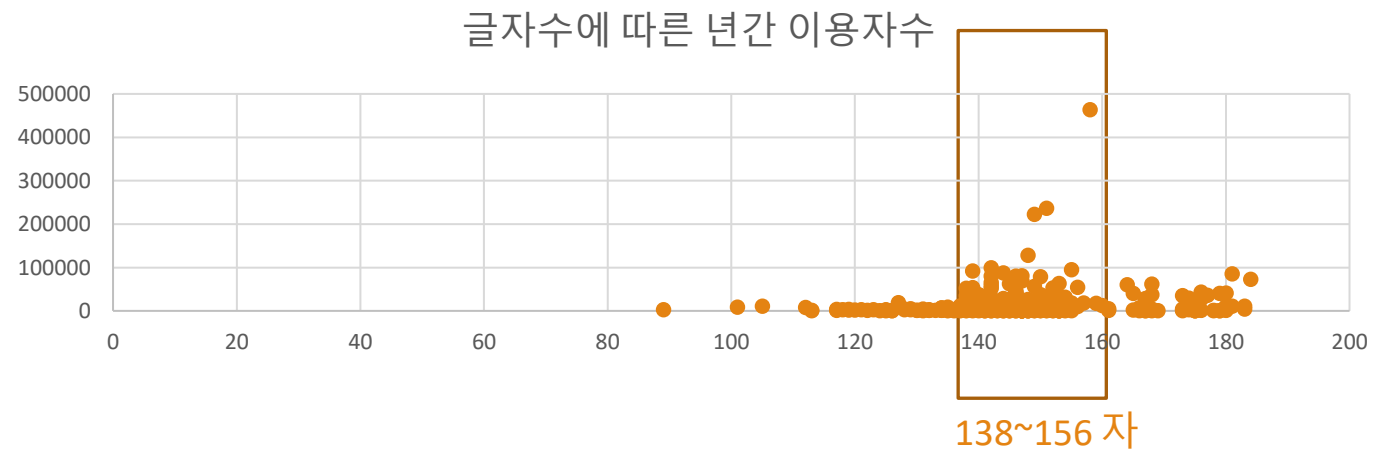
시놉시스 분석

1. 시놉시스 길이와 이용자수 관계

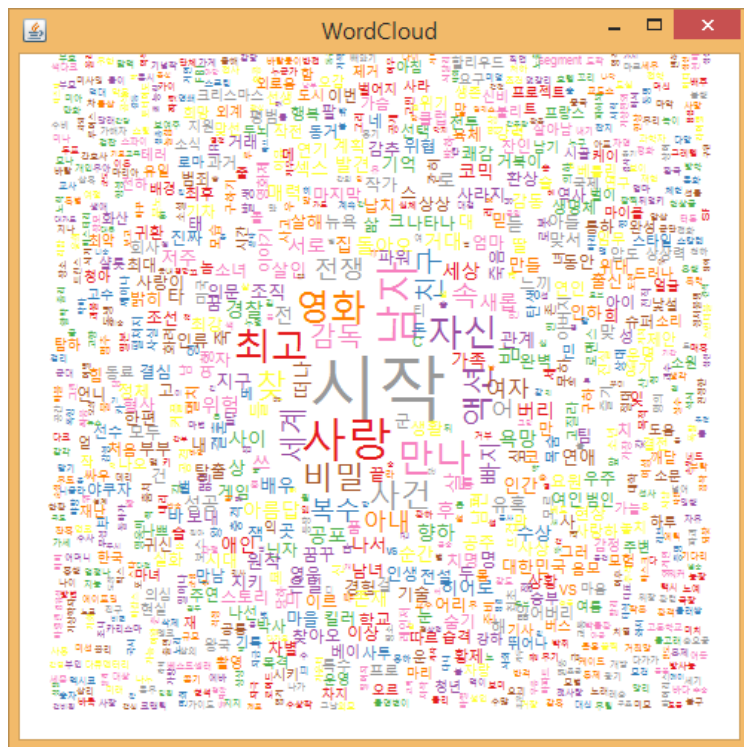
이용자수 1,000 이상의 영화 시놉시스의 평균 길이: 148 자

138 자 이상에서 이용자수가 많아졌으며, 156자까지에서 많은 시놉시스가 있음

따라서 시놉시스의 길이는 138~ 자 이상이면 되며,
관습적인 길이를 고려한다면 ~156 자 사이가 적절함



2. 상위 30% 영화에서 사용된 명사



빈도 (회)	명사
20 이상	시작, 사랑, 남자, 영화, 자신, 최고, 액션, 세계, 비밀, 사건, 친구, 감독, 전쟁, 사람, 복수, 여자, 밤, 아내, 세상
10~19	관계, 살인, 조직, 가족, 지구, 집, 사이, 경찰, 거대, 위험, 일, 욕망, 남편, 공포, 결혼, 죽음, 인간, 유혹, 순간, 상(賞), 완벽, 목숨, 이야기, 생활, 배우, 끝, 엄마, 탄생, 아버지, 의문, 탈출, 돈, 납치, 여행, 곳, 살해, 최강, 뉴욕, 소녀, 전설, 치명적, 남녀, 충격, 닌자, 마지막, 게임, 인생, 시간, 영웅, 인류, 연인, 여인, 공주, 기술, 고질라, 로맨스, 딸, 팀, 요원, 미국, 신(scene), 커플, 삶, 경험, 공격, 히어로, 연애, 모습, 원작, 매력, 코믹, 발견, 성공, 진실, 너, 로봇, 애인, 존재, 섹스, 발생, 영화제, 수상, 저주

3. 고빈도 명사 분류 (상위 30% 영화의 빈도 10 이상)

주제	어휘
사랑	사랑, 밤, 욕망, 유혹, 로맨스, 커플, 연애, 애인, 섹스
액션	액션, 전쟁, 복수, 살인, 경찰, 공포, 탈출, 납치, 살해, 영웅, 히어로, 팀, 요원, 공격
수식	최고, 거대, 위험, 완벽, 의문, 최강, 치명적, 충격, 저주, 매력, 코믹
이야기	비밀, 사건, 이야기, 전설, 게임, 진실
세계	세계, 세상, 조직, 지구
사람, 가족	사람, 남자, 여자, 여인, 남편, 아내, 인간, 엄마, 아버지, 소녀, 딸, 남녀, 연인, 인류, 자신, 친구, 관계, 사이, 공주, 가족, 집, 결혼
존재	존재, 죽음, 목숨, 생활, 탄생, 인생, 삶, 발생
영화	영화, 감독, 상(賞), 배우, 신(scene), 영화제, 원작, 수상
시간	시간, 시작, 끝, 마지막, 순간
캐릭터	넌자, 고질라
기타	일, 돈, 여행, 곳, 뉴욕, 기술, 미국, 경험, 모습, 발견, 성공, 로봇

4. 상위 30% 영화에서 사용된 동사

빈도 (회)	동사
20이상	위하다, 만나다, 찾다, 알다, 빠지다, 새롭다, 돌아오다, 펼치다
10~19	떠나다, 쓰다, 아름답다, 죽다, 향하다, 나서다, 느끼다, 나타나다, 크다, 꿈꾸다, 만들다, 믿다, 모르다, 뜨겁다, 사라지다, 구하다, 인하다, 잃다, 즐기다, 듣다, 은밀하다, 같히다, 화려하다
5~10	감추다, 지키다, 맞서다, 보내다, 밝히다, 어리다, 숨기다, 찾아오다, 반하다, 젊다, 잇다, 사랑하다, 따르다, 흠치다, 의하다, 생기다, 남다, 나오다, 위대하다, 깨어나다, 일어나다, 강력하다, 잃어버리다, 평범하다, 품다, 멈추다, 갖다, 강하다, 오르다, 작다, 놓치다, 뜨다, 시달리다, 나쁘다, 벌어지다, 드러나다, 돌아가다, 싸우다, 남기다, 뛰어나다, 탐하다, 가늘다, 낮설다, 바꾸다, 청아하다, 많다, 전하다, 막다, 얻다, 깨닫다, 살아남다, 불리다, 떨어지다, 벌이다

불용어(stopping words):

되, 있, 하, 없, 받, 았다, 가다, 보다, 살다, 오다, 말다, 싶다, 주다, 내다, 타다, 같다, 맞다, 당하다, 두다, 내리다, 통하다, 못하다, 대하다, 다니다, 이르다, 시키다

5. 고빈도 동사 분류 (상위 30% 영화의 빈도 5 이상)

주제	어휘
사랑	빠지다, 뜨겁다, 반하다, 사랑하다, 탐하다
액션	구하다, 맞서다, 흠치다, 싸우다
수식	새롭다, 아름답다, 화려하다, 위대하다, 강력하다, 평범하다, 강하다, 뛰어나다
이야기	펼치다, 꿈꾸다, 밝히다, 일어나다, 벌어지다, 듣다, 드러나다, 전하다, 벌이다
비밀	은밀하다, 감추다, 숨기다
사람	어리다, 젊다, 청아하다
존재	죽다, 생기다, 깨어나다, 살아남다, 나타나다, 사라지다
비교	크다, 작다, 가늘다, 많다, 나쁘다
인식	알다, 잊다, 낯설다, 깨닫다, 느끼다, 불리다
만남	만나다, 찾다, 돌아오다, 떠나다, 향하다, 나서다, 나오다, 보내다, 찾아오다, 따르다, 돌아가다
소유	잃다, 잃어버리다, 지키다, 품다, 얻다, 막다, 남다, 놓치다
기타	위하다, 쓰다, 만들다, 믿다, 인하다, 의하다, 즐기다, 갇히다, 멈추다, 갖다, 오르다, 뜨다, 시달리다, 남기다, 바꾸다, 떨어지다

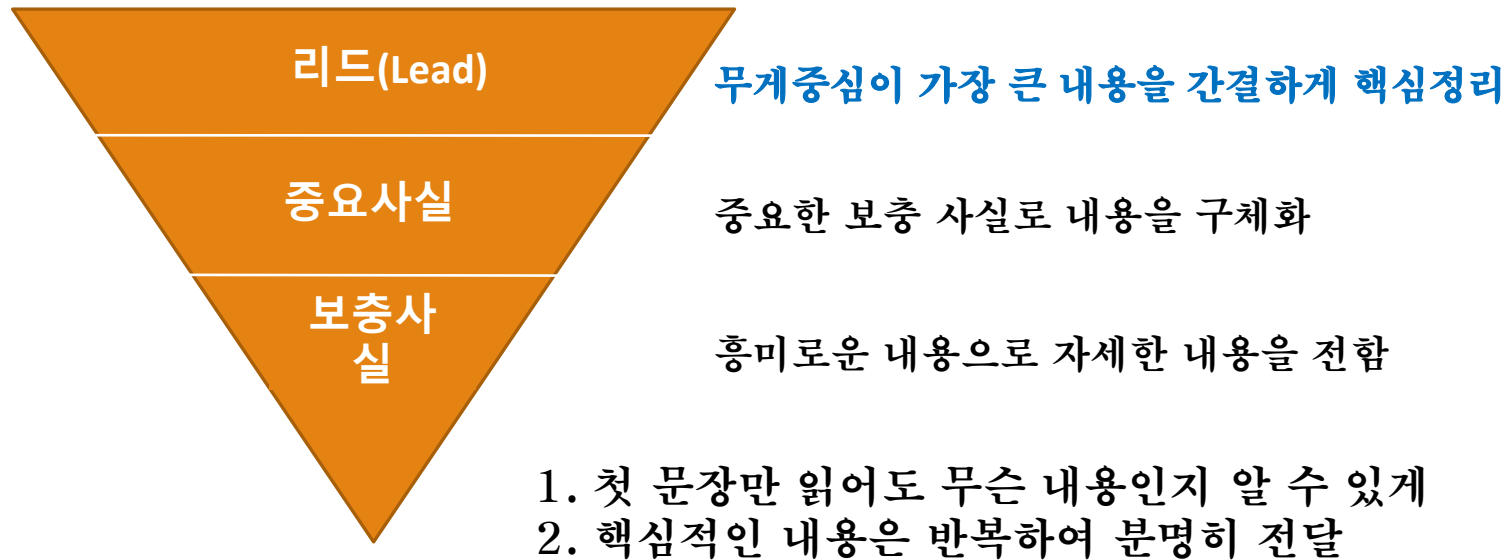
성과 증진을 위한 시놉시스 개선안

1. 시놉시스의 길이와 단어 최적화

- 전체 시놉시스의 길이는 **138자 ~ 156자** 사이가 적합
 - 너무 짧으면 줄거리를 파악하기 어렵고
 - 너무 길면 가독성이 떨어짐
- 단어는 가급적 상위 30% 영화의 빈도 5 이상의 어휘를 사용

2. Straight 기사 문장의 구조를 참조

- 시놉시스는 중요한 정보를 간결하게 전달한다는 점에서 신문 기사 유형 중 스트레이트와 유사



△스트레이트 기사(육하원칙에 따라 담백하고 객관적으로 사건을 전달하는 기사)
△르포 기사(사건이 벌어지는 현장을 스케치하는 기사)
△인터뷰 기사(인물을 취재해 그를 중심으로 쓰는 기사)
△해설성 기사(사건이 벌어진 원인이나 배경 등을 전문가들의 분석을 바탕으로 설명하는 기사)

3. (리드 이후의) 본문은 신화 구조의 핵심 요소를 참조

- 신화가 갖는 핵심 요소를 시놉시스에 반영하여 시청자의 관심을 유도함

구조	단계	핵심 요소
발단	1	일상 세계의 평범한 주인공
전개	2	미션을 받음
	3	미션을 거부
	4	멘토를 만남
	5	첫 관문 통과, 여정 시작
위기	6	시험, 적대자 만남
	7	매우 위험한 존재와 첫 만남
	8	깊은 시련
	9	보상 및 성장
절정	10	안락함을 포기, 귀환
	11	새롭게 부활 , 최대 난관과 싸우고, 이김
결말	12	새로운 존재가 되어 귀환

개선안



할리우드 액션의 새로운 역사 리암 니슨! [마이내리티 리포트][더 울버린]의 각본가 스콧 프랭크와 만나 업그레이드 된 추격 스릴러를 완성하다! 사립탐정 '맷'과 역대 최고 연쇄살인범들의 쫓고 쫓기는 극한의 추격전이 시작된다!



리암 니슨의 하드보일드 액션을 만난다! 외롭게 지내는 전직 형사 맷(리암 니슨)에게 어느날 아내의 복수를 해달라는 의뢰가 들어온다. 살인범이 잔인한 연쇄살인마임을 직감한 맷은 그들의 행적을 찾아나서고, 이 과정에서 묘지 관리인 루건을 알게 된다. 그리고 루건은 충격적인 이야기를 하는데...

요소	내용
LEAD	리암 니슨의 하드보일드 액션
주인공	외롭고 평범한 맷
미션	의뢰인 아내의 복수
적대자	잔인한 연쇄살인마
멘토	루건
성장	충격적 사실을 접하며 범인에 대해 점차 알아간다

시놉시스 개선안에 대한 효과 검증

시놉시스	개선안 준용 시 시놉시스 선호도 증가 (%)
시놉시스1	+25.5
시놉시스2	+18.5
시놉시스3	+46.0
시놉시스4	-0.5
시놉시스5	+5.5
시놉시스6	+13.0
시놉시스7	0.0
시놉시스8	+18.5
시놉시스9	+14.5
시놉시스10	+2.0

기존안과 개선안을 랜덤 하게 제시함
어느 것이 개선안인지 밝히지 않음
(블라인드 테스트)

영화 이미지와 같이 노출시킴

-100 (기존안 절대 선호) ~ 100 (개선안 절대 선호)
0.0은 중립

10건 중 8건에 대해
개선안을 더욱 선호

빈도분석 *Frequency Analysis*

네이버 영화 자료

- 깃헙에서 200,000 건의 영화평 데이터를 다운로드 받을 수 있다
- ratings.txt 파일만 받으면 된다 (pytest 폴더에 small 버전도 있음)
- <https://github.com/e9t/nsmc/>
- 인코딩을 UTF-8에서 ANSI로 바꾸어 엑셀에서 읽어보면 다음과 같다

	A	B	C
1	id	document	label
2	8112052	어릴때보고 지금다시봐도 재밌어요ㅋㅋ	1
3	8132799	디자인을 배우는 학생으로, 외국디자이너와 그들이 일군 전통을 통해 발전해가는 문화산	1
4	4655635	폴리스스토리 시리즈는 1부터 뉴까지 버릴게 하나도 없음.. 최고.	1
5	9251303	와.. 연기가 진짜 개쩔구나.. 지루할거라고 생각했는데 몰입해서 봤다.. 그래 이렇게 진짜 영	1
6	10067386	안개 자욱한 밤하늘에 떠 있는 초승달 같은 영화.	1
7	2190435	사랑을 해본사람이라면 처음부터 끝까지 웃을수 있는영화	1
8	9279041	완전 감동입니다 다시봐도 감동	1
9	7865729	개들의 전쟁2 나오나요? 나오면 1빠로 보고 싶음	1
10	7477618	굿	1
11	9250537	바보가 아니라 병 쉰 인듯	1
12	9730759	내 나이와 같은 영화를 지금 본 나는 감동적이다..하지만 훗날 다시보면대사하나하나 그	1
13	640794	재밌다	1

id

영화평 본문

금부정 표지: 1은 긍정(9~10), 0은 부정(1~4)

구글 드라이브와 연결

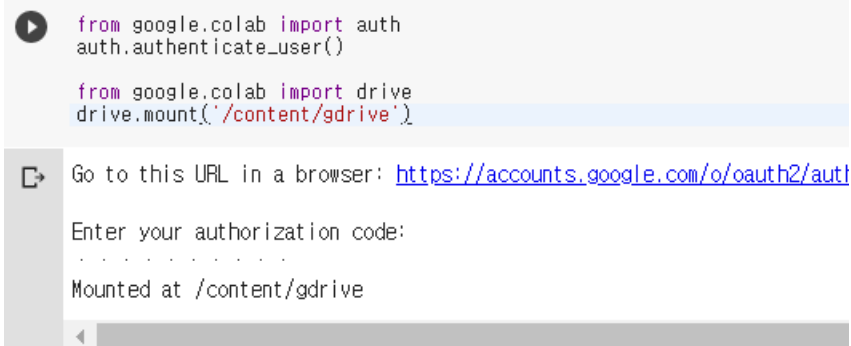
- 다음의 코드를 입력한 뒤, 절차에 따라 진행한다
- verification code 입력 후, authorization code를 입력한다
- 일정 시간(약 3시간) 이후에는 끊기므로, 매번 이 작업을 해주어야 한다

```
# from google.colab import auth  
# auth.authenticate_user()
```

이 두 줄은 접속이 잘 안되는 경우에만

최종 화면

- from google.colab import drive
- drive.mount('/content/gdrive')



```
from google.colab import auth  
auth.authenticate_user()  
  
from google.colab import drive  
drive.mount('/content/gdrive')
```

Go to this URL in a browser: <https://accounts.google.com/o/oauth2/auth>

Enter your authorization code:
.....

Mounted at /content/gdrive

Colab에서의 폰트 설정

- `import matplotlib as mpl`
- `import matplotlib.pyplot as plt`
- `%config InlineBackend.figure_format = 'retina'` # 폰트가 깨끗하게 보이도록 설정
- `!apt install fonts-nanum` # 나눔 폰트 설치
- `import matplotlib.font_manager as fm`
- `fontpath = '/usr/share/fonts/truetype/nanum/NanumMyeongjo.ttf'`
- `font = fm.FontProperties(fname=fontpath, size=9)`
- # 기본 글꼴 변경
 - `import matplotlib as mpl`
 - `mpl.font_manager._rebuild()`
 - `mpl.pyplot.rc('font', family='NanumMyeongjo')`

※ NanumGothic과 NanumMyeongjo
중 하나를 선택한다

※ 인식이 잘 안 되었을 가능성이 높으므로
"런타임 > 런타임 다시 시작" 수행 후
여기부터 다시 진행한다

<https://blog.naver.com/choeungjin/221460930665> 참조

Local PC에서의 폰트 설정

Local PC에서는 다음의 방법으로 한글폰트를 설정한다

```
import matplotlib
from matplotlib import font_manager, rc
font_path = 'C:/Windows/Fonts/malgun.ttf'
font_name = font_manager.FontProperties(fname=font_path).get_name()
matplotlib.rc('font', family=font_name)
```

형태소분석기 설치

- !apt-get update
- !apt-get install g++ openjdk-8-jdk
- !pip install JType1
- !pip install rhinoMorph

※ ! 는 사실 사용하지 않아도 되지만, 이 명령어를 넣어주어야 실행이 되는 경우가 있다

경로 변경

파일이 있는 곳으로 경로를 변경한다

- `!cd /content/gdrive/My Drive/pytest/`

※ 별도의 코드 셀에서 진행해야 한다

※ 파일이 잘 읽어지지 않으면

- 가장 처음의 구글 드라이브와의 연결을 다시 한다(`drive.mount('/content/gdrive')`)
- '!'와 'w'는 떼어도 보고, 붙여도 본다 (colab 문제)
- 그래도 안되면 '!'를 '%'로 바꾸어본다
- 그래도 안되면 `data = read_data('/content/gdrive/My Drive/pytest/ratings_small.txt', encoding='cp949')`
- 그래도 안되면 `!pip install -U -q PyDrive` 를 실행해본다

※ PyCharm에서 진행한다면 아래의 코드로 경로를 변경한다

```
import os
os.chdir("C:/pytest")
```

데이터 로딩

```
def read_data(filename, encoding='cp949'): # 읽기 함수 정의
    with open(filename, 'r', encoding=encoding) as f:
        data = [line.split('Wt') for line in f.read().splitlines()]
        data = data[1:] # txt 파일의 헤더(id document label)는 제외하기
    return data # splitlines() → 리스트 생성 → 전체로는 중첩리스트

def write_data(data, filename, encoding='cp949'): # 쓰기 함수도 정의
    with open(filename, 'w', encoding=encoding) as f:
        f.write(data)

data = read_data('ratings_small.txt', encoding='cp949') # 전체파일은 ratings.txt
```

※ 한글 인코딩

- euc-kr (메모장에서는 ANSI로 표현)
- cp949 (=MS949, euc-kr의 확장판)
- utf-8

※ 파일의 인코딩은 메모장 또는, 아래의 코드로 확인할 수 있다

```
import chardet
f = open("C:/pytest/ratings_small.txt", "rb").read()
encoding = chardet.detect(f)
print(encoding)
```

데이터 확인

```
print(len(data))  
print(len(data[0]))  
print(data[0])  
print(data[0:3])
```

500

3

['8112052', '어릴때보고 지금다시봐도 재밌어요ㅋㅋ', '1']

[['8112052', '어릴때보고 지금다시봐도 재밌어요ㅋㅋ', '1'], ['8132799', '"디자인을 배우는 학생으로, 외국디자이너와 그들이 일군 전통을 통해 발전

※ 탭으로 분리된 요소가 한 리스트의 원소가 되고,
각 문장은 하나의 리스트로서
전체가 다시 리스트로 묶인 중첩리스트이다

샘플 데이터 분석 연습

```
import rhinoMorph
rn = rhinoMorph.startRhino()

# 형태소 분석된 문장 샘플 보기. eomi=True 옵션 사용
sample_data = rhinoMorph.onlyMorph_list(
    rn, data[0][1], pos=['NNG', 'NNP', 'VV', 'VA', 'XR', 'IC', 'MM', 'MAG', 'MAJ'], eomi=True)

print('sample data:', sample_data)
sample data: ['어리다', '때', '보다', '지금', '다시', '보다', '재미있다', 'ㅋㅋ']

print('joined sample data:', ' '.join(sample_data))
joined sample data: 어리다 때 보다 지금 다시 보다 재미있다 ㅋㅋ
```

형태소 분석 결과

문자열을 공백으로 연결한다

※ 이제 영어와 유사한 상태가 되었다

전체 데이터 형태소 분석

```
morphed_data = ''
for data_each in data:
    morphed_data_each = rhinoMorph.onlyMorph_list(
        rn, data_each[1], pos=['NNG', 'NNP', 'VV', 'VA', 'XR', 'IC', 'MM', 'MAG', 'MAJ'], eomi=True)
    joined_data_each = ' '.join(morphed_data_each)           # 문자열을 하나로 연결
    if joined_data_each:                                     # 내용이 있는 경우만 저장하게 함
        morphed_data += data_each[0]+"Wt"+joined_data_each+"Wt"+data_each[2]+"Wn"

# 형태소 분석된 파일 저장
write_data(morphed_data, 'ratings_morphed.txt', encoding='cp949')
```

분석된 데이터 로딩

```
data = read_data('ratings_morphed.txt', encoding='cp949')
print(len(data))
print(len(data[0]))

data_text = [line[1] for line in data]
data_senti = [line[2] for line in data]
```

494 (일부는 내용이 남지 않아 제외 됨)
3개의 원소(컬럼)
데이터 본문
데이터 긍부정 부분 (1은 긍정, 0은 부정)

※ 방금 생성된 ratings_morphed.txt 파일을 이용한다

Counter 연습

- Counter는 리스트의 구성요소를 종류별로 빈도 계산한다

```
from collections import Counter
count = Counter(["여름", "과일", "봄", "딸기", "과일", "봄", "딸기", "봄"])
result = count.most_common(3)          # 가장 빈도가 높은 어휘 3개만
```

```
print("count: ", count)
print("result: ", result)
print("result[0]", result[0])
print("result[0][0]: ", result[0][0])    # 0 번째 요소의 0 번째
print("result[0][1]: ", result[0][1])    # 0 번째 요소의 1 번째
```

```
count: Counter({'봄': 3, '과일': 2, '딸기': 2, '여름': 1})
result: [('봄', 3), ('과일', 2), ('딸기', 2)]
result[0] ('봄', 3)
result[0][0]: 봄
result[0][1]: 3
```

빈도 구하기 - 단어 분리

```
data_text_freq = Counter(data_text)
print('data_text:', data_text_freq)
```

안 좋은 방법. 한 문장 단위로 나뉜 리스트를 바로 처리 시도
데이터를 바로 카운트하면 각 줄이 하나의 종류로 카운트된다

```
mergedText = ' '.join(data_text)
print('mergedText:', mergedText)
```

좋은 방법. 공백을 추가하며 일단 모든 리스트 요소들을 결합한다

```
test = ['디자인 배우 학생', '앞서 나 가 학생']
test2 = ' '.join(test)
test2
```

'디자인 배우 학생 앞서 나 가 학생'

```
mergedTextList = mergedText.split(' ')
print('mergedTextList:', mergedTextList)
```

결합된 요소들을 공백 단위로 분리하여 하나의 리스트로 만든다

```
data_text: ['디자인 배우다 학생 외국 디자이너 일구다 전통 통하다 발전 문화 산업 부럽다 사실 우리나라 그 어렵다 시절 끝']
mergedText: 디자인 배우다 학생 외국 디자이너 일구다 전통 통하다 발전 문화 산업 부럽다 사실 우리나라 그 어렵다 시절 끝
mergedTextList: ['디자인', '배우다', '학생', '외국', '디자이너', '일구다', '전통', '통하다', '발전', '문화', '산업',
```

빈도 구하기 - 불용어 제거

실질형태소 중 불용어 등록 및 제거 (국립국어원 말뭉치 중 고빈도 어휘)

- stopwords_ko = ["하다", "있다", "되다", "그", "않다", "없다", "나", "말", "사람", "이", "보다", "한", "때", "년", "같다", "대하다", "일", "이", "생각", "위하다", "때문", "그것", "그러나", "가다", "받다", "그렇다", "알다", "사회", "더", "그녀", "문제", "오다", "그리고", "크다", "속"]
- mergedTextList_no_stopwords = [word for word in mergedTextList if not word in stopwords_ko]
- print('mergedTextList without stopwords:', mergedTextList_no_stopwords)
- print('불용어 제거 전 길이:', len(mergedTextList))
- print('불용어 제거 후 길이:', len(mergedTextList_no_stopwords))
- mergedTextList = mergedTextList_no_stopwords # 변수명 대체

```
mergedTextList without stopwords: ['디자인', '배우다', '학생', '외국', '디자이너', '일구다',  
불용어 제거 전 길이: 4415  
불용어 제거 후 길이: 3832
```

※ 불용어 목록은 필요에 따라 적절히 수정한다

빈도 구하기 – Counter 사용

```
wordInfo = Counter(mergedTextList)          # 하나의 리스트로 묶인 분리된 요소들을 카운트한다 (내림차순)  
print('wordInfo:', wordInfo)
```

```
wordInfo: Counter({'영화': 175, 'ㅋㅋ': 60, '정말': 47, '너무': 47, '좋다': 43, '재미있다': 36, '진짜': 33, '연기': 30, '만들다': 28,
```

※ 빈도는 구해졌으므로 이제 그래프로 보기 좋게 표현한다

sorted 연습

```
sample = {'여름':1, '과일':2, '딸기':3}
```

```
print(sorted(sample))
```

```
print(sorted(sample, reverse=True))
```

역순으로 정렬

```
print(sorted(sample, key=sample.get, reverse=True))
```

sample.get의 출력된 값을 기준으로 sample을 정렬

```
print(sorted(sample.values(), reverse=True))
```

값 부분만 출력하여 정렬

```
['과일', '딸기', '여름']
```

```
['여름', '딸기', '과일']
```

```
['딸기', '과일', '여름']
```

```
[3, 2, 1]
```

※ sorted 함수는 새로운 리스트 결과를 만들어내나,
sort 함수는 원 데이터를 변경한다. 사용법도 다르다

```
mylist = [5, 7, 2, 3, 1]
```

```
print(sorted(mylist))
```

```
print(mylist.sort())
```

```
print(mylist)
```

[1, 2, 3, 5, 7]. 새로운 리스트 출력

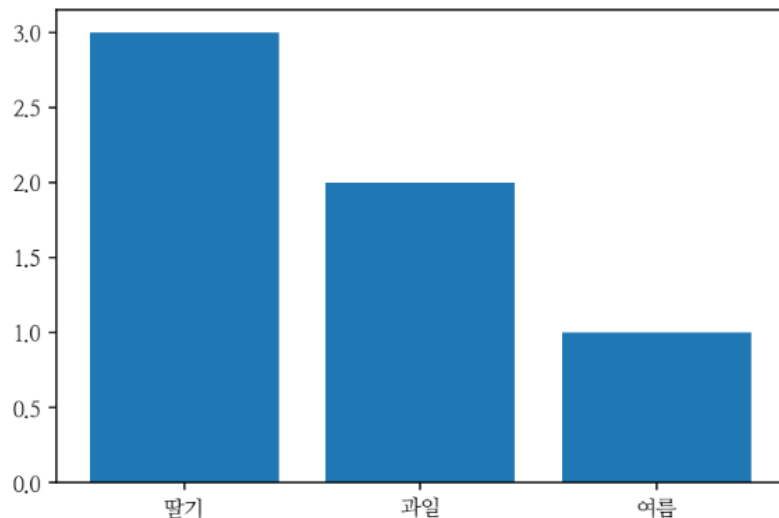
None. 결과 출력은 없다

[1, 2, 3, 5, 7]. 원 데이터가 변경되었다

bar 그래프 연습

```
wordInfo_sample = Counter({'여름':1, '과일':2, '딸기':3})  
sorted_keys_sample = sorted(wordInfo_sample, key=wordInfo_sample.get, reverse=True)  
sorted_values_sample = sorted(wordInfo_sample.values(), reverse=True)
```

```
import matplotlib.pyplot as plt  
plt.bar(range(len(wordInfo_sample)), sorted_values_sample)           # X축의 위치, 각 x의 높이  
plt.xticks(range(len(wordInfo_sample)), sorted_keys_sample)         # X축의 위치, 각 x의 라벨  
plt.show()
```

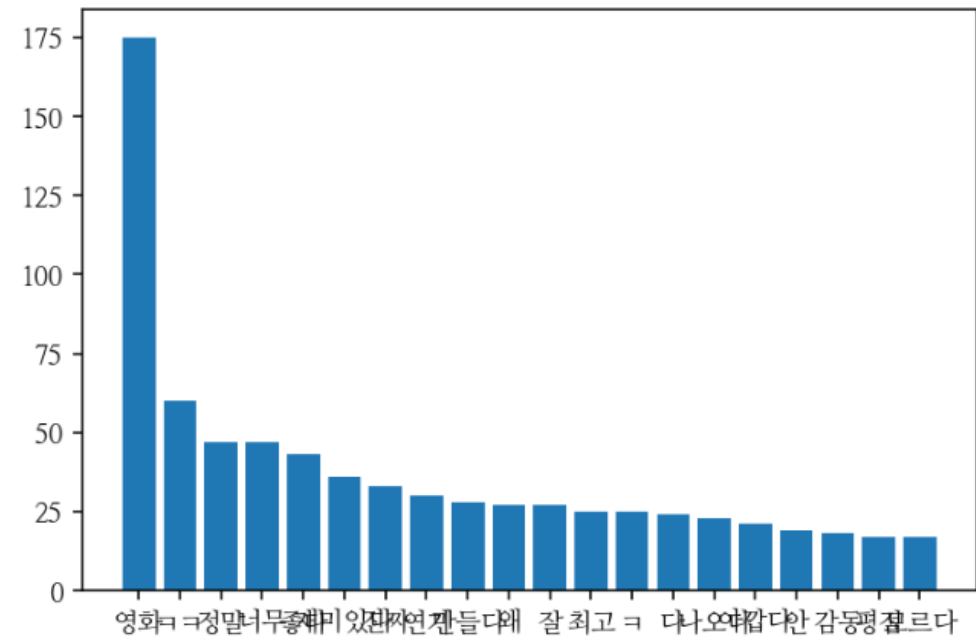


- ※ 설치한 한글이 인식되지 않는 경우가 많다
- ※ 한글 그래프는 가능한 주피터 또는 파이참에서 진행한다
- ※ 주피터에서는 %matplotlib inline 을 먼저 입력한다

그래프 그리기

앞에서 20개까지만 출력

- `sorted_keys = sorted(wordInfo, key=wordInfo.get, reverse=True)`
- `sorted_values = sorted(wordInfo.values(), reverse=True)`
- `import matplotlib.pyplot as plt`
- `plt.bar(range(20), sorted_values[:20])`
- `plt.xticks(range(20), sorted_keys[:20])`
- `plt.show()`



전체 데이터 그래프

```
sorted_keys = sorted(wordInfo, key=wordInfo.get, reverse=True)  
sorted_values = sorted(wordInfo.values(), reverse=True)
```

```
import matplotlib.pyplot as plt  
plt.bar(range(len(wordInfo)), sorted_values)  
plt.xticks(range(len(wordInfo)), sorted_keys)  
plt.show()
```



※ 전체 데이터는 항목이 너무 많아 제대로 나타나지 않는다
(5분 정도 소요)

Word Cloud

- # mergedTextList 데이터를 하나의 긴 문자열로 변환한다
- # 워드클라우드를 문자열의 공백을 기준으로 다시 분리하여 자체 카운팅한다

- linedata = ''.join(mergedTextList)
- print(linedata)

- !pip install wordcloud
- from wordcloud import WordCloud
- cloud = WordCloud(font_path=fontpath, width=800, height=600).generate(linedata)
- plt.imshow(cloud, interpolation='bilinear') # 글자를 더 부드럽게 나오게 한다
- plt.axis('off') # 축의 위치 정보 off
- plt.show()

워드클라우드 패키지 설치

background_color='white'



Word Cloud with mask

워드 클라우드에 마스크를 적용해본다

① 먼저, mask 파일이 있는 경로를 확보한다

- import os
- d = os.getcwd()
- print(d) # /content/gdrive/My Drive/pytest

② alice.png를 RGB 값으로 읽기

- import numpy as np
- from PIL import Image
- from os import path
- alice_mask = np.array(Image.open(path.join(d, "alice.png")))

③ 워드 클라우드에 mask 적용

- cloud = WordCloud(font_path=fontpath, width=800, height=600, background_color='white', **mask=alice_mask**).generate(linedata)
- plt.imshow(cloud, interpolation='bilinear')
- plt.axis('off')
- plt.show()

