

# MOVIE - TI

감정 특화 유사도 기반의 맞춤형 영화 추천 서비스를 통한  
영화 산업 활성화 전략

by

김민영 / 김채현 / 한은결 / 허지원

# 목차

1 주제 선정 배경 및 문제 원인 분석

---

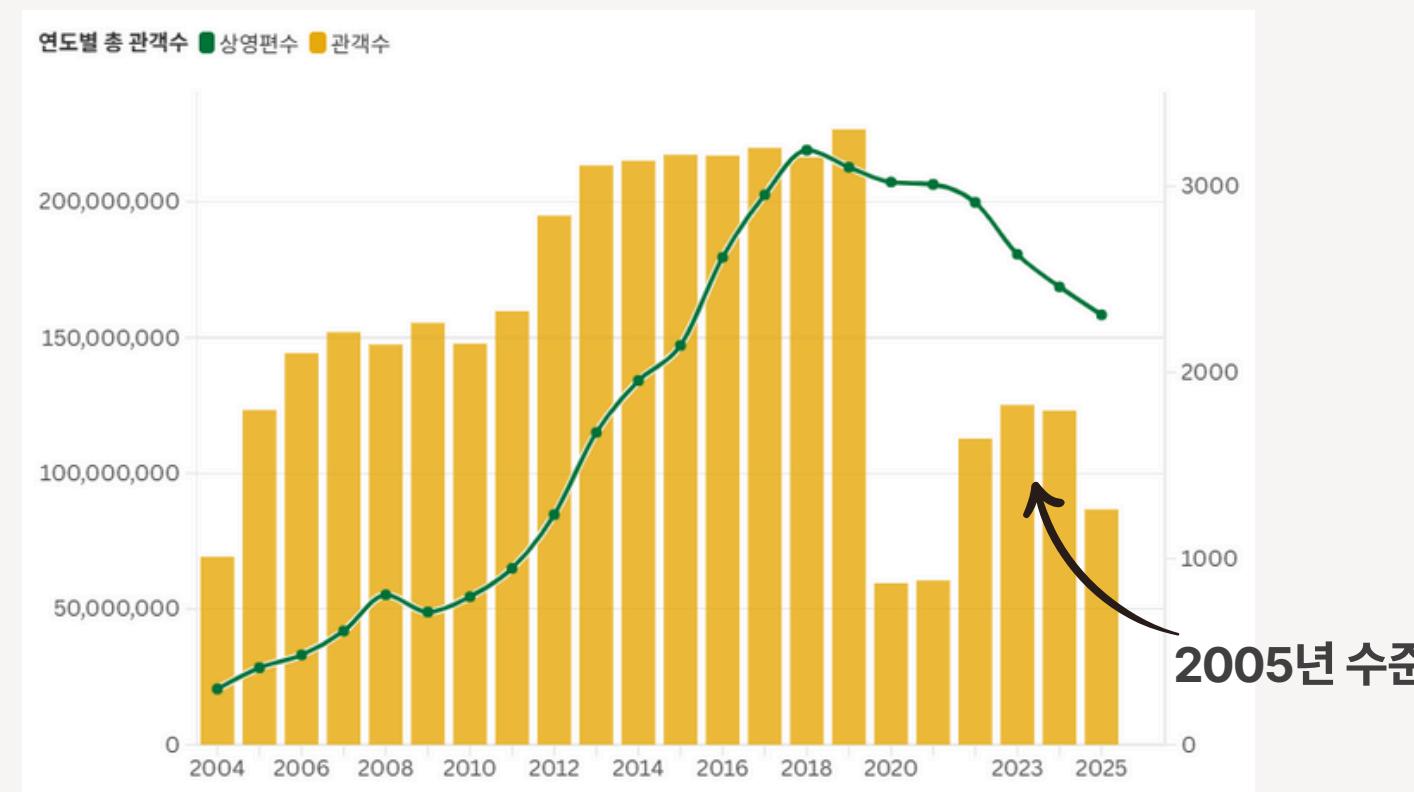
2 문제 해결 방안 : MOVIE-TI

---

3 결론 및 기대효과

# 주제 선정 배경

## 연도별 관객수 및 상영 편수



### 2000년대~

본격적인 산업화의 길에 들어서며 산업규모가 크게 성장  
→ 스크린 숫자와 상영편수가 증가하며 관객수도 같이  
증가

### 코로나 이후 3개년

상영편수(2669편)는 2016년에 근접한 반면,  
연도별 관객수(약 120만 명)는 2005년 수준에 불과

# 극장 관람 빈도 감소 이유

〈그림 2-48〉 극장 관람 빈도 감소 이유

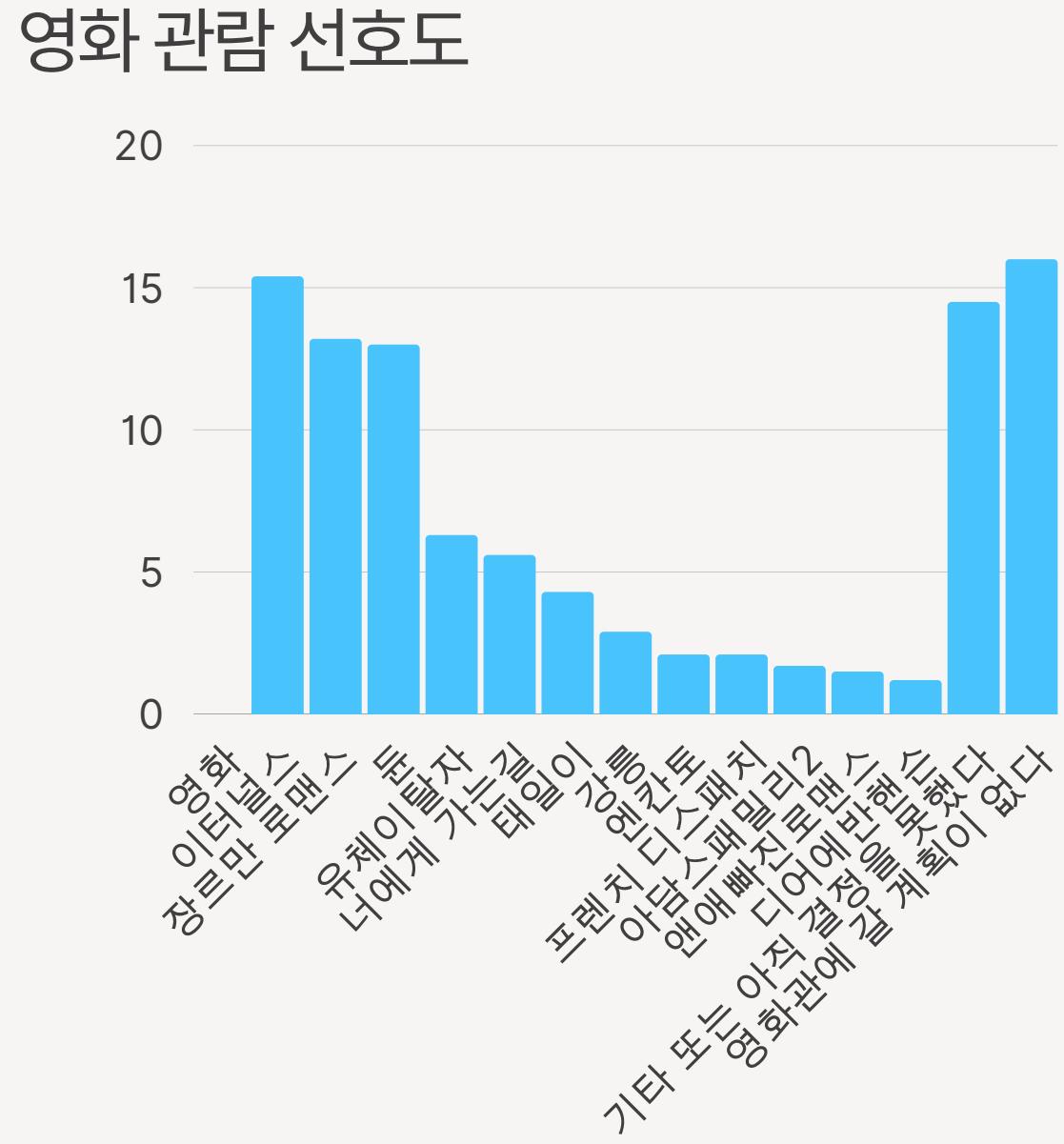
(단위:%)



극장 관람 빈도 감소 이유의 1위는 '볼만한 영화가 없어서', 2위는 '가격이 올라서'  
(2022년도 행태조사에서는 각각 3위, 1위)

2023년 영화소비자 행태조사

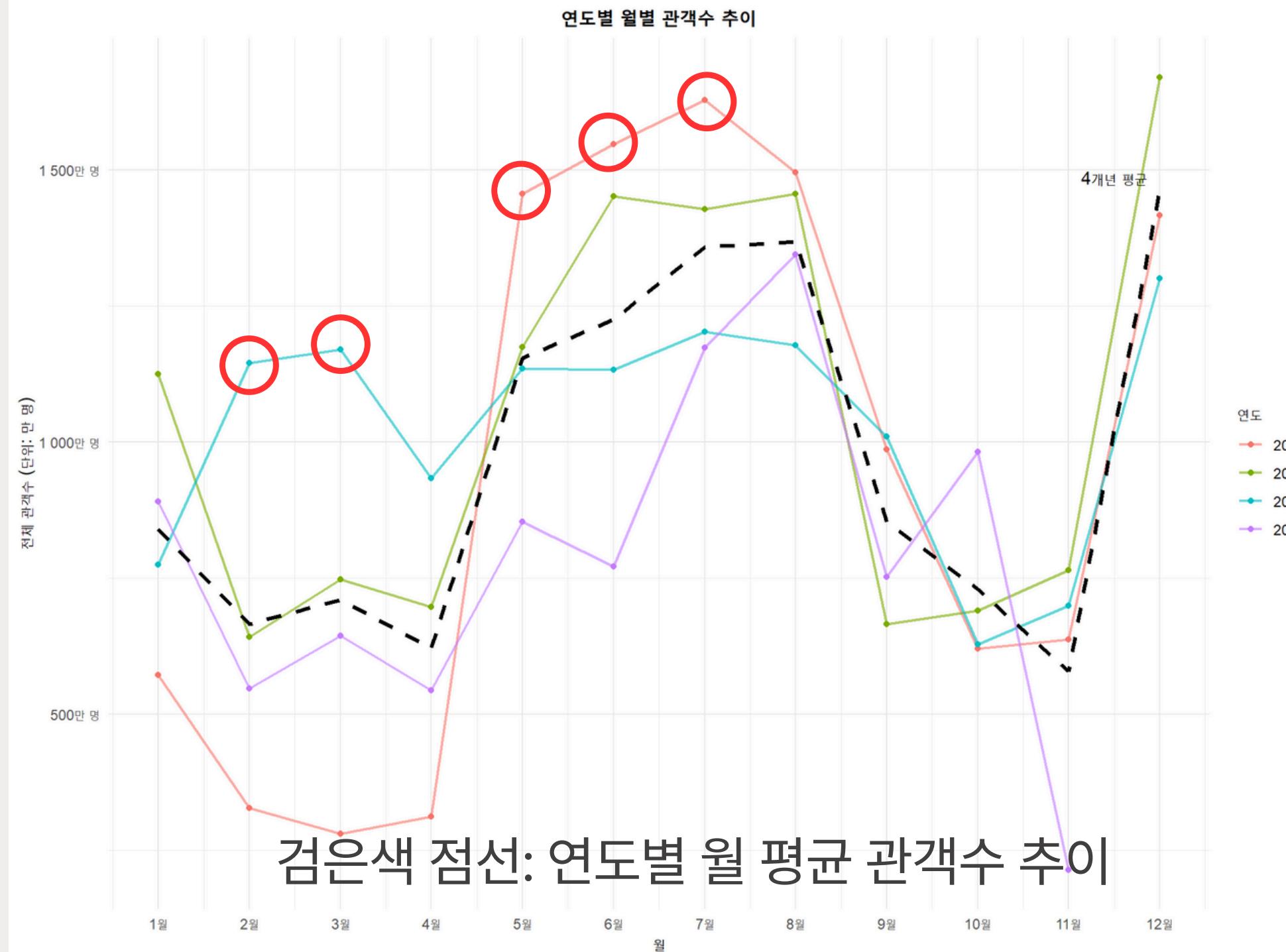
# 문제 원인 분석



- (주)리얼미터에서 진행한 영화 선호도 관련 국민인식 조사 중 2021년 11월 3주차에 해당하는 통계
  - '기타 또는 결정을 못했다'에 해당하는 수치는 14.5%로 결코 작지 않음
  - 영화관이 비활성화된 현재 상황에 대해서 해당 통계를 근거한 원인에 집중해야함

영화 선호도 관련 국민인식 조사: 11월 3주차 영화관람 선호도

# 사람들은 보고 싶은 영화가 있으면 보러 간다



평균보다 높은 수치를 보였던 시점

< 2024년 2월, 3월 >

“파묘”, “듄:파트2”, “웡카”(1월 말)

< 2022년 5월, 6월, 7월 >

“범죄도시2”

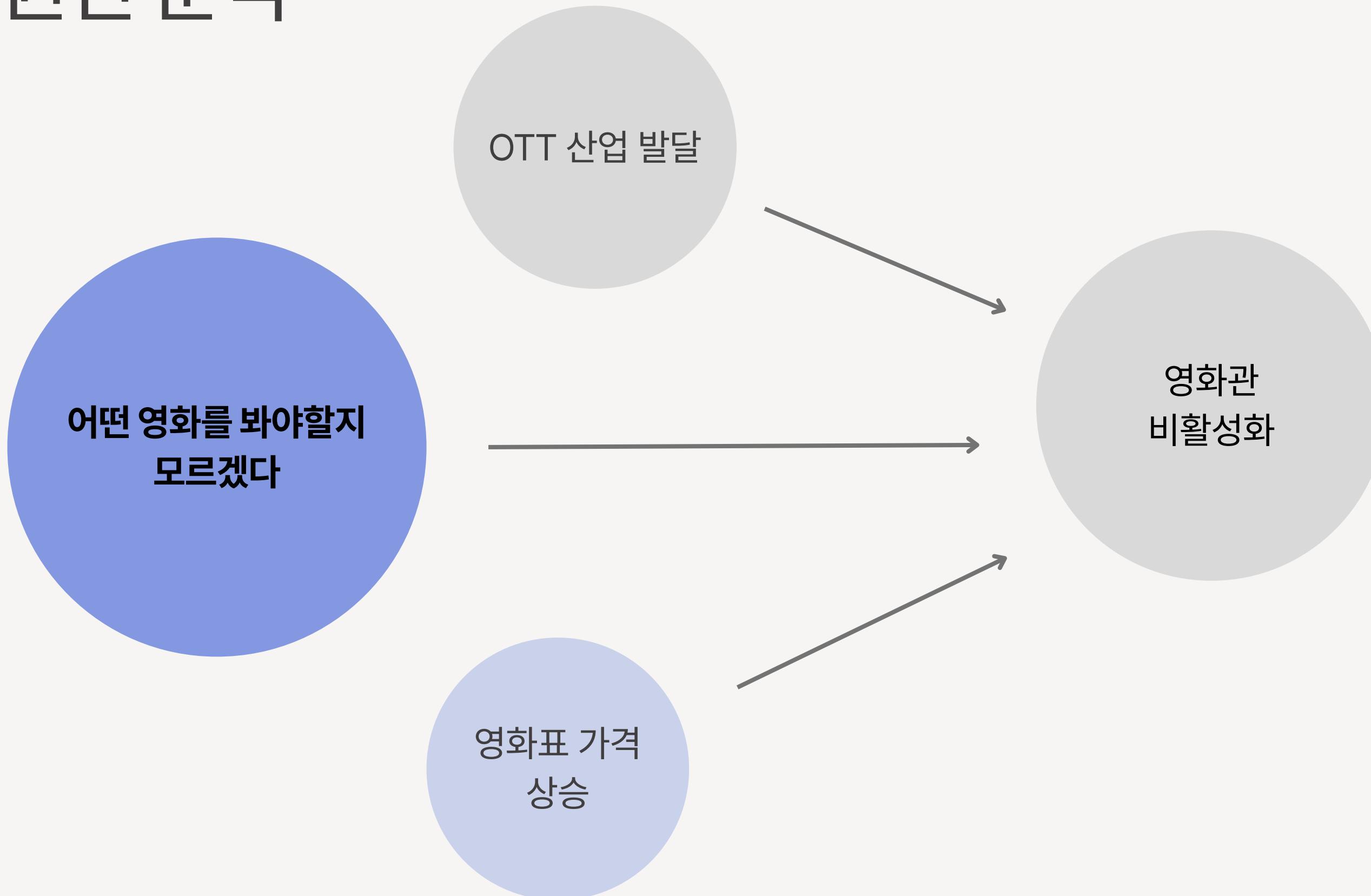
“탑건:매버릭”

“쥬라기월드 : 도미니언”

“토르 : 러브 앤 썬더”

“닥터스트레인지:대혼돈의 멀티버스”

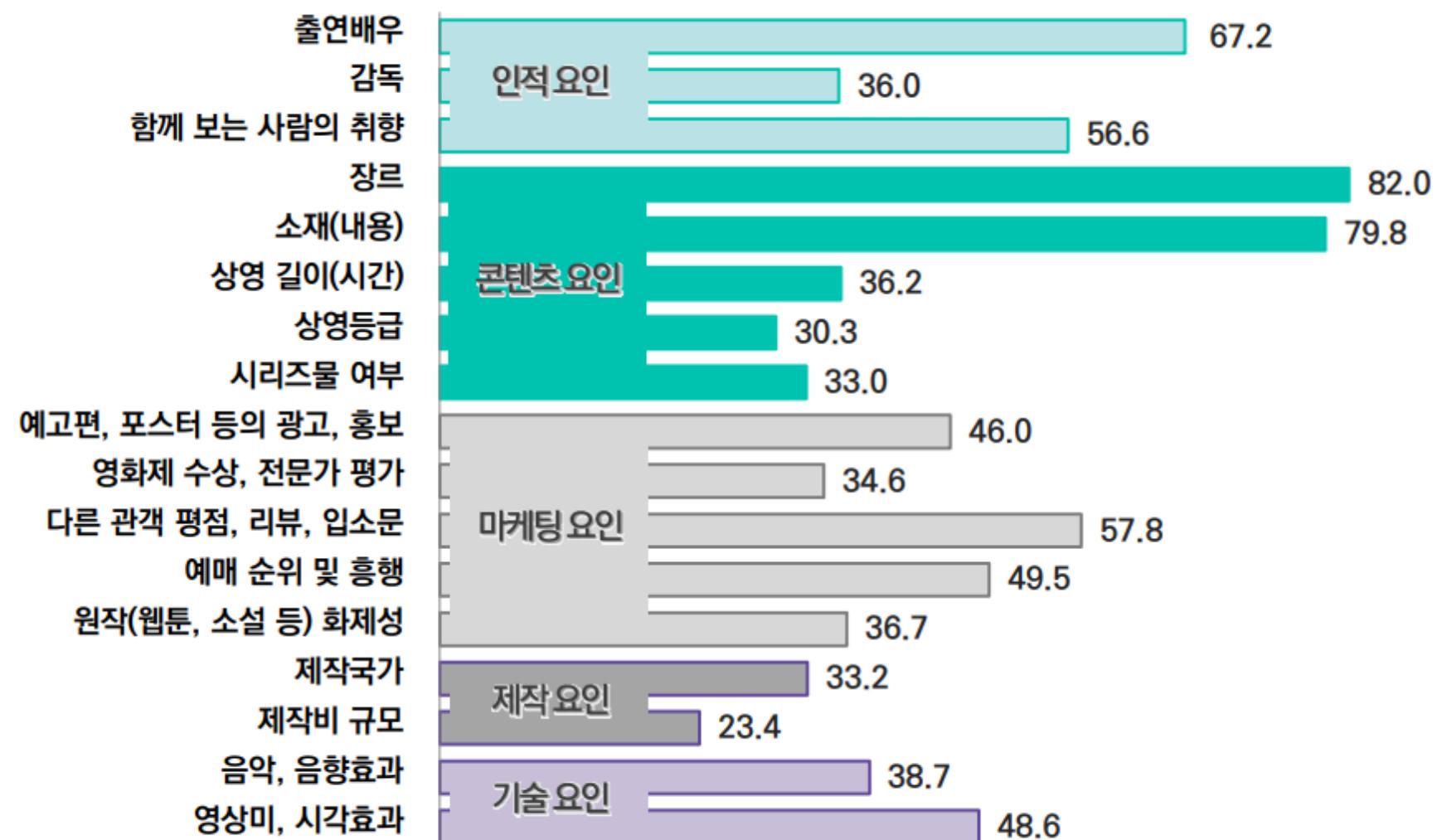
# 문제 원인 분석



# 영화 선택 요인

〈그림 2-24〉 영화 선택 요인

(단위:Top2%)



## 영화 선택 요인 1, 2위: 장르, 소재

2022년도 1위, 2위 : 소재, 장르

2020-2021년도 1위 : 소재 및 장르

영화 관람객들이 이 둘을 가장 우선적으로 고려하여 영화를 선택한다는 점을 근거로, 영화 내용의 특성을 기반으로 영화를 추천하는 아이디어 제안

## 해결방안 - 서론

# OTT 시청과 영화관람의 비율 TOP5

```
years = sorted(
    list(set([col.split('_')[0] for col in df.columns if '_모바일 콘텐츠/OTT 시청' in col])),
    reverse=True
)

# 변수 이름을 'results_all_years'로 통일
results_all_years = {}

for year in years:
    if year == '2019':
        continue
    ott_col = f'{year}_모바일 콘텐츠/OTT 시청'
    movie_col = f'{year}_영화관람'

    if ott_col in df.columns and movie_col in df.columns:
        temp_df = df[['상세구분', ott_col, movie_col]].copy()

        temp_df[ott_col] = pd.to_numeric(temp_df[ott_col], errors='coerce').fillna(0)
        temp_df[movie_col] = pd.to_numeric(temp_df[movie_col], errors='coerce').fillna(0)

        temp_df['차이(%)'] = (temp_df[ott_col] - temp_df[movie_col]).abs()

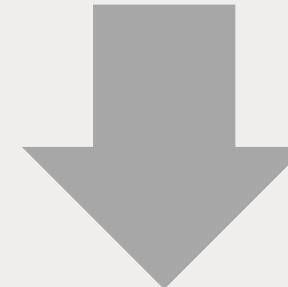
        top_5_data = temp_df[temp_df['차이(%)'] > 0].nlargest(5, '차이(%)')

        top_5_data = top_5_data.rename(columns={
            ott_col: 'OTT 시청(%)',
            movie_col: '영화관람(%)'
        })

        if not top_5_data.empty:
            # 'results_all_years' 변수에 저장
            results_all_years[year] = top_5_data
```

국민여가실태-가장 많이 참여한 여가활동(1순위)

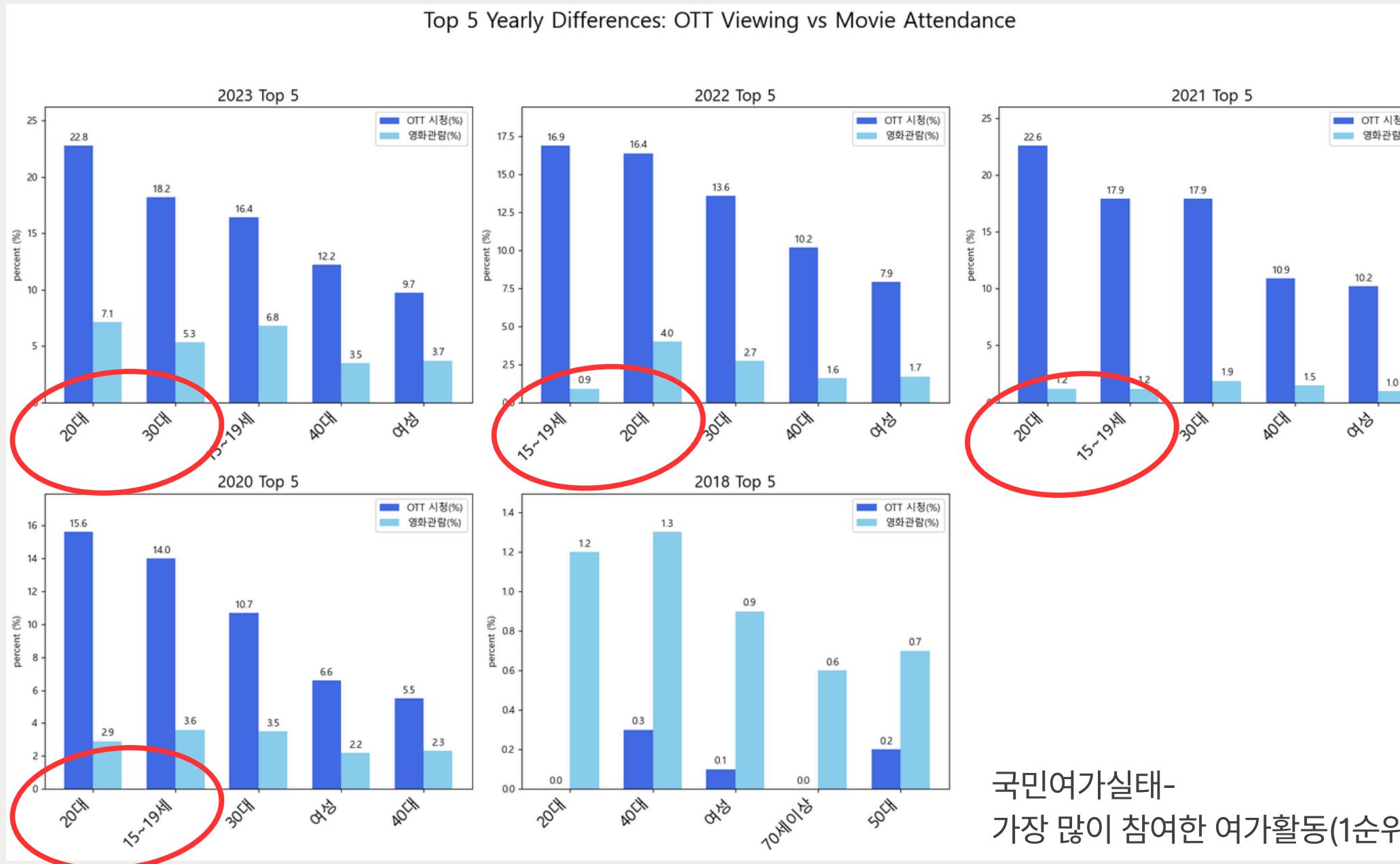
국민여가실태조사 데이터를 활용하여,  
연도별 여가활동 중  
OTT 시청과 영화관람 비율의 차이가  
가장 큰 성별/연령대 TOP 5 선정



시각화  
(matplotlib 이용)

## 해결방안 - 서론

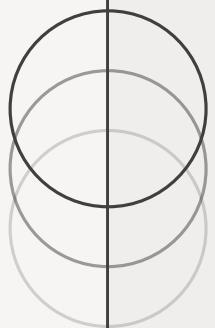
# OTT 시청과 영화관람의 비율 TOP5 - 그래프



OTT 시청과 영화관람 비율이 가장 많이 차이나는 **15~19세**와 **20대**를 공략할 필요가 있음  
영화관람 비율이 낮은 4050대는 OTT 시청도 그리 높지 않은 것으로 보아 영상 시청을 즐기질 않을 확률이 높음  
\*2019년도는 OTT시청이 조사가 되어있지 않아 제외

해결방안 -서론

# 해결 방안 제안



MBTI: 성격 유형 검사 도구

10,20대의 공감대



**MOVIE-TI**

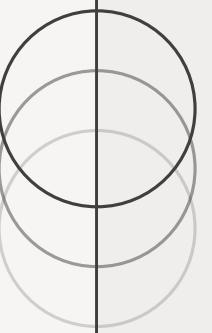
: 영화의 특징을 16가지로 분류

관객들의 영화 취향을 파악하고, 그에 맞게 현재 상영 중인 영화를 추천하여  
극장으로의 방문을 증가

해결방안 - 서론

# MOVIE - TI

## 서비스 시나리오



추천 대상 영화: 현재 상영 중인 영화



사용자는 질문을 통해  
무비티아이 테스트를 진행



무비티아이의 결과를 보여주며, 이를 바탕으로  
현재 상영중인 영화를 추천하고 예고편을 보여줌

이때, 해당 영화의 할인권을 지급하고,  
서비스 시스템에 사용자의 무비티아이를 저장

H L I A  
A P P Y     I M A G I N A R Y  
L L T

해결방안 - 서론

# MOVIE - TI

서비스 시나리오

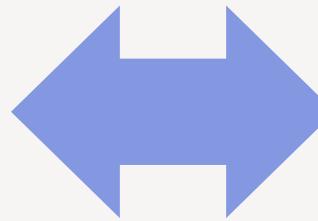


이후에도 무비티아이의 영화가 개봉하면  
사용자에게 알림을 보냄  
→ 영화관의 지속적 방문

해결방안 -본론

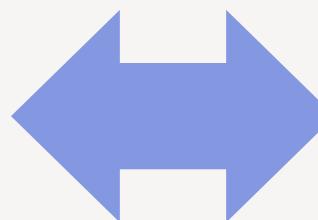
# 무비|티아이 유형 : S/H L/W R/I A/M

눈물, 비극, 슬프다, 아프다, 멍멍하다



웃기다, 즐겁다, 코믹, 기쁨, 힐링

가볍다, 아무 생각 없이, 편안하게, 오락, 퀄팅타임



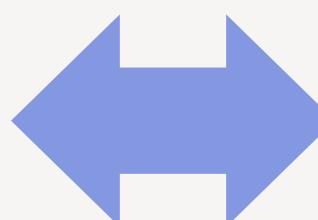
철학, 메시지, 무겁다, 사회 비판, 고찰

현실적, 실화, 공감, 주변의 이야기, 다큐



판타지, 상상력, 마법, SF, 비현실, CG

전체, 12세



15세, 청불

해결방안 - 본론

# 영화 장르 선택

선택 장르

액션 / 드라마 / 범죄/ 코미디 / 판타지 / SF / 사극

해당 장르별 영화 약 15~20편, 총 111편 선별

역대 박스오피스 300위 중  
장르별 관객수 TOP 20인 영화들로 선정

## 해결방안 - 본론

# 영화 리뷰 크롤링 - 왓챠피디아

```
#영화 검색
query_txt = input("영화명: ")

textbox = wait.until(EC.presence_of_element_located((By.NAME, "searchKeyword")))
textbox.clear()
textbox.send_keys(query_txt + "\n")
time.sleep(2)

#첫 번째 검색 결과 클릭
movie_click = wait.until(EC.element_to_be_clickable((By.CSS_SELECTOR,
'a[href*="/contents/]')))
movie_click.click()
time.sleep(2)

#리뷰 페이지로 이동
driver.execute_script("window.scrollTo(0, 1000);")

try:
    more_view = wait.until(EC.element_to_be_clickable((By.XPATH, '//a[contains(@href,
"comments")]')))
    more_view.click()
    print("더보기 클릭 → 리뷰 페이지로 이동")
except:
    print("더보기 없음, 계속 진행")

#스크롤을 200개까지
max_reviews = 200
scroll_pause = 1.2
no_increase_limit = 3

review_elements = []
seen_count = 0
no_increase_count = 0

while len(review_elements) < max_reviews:
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(scroll_pause)

    review_elements = driver.find_elements(By.CSS_SELECTOR, 'p.CommentText') # 새로 로드된 개수 출력
    if len(review_elements) != seen_count:
        seen_count = len(review_elements)
        print(f" 현재 댓글 수: {seen_count}")

    #더 이상 증가가 없을 경우
    if len(review_elements) >= max_reviews:
        print(f" {max_reviews}개 댓글 도달 → 스크롤 중단")
        break

    time.sleep(1)
```

```
#스포일러 보기 버튼 클릭
spoiler_buttons = driver.find_elements(By.CSS_SELECTOR, "button.css-1h47l7y-Button")
for btn in spoiler_buttons:
    try:
        driver.execute_script("arguments[0].click()", btn)
        time.sleep(0.3)
    except:
        pass

#리뷰 수집
review_elements = driver.find_elements(By.CSS_SELECTOR, 'p.CommentText')
data = [{"movie_title": query_txt, "review_text": el.text} for el in review_elements[:200]]

df = pd.DataFrame(data, columns=["movie_title", "review_text"])

import os

#저장
#공백->언더바
movie_name = query_txt.replace(" ", "_")

#다운로드 폴더 생성
folder_path = os.path.join(os.path.expanduser("~"), "Downloads")
os.makedirs(folder_path, exist_ok=True)

#최종 저장 경로
save_path = os.path.join(folder_path, f"watcha_review_{movie_name}.csv")

#저장
df.to_csv(save_path, index=False, encoding="utf-8-sig")

print(f"\n 저장 완료 → {save_path}")
print(f" 수집된 리뷰 수: {len(df)}")
```

리뷰에서 많이 쓰이는 단어를 가져오기 위해  
리뷰 수가 비교적 많은 **왓챠피디아** 영화 리뷰를 선택

**Selenium**을 이용하여 댓글을 **영화마다 200개씩 크롤링** (총 222000개의 댓글)  
→ 댓글 내용 추출 후 엑셀로 저장

해결방안 - 본론

# 영화 리뷰 크롤링 결과



- watcha\_review\_7번방의\_선물.csv
- watcha\_review\_굿바이\_싱글.csv
- watcha\_review\_그것만이\_내\_세상.csv
- watcha\_review\_극한직업.csv
- watcha\_review\_내\_아내의\_모든\_것.csv
- watcha\_review\_럭키.csv
- watcha\_review\_바람과\_함께\_사라진다.csv

movie_title	review_text
엑시트	감정적 찌꺼기를 남기지 않은 채 할 수 있는 것에 집중한 빅시즌 영화의 힘.
엑시트	재난과 코믹을 합친 새로운 장르.. 조정석이기에 가능하지 않았을까
엑시트	가족, 취업, 연애를 짊어지고 벽타는 한국식 왁자지껄 명랑 재난물
엑시트	한국에서 만든 재난 영화중에 가장 괜찮다 신파없고 웃다가 오면된다
엑시트	참신하고 신박한 재난코믹물, 텐트폴 영화의 특징들을 잘 갖추었다.
엑시트	만성적 분노가 일으킬 수 있는 현실적 재난에 대한 상상력. 한국적 히어로는 보통의 사람이 자신의 몫을 내어놓는 순간 탄생한다
엑시트	영리한 현실반영의 템플런.

# 리뷰 데이터 전처리 - konlpy, Okt

## 1. 리뷰에서 자주 사용되는 단어 추출

### konlpy + Okt (한국어 형태소 분석기)

- 일상 텍스트나 SNS처럼 짧고 **비정형적인 문장을 처리하는** 데 특히 강점이 있어 선택
- 문장은 형태소 단위로 분리하고, 동사는 원형으로 변환하여 분석
- **불용어 제거와 정규화를 통해 의미 있는 명사·동사·형용사만 남기도록 정제**

```
import pandas as pd
from konlpy.tag import Okt
from collections import Counter

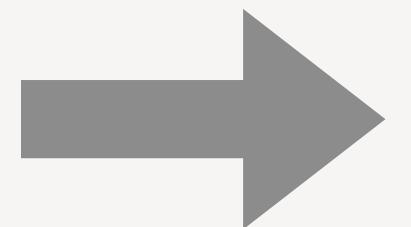
okt = Okt()
all_tokens = []
cleaned_reviews = []

for file_path in files:
    df = pd.read_csv(file_path)

    for text in df['review_text']:
        text = str(text)
        tokens = okt.pos(text, stem=True)
        filtered = [word for word, pos in tokens if pos in ['Noun', 'Verb', 'Adjective']]
        normalized = [
            (w[:-2] if w.endswith('하다') else w)
            for w in filtered
            if (w not in stopwords and len(w) > 1)
        ]
        all_tokens.extend(normalized)
        cleaned_reviews.append(" ".join(normalized))

# 전체 단어 빈도 계산 후 단어 출력
term_freq = Counter(all_tokens)
for word, count in term_freq.most_common(200):
    print(f"{word}: {count}")
```

Counter를 통해  
단어 빈도수를 계산



캡틴	: 132
오락	: 129
존재	: 126
인물	: 126
정의	: 126
등장	: 124
시간	: 123
순간	: 122
마음	: 122
시작	: 120
그냥	: 120
마지막	: 117
탑건	: 115
늘다	: 115
맞다	: 113
새롭다	: 112
인간	: 111
대사	: 111
쿠키	: 109
스타크	: 109
사실	: 108
이상	: 106
범죄	: 104
처음	: 102
사랑	: 101

# 리뷰 데이터 전처리

## 2. 중복 단어 제거 후 랜덤 섞기

```
words = [
    "사랑", "존재", "새롭다", "재밌다", "미래", "공통", "우주", "기억", "시간", "세계", "혁명",
    "액션", "운명", "시대", "매력", "현실", "상상력", "로봇", "지구", "압도", "외계인", "재미있다",
    "지루", "아름답다", "과학", "유전자", "타임", "인류", "거대", "과거", "반전", "상상", "감동",
    "희망", "놀라다", "추억", "철학", "영원", "의지", "오락", "대단", "캐릭터", "원작", "판타지",
    "시작", "이상", "전편", "성장", "주인공", "설정", "마법", "지옥", "전개", "세계관", "반지",
    "게임", "신비", "속편", "유치", "소재", "분위기", "드래곤", "서사", "전작", "마법사", "여정",
    "화려", "소설", "코미디", "웃음", "웃기다", "웃다", "가볍다", "코믹", "추리", "대사", "개그",
    "가족", "유머", "유쾌", "케미", "코드", "깔깔", "즐기다", "터지다", "공감", "사건", "억지",
    "신선", "빠진다", "나쁘다", "좀비", "하이로", "모르다", "빌려", "캡틴", "정의", "순간", "마지막",
    "늘다", "인간", "사실", "범죄", "영웅", "성공", "남다", "완벽", "상황", "훌륭", "사회",
    "기대", "인상", "세상", "싸우다", "지키다", "끌나다", "긴장감", "멋지다", "수트", "블록버스터",
    "확실", "쾌감", "치다", "나가다", "편이", "이르다", "끄다", "통쾌", "드라마", "방식", "완성",
    "이해", "괜찮다", "부족", "몰입", "친구", "한계", "악당", "전투기", "역사", "조선", "사극",
    "전투", "한국", "전쟁", "이순신", "나라", "백성", "전투씬", "안시성", "독립군", "일본군",
    "권력", "산성", "호랑이", "인물", "연기력", "아버지", "캐스팅", "정조", "인조", "장군", "죽음",
    "북적", "역할", "조커", "형사", "잡다", "조폭", "도시", "살인", "느와르", "경찰", "검사",
    "잔인", "피해자", "범죄자", "인생", "놀란", "살아가다", "아프다", "슬프다", "쉬다", "희생",
    "계급", "깊다", "힘들다", "가난", "진실", "눈물", "문제", "감사", "뜨겁다", "재난", "집중",
    '후반', '폭력', '뻔', '신파', '다루다', '녀석', '의미', '재미', '죽다'
]
```

선별된 단어들(204개)

## 중복 단어 제거

```
25 # 중복 제거
26 unique_words = []
27
28 for word in words:
29     if word not in unique_words:
30         unique_words.append(word)
31
32 print(unique_words)
33 print(len(unique_words))
```

```
['사랑', '존재', '새롭다', '재밌다', '미래', '공통', '우주', '기억', '시간', '세계', '혁명', '액션', '운명', '시대', '매력', '현실', '상상력', '로봇', '지구', '압도', '외계인', '재미있다', '지루', '아름답다', '과학', '유전자', '타임', '인류', '거대', '과거', '반전', '상상', '감동', '희망', '놀라다', '추억', '철학', '영원', '의지', '오락', '대단', '캐릭터', '원작', '판타지', '시작', '이상', '전편', '성장', '주인공', '설정', '마법', '지옥', '전개', '세계관', '반지', '게임', '신비', '속편', '유치', '소재', '분위기', '드래곤', '서사', '전작', '마법사', '여정', '화려', '소설', '코미디', '웃음', '웃기다', '웃다', '가볍다', '코믹', '추리', '대사', '개그', '가족', '유머', '유쾌', '케미', '코드', '깔깔', '즐기다', '터지다', '공감', '사건', '억지', '신선', '빠진다', '나쁘다', '좀비', '하이로', '모르다', '빌려', '캡틴', '정의', '순간', '마지막', '늘다', '인간', '사실', '범죄', '영웅', '성공', '남다', '완벽', '상황', '훌륭', '사회', '기대', '인상', '세상', '싸우다', '지키다', '끌나다', '긴장감', '멋지다', '수트', '블록버스터', '확실', '쾌감', '치다', '나가다', '편이', '이르다', '끄다', '통쾌', '드라마', '방식', '완성', '이해', '괜찮다', '부족', '몰입', '친구', '한계', '악당', '전투기', '역사', '조선', '사극', '전투', '한국', '전쟁', '이순신', '나라', '백성', '전투씬', '안시성', '독립군', '일본군', '권력', '산성', '호랑이', '인물', '연기력', '아버지', '캐스팅', '정조', '인조', '장군', '죽음', '북적', '역할', '조커', '형사', '잡다', '조폭', '도시', '살인', '느와르', '경찰', '검사', '잔인', '피해자', '범죄자', '인생', '놀란', '살아가다', '아프다', '슬프다', '쉬다', '희생', '계급', '깊다', '힘들다', '가난', '진실', '눈물', '문제', '감사', '뜨겁다', '재난', '집중', '후반', '폭력', '뻔', '신파', '다루다', '녀석', '의미', '재미', '죽다']
```

```
import random

random.shuffle(unique_words)

print(unique_words)
print(len(unique_words))
```

랜덤 섞기

## 해결방안 - 본론

# 카테고리 시드 완성

- snunlp/KR-SBERT-V40K-klueNLI-augSTS

## 3. 한국어 감정 특화 문장 유사도 모델을 통한 분류

- 한국어 문장의 의미를 벡터로 표현하는 임베딩 모델
- 40,000개의 한국어 문장쌍과 KLUE-NLI, STS 데이터로 학습되어있음  
→ 문장 간 의미 관계와 유사도를 정교하게 판단 가능
- 다양한 NLP 작업에서 높은 정확도를 제공

### + ) 단어를 라벨링하기 전 고려한 조건

#### 1. 유사도 점수가 일정 기준 이상인지 검사

유사도 점수(0~1)에서 제1사분위수(Q1) 수준 → **SIM\_THRESHOLD = 0.25**

#### 2. 두 성향의 유사도 점수 차이가 의미있게 큰지 검사

유의수준 0.05와 유사한 기준 적용 → **DIFF\_THRESHOLD = 0.05**

현실	R   0.692   Δ=0.116
끄다	S   0.270   Δ=0.059
공감	H   0.551   Δ=0.208 R   0.706   Δ=0.324
몰입	H   0.599   Δ=0.221 L   0.613   Δ=0.145
끝나다	S   0.447   Δ=0.077 R   0.344   Δ=0.076
죽음	S   0.547   Δ=0.181 W   0.542   Δ=0.154
우주	W   0.306   Δ=0.090 I   0.365   Δ=0.145
결과	

```
from sentence_transformers import SentenceTransformer, util
import torch
model = SentenceTransformer("snunlp/KR-SBERT-V40K-klueNLI-augSTS")
category_seeds = {
    "S": ["눈물", "비극", "슬프다", "아프다", "먹먹하다"],
    "H": ["웃기다", "즐겁다", "코믹", "기쁨", "힐링"],
    "L": ["가볍다", "아무 생각 없이", "편안하게", "오락", "킬링타임"],
    "W": ["철학", "메시지", "무겁다", "사회 비판", "고찰"],
    "R": ["현실적", "실화", "공감", "주변의 이야기", "다큐"],
    "I": ["판타지", "상상력", "마법", "SF", "비현실", "CG"]
}
category_embeddings = {}
for cat, words in category_seeds.items():
    emb = model.encode(words, convert_to_tensor=True)
    mean_emb = torch.mean(emb, dim=0)
    category_embeddings[cat] = mean_emb

word_list = ['세계관', '순간', '유전자', '인조', '백성', '의지', '반지', '인물', '자루', '뜨겁다', '']
pairs = [("S", "H"), ("L", "W"), ("R", "I")]
# 조건
SIM_THRESHOLD = 0.25
DIFF_THRESHOLD = 0.05

for word in word_list:
    word_emb = model.encode(word, convert_to_tensor=True)
    sims = {cat: util.cos_sim(word_emb, emb)[0].item() for cat, emb in category_embeddings.items()}
    results = []
    for a, b in pairs:
        diff = abs(sims[a] - sims[b])
        max_sim = max(sims[a], sims[b])
        if max_sim > SIM_THRESHOLD and diff > DIFF_THRESHOLD:
            winner = a if sims[a] > sims[b] else b
            results.append(f"{a}-{b}", winner, sims[winner], diff)
    if results:
        print(f"\n{word}")
        for pair_name, winner, score, diff in results:
            print(f" {winner} | {score:.3f} | Δ={diff:.3f}")
```

## 해결방안 - 본론

# 카테고리 시드 완성

## 4. 초기 시드 안정화 후 업데이트

```
word_list = ['편이', '쉬다', '일본군', '연기력', '좀비', '전편', '나라', '피해자', '희망', '나쁘다',  
'추리', '블록버스터', '빌런', '지키다', '사랑', '미래', '재미있다', '압도', '성장', '기대',  
'확실', '철학', '늘다', '전쟁', '오락', '재난', '살아가다', '인류', '살인', '재밌다',  
'게임', '즐기다', '나가다', '장군', '인생', '운명', '다루다', '가난', '느와르', '신비',  
'계급', '캐릭터', '놀란', '기억', '정의', '녀석', '육직', '개그', '사극', '범죄자',  
'코드', '웃음', '원작', '상상력', '유머', '세상', '영원', '전개', '문제', '성공', '사건',  
'쾌감', '설정', '전투씬', '희생', '상황', '범죄', '추억', '드래곤', '깊다', '역사', '감사',  
'시작', '신선', '이해', '존재', '조폭', '의미', '아름답다', '히어로', '터지다', '인상',  
'이상', '소재', '액션', '죽다', '거대', '가족', '검사', '전투기', '역할', '놀라다', '슬프다',  
'소설', '한계', '사실', '공룡', '가볍다', '시간', '괜찮다', '부족', '폭력', '새롭다', '멋지다',  
'지구', '여정', '지옥', '대사', '독립군', '속편', '외계인', '잡다', '마지막', '혁명',  
'세계', '사회', '깔깔', '매력', '악당', '훌륭', '유쾌', '산성', '치다', '과거', '드라마',  
'잔인', '긴장감', '영웅', '유치', '수트', '눈물', '분위기', '시대', '재미', '코믹',  
'억지', '인간', '정조', '신파', '한국', '캐스팅', '안시성', '집중', '과학', '아프다',  
'코미디', '권력', '마법', '이순신', '로봇', '완성', '경찰', '도시', '남다']  
  
pairs = [("S", "H"), ("L", "W"), ("R", "I")]  
SIM_THRESHOLD = 0.25  
DIFF_THRESHOLD = 0.05  
  
temp_dict = {k: [] for k in category_seeds.keys()}  
for word in word_list:  
    word_emb = model.encode(word, convert_to_tensor=True)  
    sims = {cat: util.cos_sim(word_emb, emb)[0].item() for cat, emb in category_embeddings.items()}  
    for a, b in pairs:  
        diff = abs(sims[a] - sims[b])  
        max_sim = max(sims[a], sims[b])  
        if max_sim > SIM_THRESHOLD and diff > DIFF_THRESHOLD:  
            winner = a if sims[a] > sims[b] else b  
            temp_dict[winner].append(word)  
  
print("\n 카테고리별 추가 단어:")  
for cat, words in temp_dict.items():  
    if words:  
        category_seeds[cat].extend(words)  
        print(f" {cat}: {words}")  
print("\n 업데이트된 category_seeds:")  
for k, v in category_seeds.items():  
    print(f"{k}: {v}")
```

### 1. 초기 시드 안정화

키워드의 수가 제한적인 상황에서 자동 분류 시 발생할 수 있는 **의미적 왜곡을 최소화하며**  
**분류 정확도를 높이기 위해 204개의 단어 중 50개 유사도 결과 직접 확인**

→ 직접 분류

### 2. 시드 업데이트

안정화된 시드를 바탕으로 나머지 154개의 단어를 자동으로 분류

→ 라벨 별 키워드 세트 완성

카테고리별 추가 단어:

S: ['나쁘다', '살아가다', '운명', '가난', '사극', '희생', '죽다', '슬프다', '한계', '지옥', '잔인', '눈물', '아프다']  
H: ['편이', '연기력', '희망', '추리', '블록버스터', '빌런', '미래', '재미있다', '압도', '성장', '기대', '확실', '늘다', '오락', '재밌다']  
L: ['연기력', '좀비', '블록버스터', '빌런', '재미있다', '기대', '확실', '재밌다', '즐기다', '신비', '놀란', '육직', '웃음', '상상력']  
W: ['편이', '일본군', '전편', '나라', '추리', '지키다', '사랑', '미래', '철학', '전쟁', '살아가다', '인류', '살인', '나가다', '장군', '인생']  
R: ['편이', '전편', '나라', '피해자', '나쁘다', '지키다', '사랑', '재미있다', '성장', '기대', '확실', '전쟁', '살아가다', '증기다', '장군']  
I: ['좀비', '추리', '블록버스터', '재난', '인류', '느와르', '신비', '상상력', '드래곤', '존재', '액션', '공룡', '지구', '지옥', '인생']

업데이트된 category\_seeds:

S: ['눈물', '비극', '슬프다', '아프다', '먹먹하다', '힘들다', '죽음', '싸우다', '아버지', '나쁘다', '살아가다', '운명', '가난', '사극', '희망', '웃기다', '즐겁다', '코믹', '기쁨', '힐링', '반지', '캡틴', '상상', '웃다', '친구', '통쾌', '대단', '화려', '편이', '연기력', '희망', '가볍다', '아무 생각 없이', '편안하게', '오락', '킬링타임', '상상', '마법사', '웃기다', '통쾌', '화려', '연기력', '좀비', '블록버스터', '철학', '메시지', '무겁다', '사회 비판', '고찰', '세계관', '유전자', '백성', '의지', '서사', '형사', '죽음', '우주', '아버지', '편이', '현실적', '실화', '공감', '주변의 이야기', '다큐', '현실', '감동', '진실', '아버지', '친구', '편이', '전편', '나라', '피해자', '나쁘다', '판타지', '상상력', '마법', 'SF', '비현실', 'CG', '세계관', '우주', '상상', '마법사', '화려', '좀비', '추리', '블록버스터', '재난', '인생']

## 해결방안 - 본론

# 현재 상영 중인 영화 무비티아이 추출

```
import pandas as pd
from sentence_transformers import SentenceTransformer, util
import torch
from collections import Counter

model = SentenceTransformer("snunlp/KR-SBERT-V40K-kluenLI-augSTS")
df = pd.read_csv("watcha_review_베이비걸.csv") # 파일명
df["review_text"] = df["review_text"].astype(str)

df["review_text"] = (
    df["review_text"]
    .str.replace("스포일러가 있어요!!보기", "", regex=False)
    .str.replace("Wn", " ", regex=True)
    .str.strip()
)

# 줄거리
plot_summary = """누가 누구를 지배하는가" 맨해튼의 호화로운 아파트, 자상한 예술

all_texts = " ".join(df["review_text"].tolist())
combined_text = plot_summary + " " + all_texts
print(combined_text)

category_seeds = {
    'S': ['눈물', '비극', '슬프다', '아프다', '먹먹하다', '힘들다', '죽음', '싸우다',
          'H': ['웃기다', '즐겁다', '코믹', '기쁨', '힐링', '반지', '캡틴', '상상', '웃다',
          'L': ['가볍다', '아무 생각 없이', '편안하게', '오락', '킬링타임', '상상', '마법',
          'W': ['철학', '메시지', '무겁다', '사회 비판', '고찰', '세계관', '유전자', '백
          'R': ['현실적', '실화', '공감', '주변의 이야기', '다큐', '방식', '현실', '공감
          'I': ['판타지', '상상력', '마법', 'SF', '비현실', 'CG', '세계관', '우주', '상상
    }
    영상물 등급을 입력받아
    A/M을 결정하는 부분 추가
age_input = "청불" # 연령
def get_age_label(age_text):
    if any(x in age_text for x in ["전체", "12세"]):
        return "A"
    elif any(x in age_text for x in ["15세", "청불"]):
        return "M"
    else:
        return "M"
```

```
age_label = get_age_label(age_input)

category_embeddings = {}
for cat, words in category_seeds.items():
    emb = model.encode(words, convert_to_tensor=True)
    mean_emb = torch.mean(emb, dim=0)
    category_embeddings[cat] = mean_emb

pairs = [("S", "H"), ("L", "W"), ("R", "I")]

text_emb = model.encode(combined_text, convert_to_tensor=True)
sims = {cat: util.cos_sim(text_emb, emb)[0].item() for cat, emb in category_embeddings.items()}

results = {}
for a, b in pairs:
    diff = abs(sims[a] - sims[b])
    winner = a if sims[a] > sims[b] else b
    results[f"{a}-{b}"] = {"winner": winner, "score_a": sims[a], "score_b": sims[b], "diff": diff}

print("감정 카테고리 판정 결과:")
for pair, info in results.items():
    print(f" [{pair}] → {info['winner']} (Δ={info['diff']:.3f})")
emotion_code = ".join([info["winner"] for info in results.values()])
final_code = emotion_code + age_label
print(final_code)
```

## 완성된 키워드 세트를 이용해 진행한 테스트

입력: 영화 리뷰 데이터, 영화의 줄거리, 영상물 등급  
출력: MOVIE - TI

## 감정 카테고리 판정 결과:

[S-H] → S ( $\Delta=0.038$ )

[L-W] → L ( $\Delta=0.113$ )

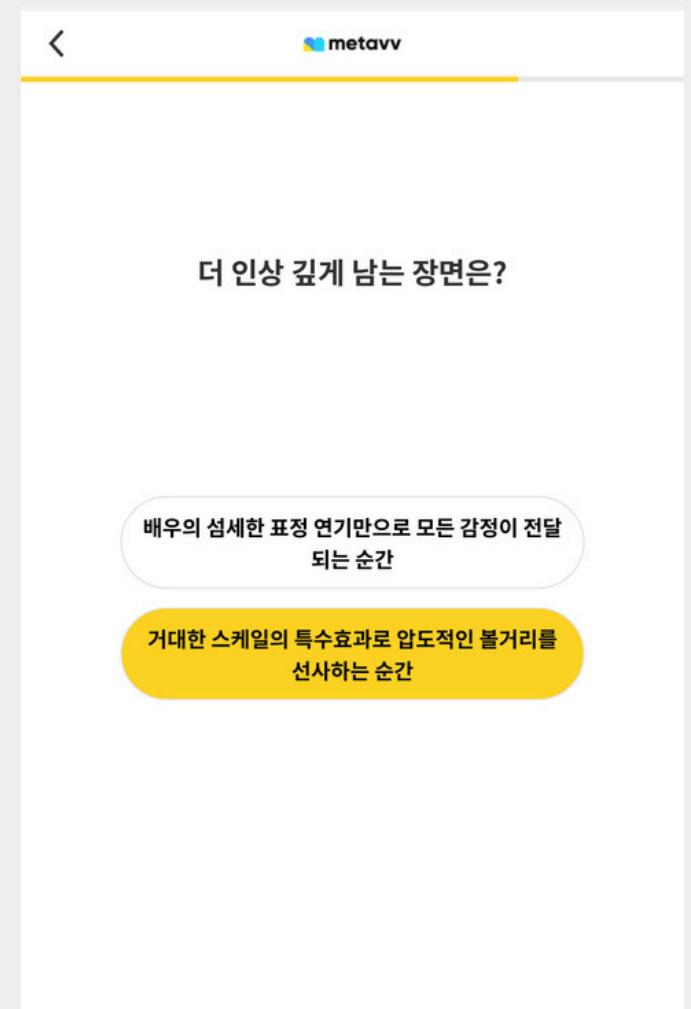
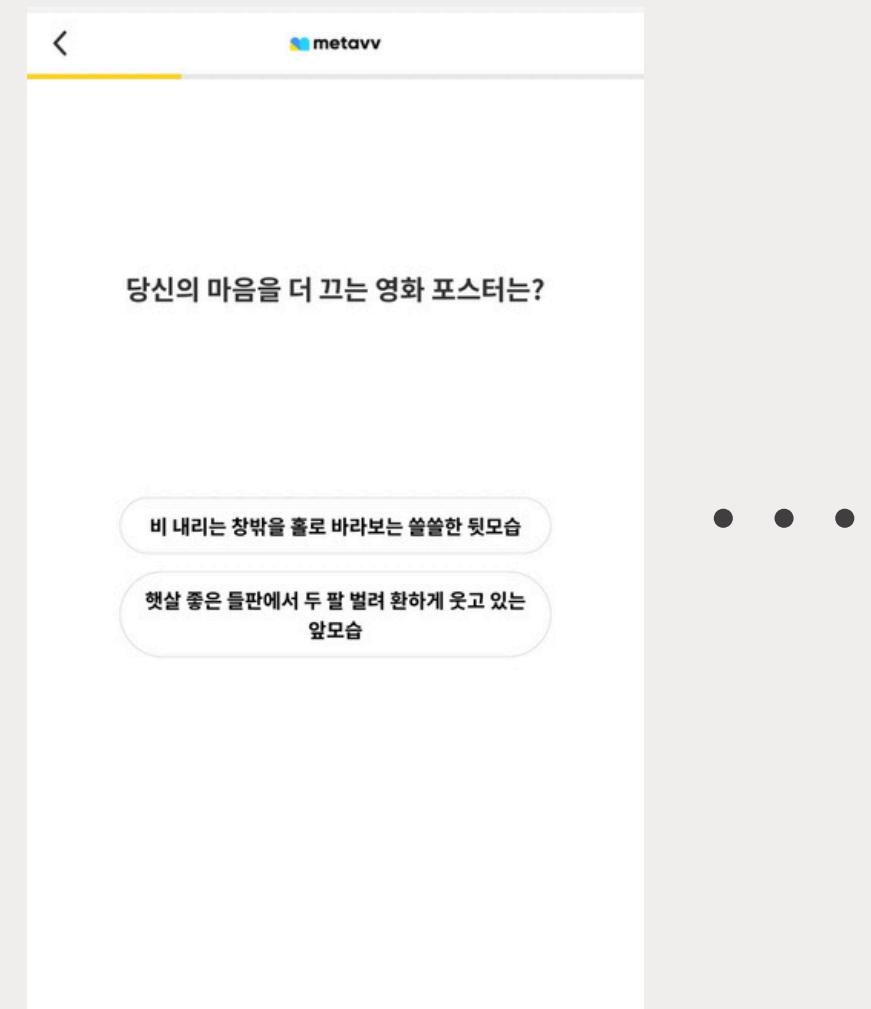
[R-I] → R ( $\Delta=0.084$ )

SLRM

2025년 10월 개봉한 “베이비걸”의 결과  
비교적 적은 리뷰만으로 영화의 특성이 분류됨

해결방안 - 결론

# Movie-TI 테스트 시연 예시



Movie-TI 테스트  
Movie-TI 테스트  
나의 무비티아이는?

A movie poster for 'Predator: 죽음의 땅'. It features a woman in a futuristic suit holding a gun, standing in a jungle environment. The title 'Predator' and '죽음의 땅' are at the bottom, along with the release date '11월 5일 극장개봉'.

Press and hold to save the image

HLIM: "짜릿한 몽상가" ↗

현실의 복잡함은 잠시 잊고, 기쁘고 가벼우면서 상상력 넘치고 살짝 짜릿한 매력이 있는 영화를 보며 힐링하는 타입이에요. 따뜻하면서도 어딘가 자극적인 이야기를 좋아해요 😊

🎥 이거 보러 가는 거 어때~?  
: <https://youtu.be/l3awPAzyOtY?si=BRdLWf2m98TNAutq>

# 결론 및 기대효과

## 개인 선호도별 영화 추천

앞선 통계에서 봤듯이 적지 않은 사람들이 무슨 영화를 볼지 고민한다. 우리가 제안한 무비티아이는 사람들의 취향별 영화 추천을 수행하여, 이러한 고민을 줄여준다.

## 영화값 할인

영화 소비자 행태조사에 따르면 사람들이 영화 관람을 하지 않는 이유 중 하나로 가격을 꼽는다. 쿠폰을 통해 가격을 약간 내리는 것은 영화 관람의 문턱을 낮출 기회이다.

## 지속적인 영화 관람객 확보

Movie-TI도 MBTI처럼 하나의 유행이 된다면 영화관은 지속적으로 관람객을 확보할 수 있을 것이다.

**THANK YOU**

뿌

김민영 / 김채현 / 한은결 / 허지원