영화 추천 시스템 뽀개기

한림대학교 소프트웨어융합대학 빅데이터 전공

김도경
h20151505@g-mail.hallym.ac.kr

지도교수 : 김유섭

세부 내용 PPT

더 자세한 설명을
보고 싶다면 스캔 !

요약

본 연구의 목적은 적은 양의 데이터와 간단한 알고리즘으로 사용자가 좋아하는 영화 제목을 입력했을 때 그와 비슷한 영화를 추천, 추천된 영화 제목과 장르, 영화에 대한 키워드를 같이 보여주는 영화 추천 시스템을 구현하는 것이다. 각 영화를 문서화하여 코퍼스 생성, okt.nouns를 사용하여 명사 추출 후 TF-IDF Matrix화 시켜 term-document matrix를 만든 뒤 코사인 유사도를 따져 영화의 유사도를 파악하는 방식이다. 6가지의 시스템 성능을 비교하여 피처에 적당한 노이즈가 있어야 전혀 상관없는 장르의 영화 추천 현상이 적다는 것을 알게 되었다. 줄거리 + 한줄평의 명사를 피처로 사용한 시스템 6이 전혀 상관없는 장르의 영화 추천 현상이 적을뿐만 아니라 로맨스 외의 대부분 영화들의 유사도가 높게 나와 연구 목적을 달성하였다.

연구 배경 및 연구 질문

추천 시스템은 정보 필터링 시스템의 일종으로, 특정 사용자가 관심을 가질만한 정보를 추천하는 것이다. 1990년대 중후반 추천 시스템의 중 하나인 협업 필터링에 관련 연구 논문이 등장한 이후 추천 시스템은 주요한 연구 분야가 되었다. 추천 시스템에 대한 관심과 활용도가 높아지면서 상품 추천, 영화 추천, 뉴스 추천 등 많은 곳에서 필요한 부분이 되었다. 그중 넷플릭스가 해주는 영화 추천 시스템은 내용 기반 필터링과 협업 필터링이 결합된 하이브리드 추천 시스템이다.

방대한 양의 데이터와 수준 높은 알고리즘은 없지만 이러한 영화 추천 시스템을 비슷하게나마 구현할 수 있을까가 이 연구의 시작점이자 연구 질문이다.

연구 방법

데이터 설명

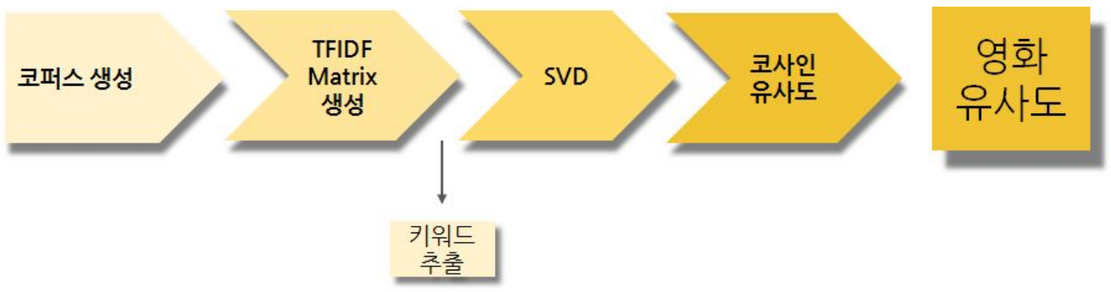
1. 영화 데이터 셋 - 네이버 영화 랭킹 크롤링 총 17가지 장르 3957개 영화

- 드라마, 가족, 코미디, 멜로/애정/로맨스
- 판타지, 모험, SF
- 공포, 미스터리, 스릴러
- 전쟁, 범죄, 액션, 느와르
- 뮤지컬, 애니메이션, 다큐멘터리

제목, 영어제목, 등수, 평점, 장르, 제작국가, 러닝타임, 개봉일, 제작년도, 감독, 출연, 등급, 줄거리, 한줄평

2. 영화 별 문서를 생성, 문서를 줄거리, 한줄평, 줄거리 + 한줄평 세가지 종류로 하여 성능 비교
   문서의 유사도를 따지는 것이 즉 영화 유사도를 따지는 것

분석방법



연구 과정

코퍼스 생성 형태소 분석기 : okt.nouns

문서 토크나이징을 하기 위해서 적절한 형태소 분석기 선정
: KoNLPy 형태소 분석기 _ Okt(Twitter)

토크나이저 종류	사용한 데이터	추출 형태	TFIDF_matrix(자원수)
okt.phrases	줄거리	어절	(3957, 71236)
okt.morphs	줄거리	형태소	(3957, 32491)
okt.nouns	줄거리	명사	(3957, 25309)

" 3957은 문서 수, 뒤의 25309은 feature 개수
3957개의 문서가 25309 토큰으로 표현이 된 것 "

okt.nouns가 자원 수도 더 적고 행렬 vocab top15에서 의미 있는 단어가 더 많이 보이기에 형태소 분석기로 선택하여 토크나이징을 진행

=> 문서가 이젠 추출된 명사로 이루어져 있음

〈 TF-IDF 행렬 vocab top 15 〉

```
vocab = {tfidf.vocabulary_[token]:token for token in tfidf.vocabulary_.keys()}
doc_tfidf, result = tfidf_matrix[14], []
for id, data in zip(doc_tfidf.indices, doc_tfidf.data):
    result.append((vocab[id],data))
print(len(result))
sorted(result, key=lambda x : x[1], reverse = True)
```

vocabulary_ :
각 문서와 연결된
피쳐 인덱스 확인가능

okt.phrases

```
64
[('사춘지간', 0.3513434472608047),
 ('몽주', 0.3138673422247194),
 ('몽규', 0.2635078544560011),
 ('졸업내기', 0.2592660694411221),
 ('오른', 0.2354659629279376),
 ('친구이자', 0.2298540982583172),
 ('폭탄', 0.1961080558664644),
 ('일본', 0.1881940904361408),
 ('청년', 0.1845729930811128),
 ('유학', 0.1825549034197198),
 ('송몽규', 0.17567172363040073),
 ('윤동주', 0.17567172363040073),
 ('몽규시인', 0.17567172363040073),
 ('영원한', 0.1660212677736681),
 ('라이벌이었던', 0.1569336711123597),
```

okt.morphs

```
99
[('몽주', 0.4231537748848565),
 ('몽규', 0.3157858665929035),
 ('일본', 0.16880981528972763),
 ('청년', 0.16497044568455807),
 ('시대', 0.16261696703241568),
 ('송몽규', 0.15789293329645174),
 ('윤동주', 0.15789293329645174),
 ('창씨개명', 0.15789293329645174),
 ('몽규시인', 0.15789293329645174),
 ('촌지', 0.15044037385430056),
 ('언니가다', 0.13486680628744194),
 ('일제강점기', 0.13024757815704543),
 ('매진', 0.12885463855570797),
 ('시인', 0.12751739647012772),
 ('언어', 0.11557078766022434),
```

okt.nouns

```
54
[('몽규', 0.5047310363095024),
 ('몽주', 0.4508938195372066),
 ('일본', 0.17951849902689818),
 ('청년', 0.17578516080993534),
 ('시대', 0.17237689004921644),
 ('송몽규', 0.16824367876983415),
 ('윤동주', 0.16824367876983415),
 ('창씨개명', 0.16824367876983415),
 ('촌지', 0.16030256370013814),
 ('일제강점기', 0.13878601937719965),
 ('매진', 0.13781366432706927),
 ('시인', 0.12751739647012772),
 ('언어', 0.12314708497926459),
 ('졸업', 0.12128041587078317),
 ('폭탄', 0.12040981912887291),
```

TF-IDF Matrix 생성

TF-IDF 방식으로 단어의 가중치를 조정한 BOW 벡터를 만든다.
=> 문서를 tf-idf의 feature matrix로 변환

```
num
```

359559

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(min_df =1, ngram_range=(1,1),lowercase= True, tokenizer = lambda x: x.split())
tfidf_matrix = tfidf.fit_transform(nouns_corpus)
print(tfidf_matrix.shape)
```

(3957, 25309)

=> term-document matrix 생성
희소 행렬(대부분의 값이 0)

min_df : 단어장에 포함되기 위한 최소 빈도
ngram_range : 유니그램, 바이그램 등등
lowercase : 토큰화하기 전 소문자로 바꾸기
tokenizer : 토큰화 방법 명시, 띄어쓰기 별로 토큰화

term-document matrix

피쳐(단어 토큰)

```
pd.DataFrame(tfidf_matrix[:100].toarray(), columns=vocab2).head()
```

영화	언제	어디	서든	생활	전재	뮤지션	돈	설리	원직	반직	…	양원	곡리	전랑	렁렁	오경	운전대	무전도사	무치	타이타닉	답습
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 25309 columns

SVD 차원축소, 코사인 유사도

- 차원 축소 : 희소 행렬 => 밀집 행렬(dense, 실수값)
TF-IDF matrix에 SVD 진행 500차원으로 축소
- 코사인 유사도 측정 :
모든 영화에 대해서 입력받은 영화와의 코사인 유사도를 구해 유사도에 따라 정렬 후 top 3 반환

SVD 차원축소 후 코사인 유사도

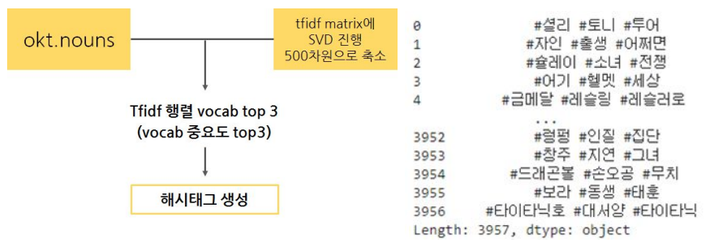
```
...
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=500)
vecs = svd.fit_transform(tfidf_matrix)
```

```
print(vecs.shape)
```

(3957, 500)

```
from sklearn.metrics.pairwise import linear_kernel
cosine_sim = linear_kernel(vecs, vecs)
```

키워드 추출 해시태그 표시




```
0          #셜리 #토니 #투어
1          #자인 #출생 #어쩌면
2          #율레이 #소녀 #전쟁
3          #어기 #헬멧 #세상
4        #금메달 #레슬링 #레슬러로
...
3952       #형평 #인질 #집단
3953       #창주 #지연 #그녀
3954     #드래곤볼 #손오공 #무치
3955       #보라 #동생 #태훈
3956    #타이타닉호 #대서양 #타이타닉
Length: 3957, dtype: object
```

최종 선택 : 시스템 6

전혀 다른 장르의 영화가 추천되는 문제
=> 어느 정도 노이즈가 필요하다고 봄

1) 줄거리, 한줄평을 사용해 나온 각각의 순위를 합쳐 다시 순위를 매기는 것 => 시스템 5
2) 줄거리와 한줄평을 한 문서로 만든 시스템 6

〈최종 선택 이유〉
- 로맨스 외의 대부분 영화들의 유사도가 높음
- 다른 장르의 영화 추천 현상이 현저히 줆

연구 과정

시스템 성능 평가에 사용될 지표

17개의 장르를 5개 부문으로 나눈 후 부문 별 3개의 영화

- 드라마, 가족, 코미디, 멜로/애정/로맨스 — 러브 액츄얼리, 시네마 천국, 동주
- 판타지, 모험, SF — 스파이더맨, 헝거게임, 지구가 멈추는 날
- 공포, 미스터리, 스릴러 — 이끼, 쏘우, 큐브
- 전쟁, 범죄, 액션, 느와르 — 조커, 나우 유 씨 미 : 마술사기단, 어벤져스
- 뮤지컬, 애니메이션, 다큐멘터리 — 말레피센트, 겨울왕국, 다이빙벨

개요

코퍼스 구성을 다르게 하고 다른 매개변수는 동일하게 설정한 후 6개의 시스템을 비교

	시스템1	시스템2	시스템3	시스템4	시스템5		시스템6
토크나이저	okt.nouns	okt.nouns	okt.nouns	okt.nouns	okt.nouns		okt.nouns
raw데이터	줄거리	줄거리	한줄평	한줄평	줄거리	한줄평	줄거리 + 한줄평
코퍼스	모든 명사	두 음절 이상 명사	모든 명사	두 음절 이상	모든 명사	두 음절 이상 명사	줄거리 (모든 명사) + 한줄평 (두 음절 이상 명사)
SVD	500	500	500	500	500	500	500
결과	O	X	X	O	X		O
최대 유사도(%)	38	39	19	27	59		62

연구 결과 & 결론

1) 줄거리 코퍼스
- 로맨스 영화 결과 bad
- 2음절 이상 명사는 전혀 다른 장르의 영화가 추천됨
  => 어느 정도 노이즈가 필요
2) 한줄평 코퍼스
- 로맨스 영화 결과 good
- 줄거리 코퍼스보단 키워드가 비슷한 영화가 확실히 적음

적당한 노이즈를 주기 위해 줄거리와 한줄평을 각 문서로 하는 시스템6을 구현. 그 결과 다른 장르의 영화 추천 현상이 적고, 확인한 영화 중 최대 유사도가 62이며 시리즈 영화뿐만 아니라 비슷한 키워드를 가진 영화를 잘 추천해주는 것을 볼 수 있었음

accuracy가 정확히 나오는 모델이 아니어서 아쉬움이 남아 차후에 accuracy를 확인 수 있게 다른 모델을 사용하고 웹페이지에서 서비스를 사용해 볼 수 있게 구현할 예정

시스템 6 실행 결과

```
get_recommendations2('신비한 동물사전')
```

	제목	장르	키워드
1697	신비한 동물들과 그린델왈드의 범죄	모험, 가족, 판타지	#마법사 #뉴트 #마법
1011	해리 포터와 아즈카반의 죄수	판타지, 가족, 모험, 액션	#해리 #마법 #시리우스
1603	해리 포터와 죽음의 성물 - 2부	모험, 판타지, 미스터리	#볼드모트 #덤블도어 #해리

```
ons2('동주')
```

	제목	장르	키워드
	국제시장	드라마	#아버지 #시대 #덕수
	마이웨이	드라마	#청년 #타즈오 #조선
	밀정	액션	#의열단 #경성 #폭탄

```
get_recommendations2('조커')
```

	제목	장르	키워드
3307	다크 나이트	액션, 범죄, 드라마, 미스터리	#조커 #배트맨 #당시
3390	배트맨	액션, 범죄, 스릴러	#배트맨 #비키 #당시
3353	다크 나이트 라이즈	액션, 범죄, 스릴러	#배트맨 #베인 #멘트

```
endations2('다이빙벨')
```

	제목	장르	키워드
	쿠르스크	드라마	#쿠르스크 #미하일 #잠수함
	더 포스트	드라마, 스릴러	#베트남 #보도 #정부
	공범자들	다큐멘터리	#방송 #권력 #실체

참고문헌

(1) 배은영·유석종, 「키워드 기반 추천시스템 데이터 셋 구축 및 분석」, 한국정보기술학회 논문지, 2018.
(2) 유원준, 「딥 러닝을 이용한 자연어 처리 입문」, 위키독스, 2019.
(3) 이기창, 「한국어 임베딩」, 에이콘출판사, 2019.