1) 영화 추천 알고리즘 모델링

- 제한적인 사용자 리뷰 및 영화 정보를 이용한
고객 성향 분석 정보 및 영화 추천 시스템 구축 -

Introduction

네이버는 영화페이지를 유지하고 있고, 지속적으로 관리하고 있으나, 서비스는 영화에 대한 정보와 사용자가 리뷰를 입력/조회할 수 있는 것이 전부이다. 왓챠, 버즈니와 같은 경쟁사에서 제공하는 사용자 성향 분석 혹은 영화 추천과 같은 서비스는 제공하지 않고 있다.
그러나, 네이버 입장에서 개선된 서비스를 제공하지 않는 것은 다음과 같은 문제가 발생할 수 있다.

1. 데이터 보유량 격차 심화
2. 통합 서비스 플랫폼으로써의 브랜드 이미지 손실과 트래픽 감소

현재 영화 스트리밍 방면에서 소비자의 TOM에 위치한 왓챠로 몰리는 데이터 집중도는 엄청나다. 네이버 영화와 왓챠의 서비스는 차별성이 없는 동질적인 서비스인데, 추천/예측 알고리즘으로 왓챠가 우위를 점한 상태이다. 분석을 진행하는 2개월 간 왓챠의 리뷰 데이터는 천만 단위로 증가한 반면, 네이버 리뷰 데이터는 10만 단위로 증가했다. 데이터 보유량 격차는 이후 네이버가 새로운 서비스를 시작할 때, 따라잡을 수 없는 정교함의 차이로 나타날 것이다.
또, 사용자 트래픽 감소는 영화 페이지뿐 아니라 네이버의 시장 점유율에서 전반적으로 발생하는 문제이다. 네이버의 시장 점유율은 2011-77%에서 2016-50%까지 줄어든 상태이며, 그 이유 중 하나는 네이버에서 제공하는 많은 서비스가 미투 상품을 만들어내기 좋은 서비스이거나, 네이버의 서비스가 미투 상품인 경우가 대부분이기 때문이다. 시장에서의 네이버의 no.1 위치가 다방면에 있어서 적정 수준 이상의 서비스를 제공한다는 것, 한국인에 친화적인 특화 서비스를 제공한다는 것, 기존 인지도를 바탕으로 이루어진다는 것을 비추어보았을 때 격차가 벌어지는 것은 특히 치명적일 수 있다.

Introduction

네이버가 추가 서비스를 제공하지 않는 이유를 다음과 같이 추측할 수 있다.

1. 상대적으로 적은 데이터 양 (왓챠 - 3억개 이상, 네이버 - 1000만개, 2017.6 기준)
2. (서비스 실패시) 포탈 플랫폼 1인자로서의 신용도 하락

2번 문제에 대해 2억개와 1000만개의 차이를 뛰어넘을 수 있다. 모든 사용자 개인에 대한 추천과 성향 분석의 정교함은 차이가 있을 수 있으나, 이미 충분히 많은 리뷰를 작성한 사용자에 대한 정확성은 유의미한 수준으로 확보할 가능성이 있고, 부족한 정확성은 대표 포탈 사이트인 네이버의 접근성으로 상쇄할 수 있다. 게다가, 서비스를 제공함으로써 사용자의 추가 이탈을 막을 수 있고, 왓챠를 사용하지 않는 신규 사용자 유입의 가능성을 높일 수 있다.

추천 알고리즘의 프로토타입을 제작하여 충분한 시간 동안 검증한 뒤 유의 수준의 신뢰성이 확보되면 런칭하는 방법으로 2번 문제를 해결할 수 있다. 왓챠와 같은 스트리밍 서비스를 이미 제공하고 있고, 수평적으로 유사한 서비스(만화,쇼핑) 다수 운영하는 네이버 입장에서 영화 페이지의 추천/예측 데이터 전략이 성공할 경우, 다른 서비스에 미치는 연쇄적 효과를 고려하면 비용을 넘어서는 유의미한 투자라고 할 수 있다.

Introduction

이에 대해 네이버의 향후 방안은 다음과 같을 수 있다.

A. 선택과 집중 - 혁신적 서비스에 집중하고 적정 수준 이하의 대체 가능한 서비스는 폐지
B. 올라운더의 위치 고수 - 다방면의 서비스를 평균 이상의 레벨까지 끌어올린다.

네이버의 현재 시장 내 위치와 비전을 고려할 때 A를 선택하는 것은 어려우며, 이미 혁신적 서비스를 개발하기 위해 과감한 R&D 투자, 스핀오프를 감행하고 있지만, 그 성과가 사용자에게 나타나는 시기는 짐작하기 어렵다. 그런 의미에서 B 전략이 현실적이다. 영화 페이지에서 추천/고객 성향 분석 서비스를 성공하면, 다른 수평적 서비스인 웹툰/음악 등의 서비스에도 쉽게 적용할 수 있다. 이를 위한 첫 단계는 영화 서비스를 개선해 왓챠와 동질적인 서비스를 제공하는 상태에서 네이버의 플랫폼 파워로 부족한 데이터 보유량을 넘어서는 경쟁력을 갖추는 것이다.

이를 위해, 네이버 영화 페이지 추천 알고리즘을 위한 데이터 분석을 진행한다.

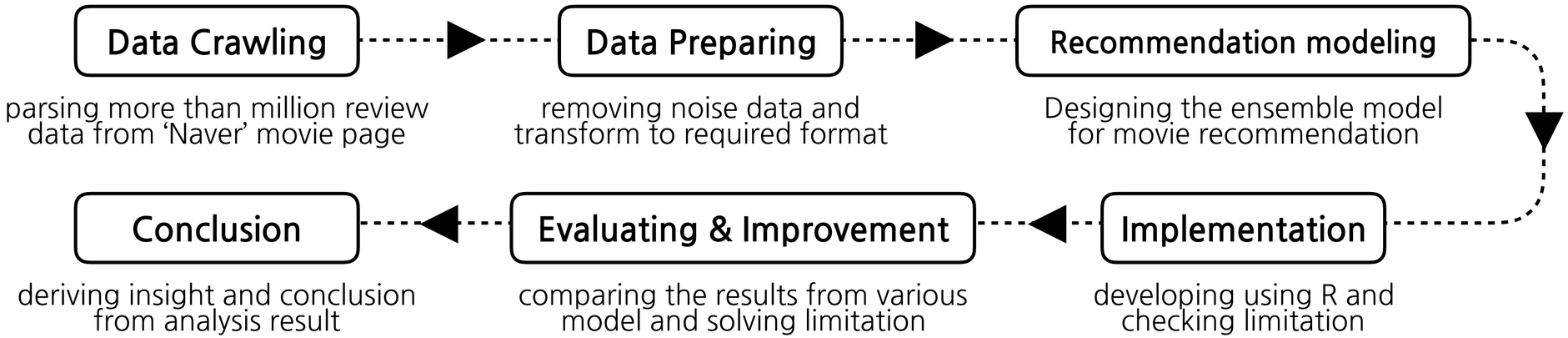
Objectives & Process

Objectives :

*** 영화 추천 : 고객의 기존 관람 영화를 기반으로 영화를 추천한다.**

\+ 텍스트 마이닝을 활용한 활용 : 스코어가 없는 영화 리뷰 점수를 통해 고객의 감정을 분석하여 추후 서비스에 활용한다.

Process :



Data Crawling

Python 라이브러리인 Beautiful Soup4를 이용하여 네이버 페이지의 웹 정보를 파싱

* http://movie.naver.com/에서 영화 정보를 수집한다.

〈Entire review page〉

〈Movie review page〉

〈User review page〉

Data Crawling

제한 사항 및 솔루션 :

A. 네이버는 페이지당 10개씩 총 1000페이지의 리뷰 정보를 제공하므로, 파싱할 수 있는 인스턴스는 1만개뿐이다.
-> user 페이지로 이동할 경우 해당 유저가 등록한 영화 리뷰 정보를 조회할 수 있다. 1000페이지에 존재하는 모든 유저 페이지로 들어가 유저가 관람한 영화 제목을 수집해 영화 리스트를 만든다. 각 영화 페이지에 리뷰를 등록한 모든 유저 페이지로 다시 들어가 해당 유저의 모든 리뷰 데이터를 크롤링한다.

B. review별로 Primary key로 사용할 수 있는 review id는 존재하지만, user id는 존재하지 않으며 user id 또한 비식별화된 정보뿐으로, 유저간 식별이 불가했다. movielover123과 moviehater456은 둘 다 movie****라는 id로 제공되므로, 실제론 다른 유저임에도 식별이 불가능하게 되는 것을 의미한다.**
-> 작업의 한 단위를 유저 1명의 모든 리뷰 데이터 파싱으로 설정한다. 한 단위의 작업을 실행할 때마다 각 유저에게 임의의 Primary key를 부여하고, 새로운 리뷰 데이터 행을 수집할 때 다음과 같이 분기하여 해결한다.
1) user id가 기존 테이블에 존재한다.
1-1) 해당 인스턴스의 review id가 존재한다. -> 중복 데이터이므로 스킵
1-2) 해당 인스턴스의 review id가 존재하지 않는다. -> 다른 유저이므로 새로운 PK를 부여하고 작업 시행
2) user id가 기존 테이블에 존재하지 않는다. -> 다른 유저이므로 새로운 PK를 부여하고 작업 시행

C. 네이버는 일정 시간이 지나면 네트워크를 차단하여 작업이 중단된다.
-> 영화 리스트의 범위를 나누어, 프로그램을 여러번 수행한다. 총 3900여개의 영화를 5개 단위로 나누어 수행하고, 목표한 리뷰 데이터 수인 100만개를 크롤링한 후 종료한다.

Data Crawling

Data table schema :

* Movie data

{title, genre, nation, time, date, grade, director, star, aud_score, cri_score, net_score}

title	genre	nation	time	date	grade	director	star	aud_score	cri_score	net_score
태평륜피안	['드라마', '액션', '전쟁']	중국	126	2016.08.18	15세 관람가	오우삼	['장쯔이', '금성무', '송혜교']	NA	7.86	NA
필라델피아	['드라마']	미국	125	1994.03.26	15세 관람가	조나단 드미	['톰 행크스', '덴젤 워싱톤']	NA	8.61	NA
닌자터틀	['모험', '액션', '코미디', '판타지', 'SF']	미국	101	2014.08.28	12세 관람가	조나단 리브스만	['메간 폭스', '피터 플로스잭', '제레미 하워드']	8.07	4.71	7.7
부러진 화살	['드라마']	한국	100	2012.01.18	15세 관람가	정지영	['안성기', '박원상', '나영희']	NA	7.61	8.92
송어	['스릴러']	한국	100	1999.11.06	12세 관람가	박종원	['강수연', '황인성', '설경구']	NA	8.22	NA
이별계약	['드라마', '멜로/로맨스']	중국	103	2013.06.20	12세 관람가	오기환	['펑위옌', '바이바이허', '장경부']	NA	5	8.37

* Review data

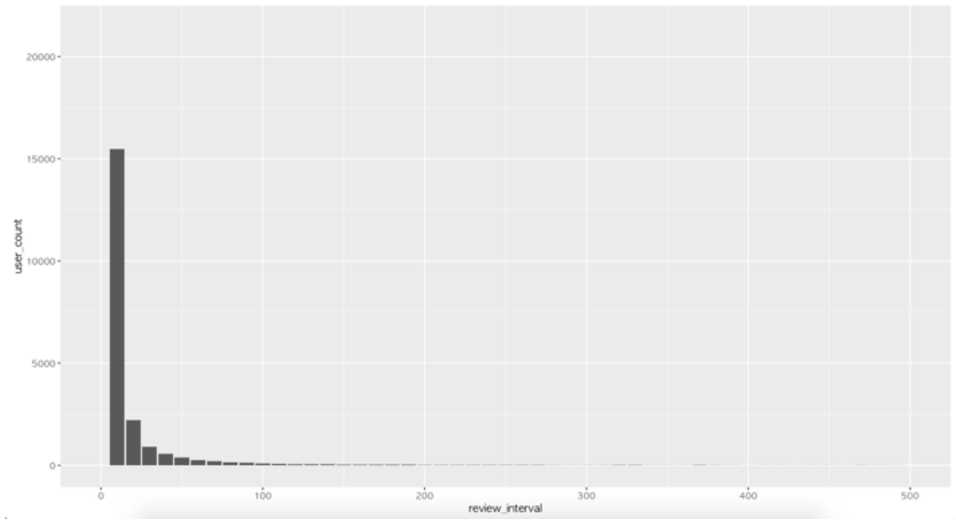
{pk, user_pk, id, review_count, title, score, content}

pk	user_pk	id	review_count	title	score	content
1	1706	thfk****	43	프리즌	5	재밌는데 지루함 멋있는데 지루함 정말 딱뭣도 아닌..난이걸얼마나 기대했던가..
2	1706	thfk****	43	23 아이덴티티	1	아진짜알바좀쓰지마세요ㅡㅡ방금보고나왔는ㄷㅔ내돈주고본영화2위임 쓰레기 1위는 미스터페레그린
3	1706	thfk****	43	트리플 엑스 리턴즈	10	후기;처음부터 끝까지 액션 진짜 액션은 진리다 끝.그래도 나쁘지않았어요ㅋㅋㅋㅋ스토리는10% 액션은90%
4	1706	thfk****	43	공조	10	진짜 현빈도현빈이지만 진짜믿고보는 유해진ㅋㅋㅋㅋ재밋어요 유해진짱짱
5	1706	thfk****	43	형	10	진짜요즘들어서 본영화중최고 럭키보다재밋음사실럭키별로ㅠㅠ 도경수넘귀엽다 진짜 끝나고둘이노래부르는거 굿 알바평점들에 속아서라도꼭봐야ㅎ
6	1706	thfk****	43	트랜스포머: 사라진 시대	10	시간가는줄보고밧어요 기대치몬한다는 글들을 많이봐서미루다 미루다본건데요 전 기대이상으로 잘밧네요

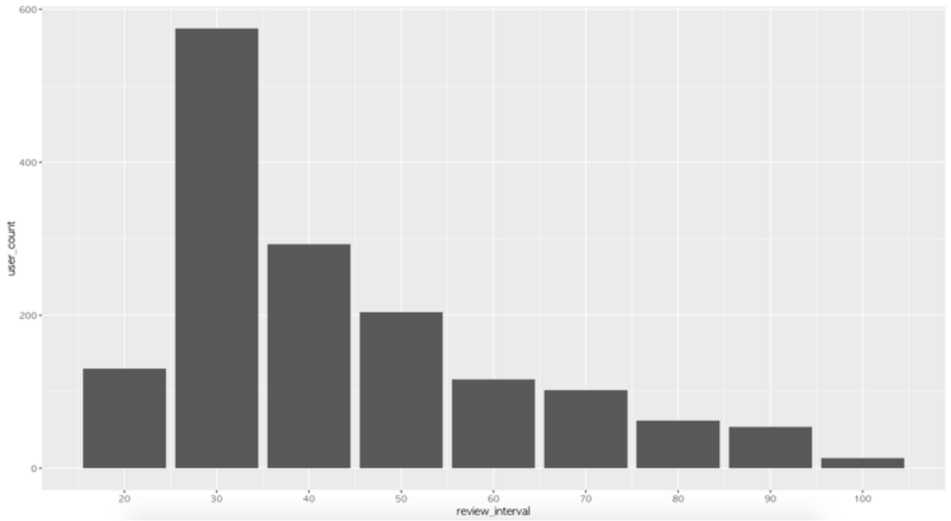
Data Preparing

Review data 수에 따른 회원수 분포 :

샘플링 전 유저 분포는 불규칙하고 review가 10개 미만인 회원의 수가 과반수 이상이라 정확성이 떨어질 것으로 예상되었다. 따라서 결과의 유효성을 높이기 위해 review의 상한/하한선을 정하고 그에 따라 샘플링 후 분석을 진행하였다.



〈샘플링 전〉



〈샘플링 후〉

Data Preparing

1) Movie data 확인
영화는 Web parsing 단계에서 PK를 설정하지 못했기 때문에 영화 제목이 중복되는 경우 영화 간 식별이 어려워진다. 제한적인 방법이지만, 중복되는 제목이 있는 경우 가장 최신의 영화를 남겨두고 관련된 항목을 제거했다.

2) Review data 확인
2-1) 각 유저 정보 내에 중복된 항목이 있는지 확인하여 중복항 제거
2-2) 중복항 제거 후 review_count 열 update
* 총 130만 건의 데이터를 수집했으나, 130만 건은 데이터를 읽어들이는 동안에도 프로그램이 다운되었다. 우선 약 10만 건의 데이터만 선별하여 분석을 진행하고 결과에 따라 추후 다른 방법으로 데이터 양을 확장한다.

3) 각 항목에서 string 타입 중 numeric으로 바꿀 수 있는 데이터 형 전환

4) 영화 특징 정보 가공
4-1) 영화 정보에서 수집한 모든 영화의 장르/국가/등급 정보를 추출한다.
4-2) 유저의 인구통계학적 정보나, 영화의 다른 정보를 얻기 어려운 상황에서 유일하게 회원의 성향이나 영화의 특징을 의미하는 정보는 각 영화의 장르/국가/등급에 대한 정보이다. 그러므로, 이 후 분석을 용이하게 하기 위해 각 영화별 장르/국가/등급 정보를 추출해 분석하기 쉬운 형태로 가공한다.
4-3) 영화와 영화의 등급 및 국가는 1:N 대응이지만, 영화와 장르는 N:M 대응이다. 그러므로, 영화가 포함하는 장르만큼 추가 열을 만들어 One-hot 코딩

Data Preparing

5) 장르/국가/등급 정보의 변환

5-1) 각 회원별로 카테고리별 영화의 평균 평점과 표준편차, 샘플수를 기록하는 테이블을 만든다.
eg. 회원 A의 SF 장르에 속하는 영화의 평균 평점, 표준편차, 관람(작성)한 리뷰 수

5-2) 앞서 구한 카테고리별 항목의 평균/표준편차/샘플 수를 이용해 각 항목에 대한 선호도/신뢰도를 도출한다.
* 선호도 : 카테고리의 각 항목에 대한 평균 평점을 카테고리 내 전체 항목의 평균과 표준편차로 정규화하여 0~1 사이의 값으로 전환한다. 이를 통해, 전체 항목의 평균 점수 중 한 항목 평균 점수의 상대적 위치를 나타낼 수 있다.
* 신뢰도 : Document Frequency를 이용한다. (특정 항목 샘플수/전체 샘플수)는 특정 항목의 비율을 나타내는데, log를 이용해 그 차이를 줄이고 표준화하여 0~1의 값으로 전환한다.
cf. DF('특정 항목 샘플수/전체 샘플수')는 회원이 특정 항목의 영화를 볼 확률을 의미한다.
cf. 100개의 영화를 보고 그 중 80개의 액션 영화를 본 회원 A와 100개의 영화 중 60개의 액션 영화를 본 회원 B의 평점은 신뢰도의 차이는 존재하지만, 그 차이가 크지 않을 것이라 가정하여 log를 사용하였다.

5-3) 이 후의 기법은 항목에 대한 평균 평점이 아닌 앞서 구한 선호도를 기반으로 분석한다.

6) Dividing into Training set & Test set

초기 10만 건 선별 후, 가공 작업을 거쳐 9.2만 건의 데이터가 남았다. 이 데이터를 8:2로 training data와 test data로 나눈다. 각 유저의 review data 중 임의로 20%의 영화를 추출하여 test set으로, 남은 데이터는 training set으로 분류한다.

Movie data analysis

User tendency

1) k-means clustering : 모든 유저를 카테고리(장르/국가/등급)에 대한 선호도를 바탕으로 클러스터링한 뒤, 동일 클러스터 내의 유저들이 본 영화 중 가장 빈도가 높은 영화 목록을 만든다. 해당 목록에서 각 유저가 본 (training set에 존재하는) 영화를 제외한 목록을 생성한다. 각 영화에 대한 해당 클러스터 내 회원의 평균 평점도 함께 기록한다.

* 클러스터링은 k=15, k=sqrt(회원수)=40으로 2번 진행한다.
* 장르/국가/등급별로 나누어 클러스터링을 진행한다. (총 6번)

2) Collaborative Filtering : 카테고리 내 항목의 선호도를 바탕으로 회원 간 거리를 구한다. 거리는 cosine 거리와 Euclidean 거리를 이용해 각각 100명의 회원을 뽑은 뒤 그 교집합을 이용한다. (교집합의 크기가 0일 경우 Euclidean 거리를 적용한다.) 이렇게 뽑은 100명의 회원이 본 영화 중 가장 빈도가 높은 영화 목록을 만든다. 해당 목록에서 각 유저가 본(training set에 존재하는) 영화를 제외한 목록을 생성한다. 각 영화에 100명의 회원의 대한 평균 평점도 함께 기록한다.

* 장르/국가/등급별로 나눈 경우와 3가지를 모두 합친 경우의 CF를 진행한다. (총 4번)

Recommendation Model

실제로 parsing한 130만개의 data 중, 각 user별 review data의 수는 적게는 5개 미만에서 많게는 500개 이상까지 편차가 크다. 샘플링한 data에서도 전체 데이터 인스턴스는 9.2만개지만, user별 review data의 수는 적게는 20개에서 많게는 100개에 달한다. Supervised learning을 할 경우 각 user에게 추천할 수 있는 모델을 만들어내야 하는데, 적은 수의 review data를 가지고 있는 user 수의 분포를 고려하면 과반수 이상의 user는 모델을 훈련시킬 data가 충분하지 않았다. 100개가 넘는 유저에 대해선 NN을 활용할 수 있겠지만 그 정확도가 높지 않을 것이며, 그 이하인 유저의 분포가 더 많으므로 적절한 방법은 아니라고 판단했다. 따라서, supervised learning은 피하고 모델을 훈련시킬 필요가 없이 제한된 정보로 유효한 결과를 얻을 수 있는 기법을 주로 사용했다.

Recommendation Model

모델은 다음의 4가지 방법으로 접근했다.

1) k-means clustering : 모든 유저를 카테고리(장르/국가/등급)에 대한 선호도를 바탕으로 클러스터링한 뒤, 동일 클러스터 내의 유저들이 본 영화 중 가장 빈도가 높은 영화 목록을 만든다. 해당 목록에서 각 유저가 본 (training set에 존재하는) 영화를 제외한 목록을 생성한다. 각 영화에 대한 해당 클러스터 내 회원의 평균 평점도 함께 기록한다.

* 클러스터링은 k=15, k=sqrt(회원수)=40으로 2번 진행한다.
* 장르/국가/등급별로 나누어 클러스터링을 진행한다. (총 6번)

2) Collaborative Filtering : 카테고리 내 항목의 선호도를 바탕으로 회원 간 거리를 구한다. 거리는 cosine 거리와 Euclidean 거리를 이용해 각각 100명의 회원을 뽑은 뒤 그 교집합을 이용한다. (교집합의 크기가 0일 경우 Euclidean 거리를 적용한다.) 이렇게 뽑은 100명의 회원이 본 영화 중 가장 빈도가 높은 영화 목록을 만든다. 해당 목록에서 각 유저가 본(training set에 존재하는) 영화를 제외한 목록을 생성한다. 각 영화에 100명의 회원의 대한 평균 평점도 함께 기록한다.

* 장르/국가/등급별로 나눈 경우와 3가지를 모두 합친 경우의 CF를 진행한다. (총 4번)

Recommendation Model

3) Association Rule : 각 회원별로 관람한 영화를 하나의 transaction으로 가정하여, Association Rule 분석을 진행한다. 9.2만개의 데이터는 1594명의 회원의 영화 리뷰 데이터이므로, 1594개의 거래 내역이 발생한다. support=0.1, confidence=0.4로 설정한 규칙을 생성한다. 생성된 규칙 중 회원의 관람 영화(training set 내의 영화)가 lhs에 포함되면서 rhs에는 포함되지 않는 규칙을 선별하고, 각 규칙의 rhs에 해당하는 영화 목록을 기록한다.

4) Contents Based Recommendation : 유저가 관람한 영화의 카테고리 항목별 평균 점수를 이용해 n개의 항목 차원 내에서 유저의 위치를 구한다. 카테고리의 항목을 이용해 해당 유저와 가장 거리가 가까운 영화 n개의 목록을 생성한다.

* 장르/국가/등급의 각 속성은 One-hot 코딩을 거쳤으므로 0과 1로 표현된다. 이 경우, Euclidean이나 Jaccard 거리를 쓰면, 다른 항목은 모두 0이고 1가지만 1인 영화가 가장 가까울 확률이 높아진다. 그러므로 여기서 cosine 거리를 이용하여 유사도를 측정한다.

Recommendation Model

* 각각의 개별적인 모델로 이루어진 추천 엔진과 앙상블 모델을 모두 구성하여 결과를 평가한다. 앙상블은 각각의 모델을 아래 3가지 방법으로 조합하여 구성한다.

union

: 선택된 추천 목록의 합집합을 구한다. 추천되는 목록이 수십개에서 최대 수백개까지 넘어갈 수 있으므로 union을 통해 생성된 목록을 추천 엔진에 사용하는 것은 무리가 있지만, union 전의 결과와 union 후의 결과를 비교함으로써 각 추천 목록의 중복 정도를 평가할 수 있다.

intersect

: 선택된 추천 목록의 교집합을 구한다. 추천되는 목록이 최소 1개에서 최대 30개까지로 추천 엔진을 위한 범위로는 가장 적절하지만, 교집합으로 가장 보수적인 목록을 제공하기 때문에 예측률이 낮아질 리스크가 비교적 높다.

frequency

: 선택된 추천 목록으로 frequency table을 구성하여 frequency가 높은 상위 30개 목록을 구한다. 예측률이 낮아질 리스크도 비교적 낮으며, 목록의 개수는 30개로 고정되므로 추천 엔진에 사용하기에도 적합하지만, union보다 예측률이 낮을 가능성이 높다.

Recommendation Model

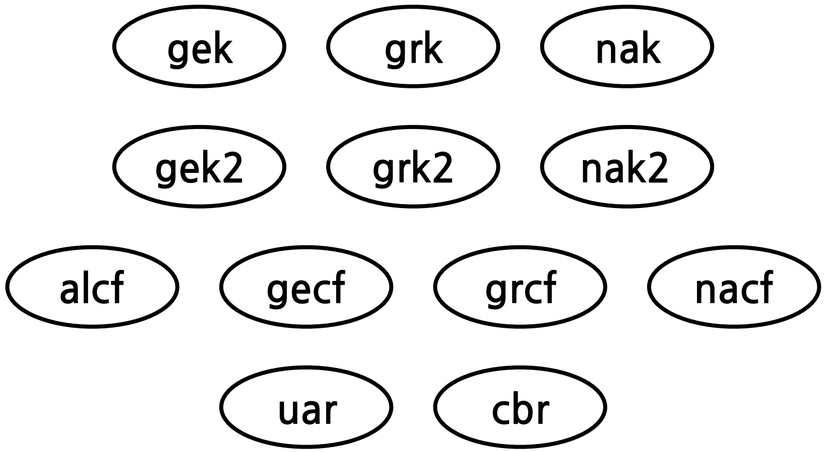
*** Reference model**
rar : random sampled movies
scr : top 30 scored movies
cor : top 30 reviewed movies

*** Single Analysis model**
alcf : all_collaborative_filtering
gecf : genre_collaborative_filtering
grcf : grade_collaborative_filtering
nacf : nation_collaborative_filtering
gek : genre_k_means, k=15
grk : grade_k_means, k=15
nak : nation_k_means, k=15
gek2 : genre_k_means, k=40
grk2 : grade_k_means, k=40
nak2 : nation_k_means, k=40
uar : association_rule
ccbr : cosine_contents_based_recommendation

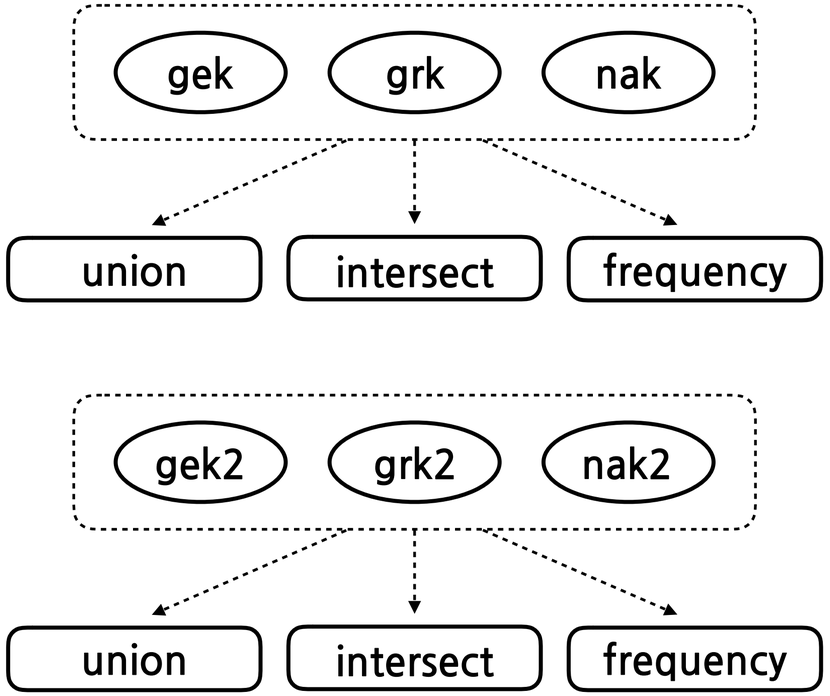
*** Ensemble Analysis model**
k_m_inter : intersect(gek, grk, nak)
k_m_union : union(gek, grk, nak)
k_m_freq : frequency(gek, grk, nak)
k_m2_inter : intersect(gek2, grk2, nak2)
k_m2_union : union(gek2, grk2, nak2)
k_m2_freq : frequency(gek2, grk2, nak2)
k_m_all_inter : intersect(gek, grk, nak, gek2, grk2, nak2)
k_m_all_union : union(gek, grk, nak, gek2, grk2, nak2)
k_m_all_freq : frequency(gek, grk, nak, gek2, grk2, nak2)
cf_inter : intersect(alcf, gecf, grcf, nacf)
cf_union : union(alcf, gecf, grcf, nacf)
cf_frequency : frequency(alcf, gecf, grcf, nacf)
all_inter : intersect(all)
all_union : union(all)
all_freq : frequency(all)

Recommendation Model

* Single model

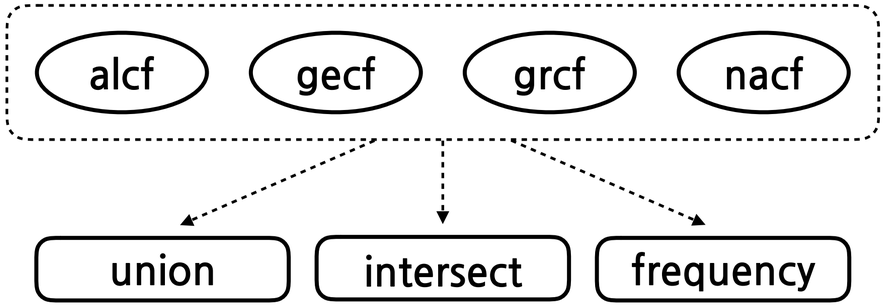


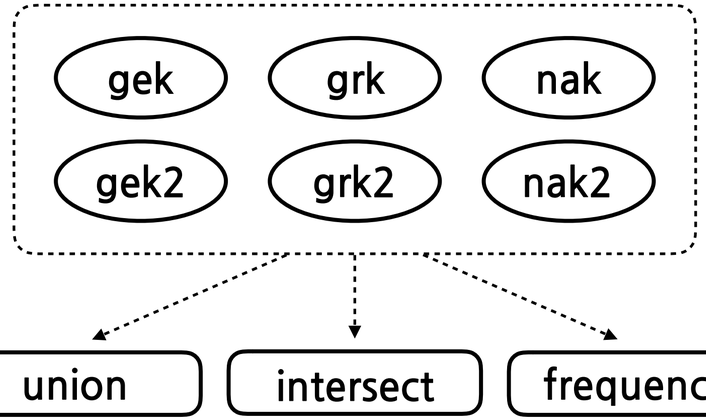
* Ensemble model

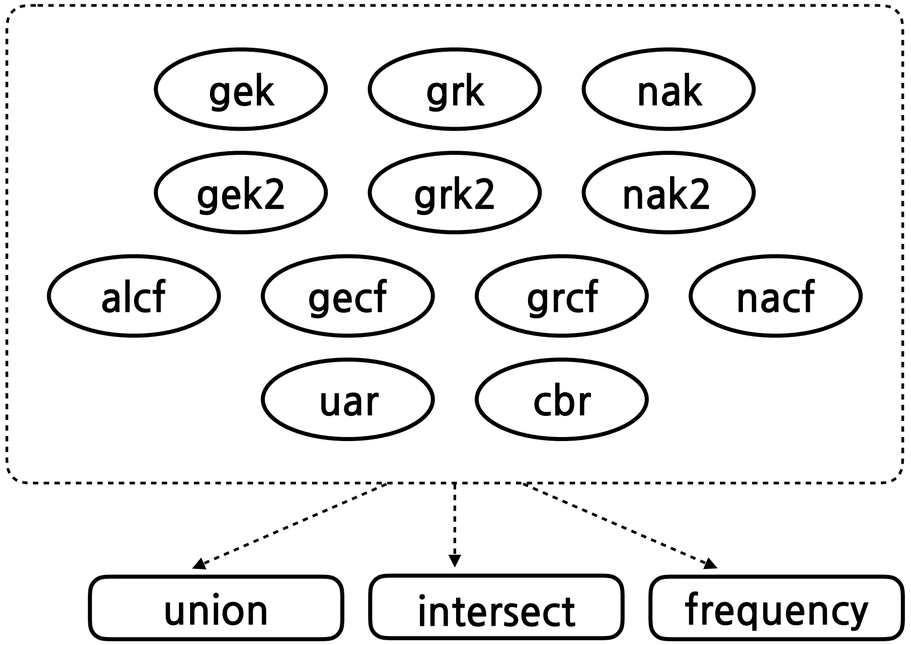


Recommendation Model

* Ensemble model







Implementation

* 소스코드는 깃헙에서 확인 가능
https://github.com/bigshanedogg/movie_recommend

Implementation

* 제한 사항 :

1) 처리 가능한 데이터 양의 제한
- 최초 데이터는 130만 건을 수집하였으나, 컴퓨터 처리 능력의 한계 때문에 실질적으로 4~10만 건의 데이터만 사용이 가능했다. (50만 건까지는 데이터를 읽어들이는 것까지 겨우 가능했고, 100만 건은 읽어들이는 것조차 불가능)
- 분석이 가능한 최대 개수인 10만 건으로 진행했으나, 한 유저당 유의미한 결과를 도출하기 위해 최소 리뷰 수를 25개로 제한한 결과, 10만 건에 속하는 회원 수는 2000명 미만이었다.

-> 컴퓨터 처리 능력 문제를 극복하고 130만 건의 데이터를 모두 활용한다면 더 유의미한 결과를 도출할 수 있을 것으로 예상된다.

2) 정보 범위의 제한
- 회원의 특성과 영화의 특성을 구분할 수 있는 정보로 장르/국가/등급를 이용했으나, 국가는 미국/한국이 데이터의 80% 이상을 차지하고 있었고 등급의 경우 논리적으로 크게 유의미하다고 보기 어렵다.
- 회원 정보가 비식별화되어 있기 때문에 인구통계학적 정보를 구할 수 없었다. 인구통계학적 정보는 회원 특성을 파악하는데 위의 장르/국가/등급 만큼이나 큰 영향을 끼쳤을 것으로 예상된다.
- 영화 자체에 대한 정보도 장르만으로는 신뢰도가 떨어진다. 예를 들어, 스파이더맨과 배트맨의 경우 둘 다 액션,히어로/미국/15세이상에 속하지만, 장르/국가/등급 정보만으로는 두 영화를 구분하기 어렵다. 이를 해결하기 위해선 영화 줄거리의 텍스트를 분석해야 한다. 줄거리에서 키워드를 뽑아내고, TF-IDF로 불필요한 word를 제거할 경우 '거미/MARVEL', '박쥐/DC'와 같은 키워드를 추출하여 두 영화를 구분할 수 있을 것이다.

-> 만약 네이버 데이터베이스를 직접 활용해 추천 서비스를 설계한다면 이 한계는 상당부분 극복될 것이다. 우선 인구통계학적 정보를 사용할 수 있을 것이고, 영화 줄거리를 분석하기 위한 자원도 충분히 활용할 수 있을 것으로 예상된다.

Model Evaluating

*** 각 항목 정보**

predicted_mean : 추천한 영화 개수의 평균
predicted_min : 추천한 영화 개수의 최소값
predicted_max : 추천한 영화 개수의 최대값
intersect_mean : 추천이 적중한 영화 개수의 평균
intersect_min : 추천이 적중한 영화 개수의 최소값
intersect_max : 추천이 적중한 영화 개수의 최대값
actual_mean : test data로 분류된 영화 개수의 평균
actual_min : test data로 분류된 영화 개수의 최소값
actual_max : test data로 분류된 영화 개수의 최대값
length(!=0) : 전체 user 중 1개도 적중하지 못한 user수
length(==0) : 전체 user 중 1개도 적중하지 못한 user의 비율
0_ratio : 전체 user 중 1개도 적중하지 못한 user의 비율
rate_mean : test data로 분류된 영화 중 추천한 영화가 포함된 비율의 평균
rate(>0) : 1개도 적중하지 못한 user를 제외한 test data로 분류된 영화 중 추천한 영화가 포함된 비율의 평균
rate_max : test data로 분류된 영화 중 추천한 영화가 포함된 비율의 최대값
rate_min : test data로 분류된 영화 중 추천한 영화가 포함된 비율의 최소값
NaN : 정확도 계산시 오류가 있는지 여부
ints_length : 전체 test 항목 중 추천이 적중한 영화의 수
pre_ac_cor : 예상 고객 평점과 실제 평점의 상관 관계
mean_abs : 예상 고객 평점과 실제 평점 차이의 절대값
pre_ac_rate : 예측한 영화의 수/실제 유저가 review를 남긴 test 데이터의 수/10

* 해당 적중률을 달성하기 위해 몇 번의 추천을 시도했는지를 의미하는 지표이므로, 이 값이 높을수록 실질적인 정확도가 낮아진다고 판단할 수 있다.

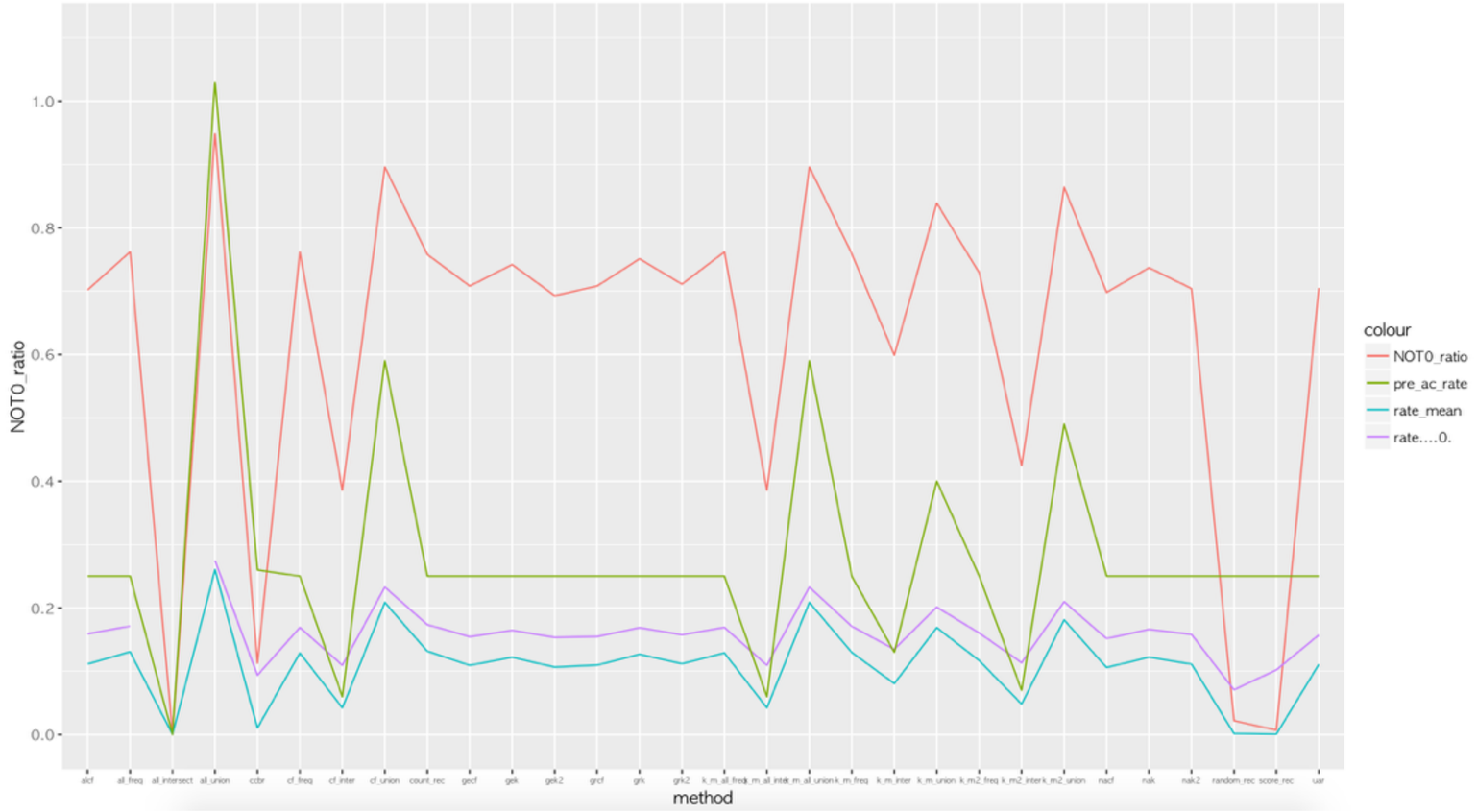
NOT0_ratio : 1 - 0_ratio

Model Evaluating

method	predicted_mean	predicted_min	predicted_max	intersect_mean	intersect_min	intersect_max	actual_mean	actual_min	actual_max	length....0.	length.....0.	X0_ratio	rate_mean	rate....0.	rate_max	rate_min	NaN.	ints_length	pre_ac_cor	mean_abs	pre_ac_rate	NOT0_ratio
random_rec	30	30	30	0.0225952227243383	0	2	11.8850871530019	5	27	34	1515	0.978	0.00155301339117058	0.070753463027154	0.142857142857143	0	N	35	0.110347457773142	1.16042295016478	0.25	0.022
score_rec	30	30	30	0.00710135571336346	0	1	11.8850871530019	5	27	11	1538	0.993	0.00072582356313796	0.102209154481882	0.142857142857143	0	N	11	NA	NA	0.25	0.00700000000000001
count_rec	30	30	30	1.57456423499032	0	9	11.8850871530019	5	27	1174	375	0.242	0.131564668209828	0.173589157629492	0.666666666666667	0	N	2439	0.27910733272593	2.05712535950359	0.25	0.758
alcf	30	30	30	1.33376371852808	0	8	11.8850871530019	5	27	1087	462	0.298	0.111606174819365	0.159041365956943	0.625	0	N	2066	0.255865671723796	1.98962716196312	0.25	0.702
gecf	30	30	30	1.31891542930923	0	7	11.8850871530019	5	27	1096	453	0.292	0.109377438556641	0.154585449200946	0.571428571428571	0	N	2043	0.235344984642763	2.01798341179894	0.25	0.708
grcf	30	30	30	1.3208521626856	0	8	11.8850871530019	5	27	1097	452	0.292	0.109683461646091	0.154876647301545	0.555555555555556	0	N	2046	0.286531791470615	2.00694398601137	0.25	0.708
nacf	30	30	30	1.25758553905746	0	8	11.8850871530019	5	27	1081	468	0.302	0.105828232141103	0.151644710070727	0.625	0	N	1948	0.220307691939921	2.04754183835937	0.25	0.698
gek	30	30	30	1.46868947708199	0	9	11.8850871530019	5	27	1150	399	0.258	0.122132683563672	0.164507414643589	0.571428571428571	0	N	2275	0.280632686231248	2.01347563709248	0.25	0.742
gek2	30	30	30	1.29180116204003	0	7	11.8850871530019	5	27	1074	475	0.307	0.106488198208259	0.153584933914891	0.571428571428571	0	N	2001	0.257256205143774	2.02111388199694	0.25	0.693
grk	30	30	30	1.51517107811491	0	8	11.8850871530019	5	27	1164	385	0.249	0.12668144521433	0.168582095048967	0.555555555555556	0	N	2347	0.255646964946259	2.05441555971323	0.25	0.751
grk2	30	30	30	1.34409296320207	0	7	11.8850871530019	5	27	1101	448	0.289	0.112011926428355	0.157589894675315	0.5	0	N	2082	0.251339792472425	2.00594207751602	0.25	0.711
nak	30	30	30	1.45384118786314	0	8	11.8850871530019	5	27	1141	408	0.263	0.122259267200424	0.165976866690146	0.615384615384615	0	N	2252	0.249474112056879	2.01519947723155	0.25	0.737
nak2	30	30	30	1.32020658489348	0	8	11.8850871530019	5	27	1090	459	0.296	0.111327429633071	0.158207512386813	0.571428571428571	0	N	2045	0.223462882820464	2.05599629951821	0.25	0.704
uar	30	30	30	1.32472562943835	0	8	11.8850871530019	5	27	1092	457	0.295	0.110837611021568	0.157222948234806	0.571428571428571	0	N	2052	0.226995316078405	1.97243494718376	0.25	0.705
ccbr	31	31	31	0.123950936087799	0	3	11.8850871530019	5	27	175	1374	0.887	0.0105995255890112	0.0938209436421616	0.3	0	N	192	-0.106992972126735	3.09145361529168	0.26	0.113
k_m_inter	15.688186	6	24	0.971594577146546	0	8	11.8850871530019	5	27	928	621	0.401	0.0806791631072862	0.134668128936623	0.444444444444444	0	N	1505	0.226524137000863	2.00503726325237	0.13	0.599
k_m_union	47.347966	37	64	2.01613944480031	0	9	11.8850871530019	5	27	1299	250	0.161	0.168794655180149	0.201280154637453	0.666666666666667	0	N	3123	0.27146736275739	2.059287818800153	0.4	0.839
k_m_freq	30	30	30	1.54938670109748	0	8	11.8850871530019	5	27	1175	374	0.241	0.129631039171103	0.170892323128447	0.6	0	N	2400	0.271111074909766	2.03190480829142	0.25	0.759
k_m2_inter	8.743706	1	17	0.577792123950936	0	5	11.8850871530019	5	27	658	891	0.575	0.0482469933704622	0.113578408405541	0.428571428571429	0	N	895	0.208894650560089	2.03838808592087	0.07	0.425
k_m2_union	58.727566	45	74	2.17753389283409	0	11	11.8850871530019	5	27	1338	211	0.136	0.181156640790318	0.209724691019583	0.636363636363636	0	N	3373	0.263062025972159	2.04133764415025	0.49	0.864
k_m2_freq	30	30	30	1.39703034215623	0	8	11.8850871530019	5	27	1129	420	0.271	0.11685732016 13	0.160329485323166	0.571428571428571	0	N	2164	0.248784471207717	2.02533985514868	0.25	0.729
k_m_all_inter	7.233699	0	17	0.508715300193673	0	5	11.8850871530019	5	27	598	951	0.614	0.042268635760317	0.109488489619952	0.375	0	N	788	0.18996688703154	2.01746853478986	0.06	0.386
k_m_all_union	69.821821	52	92	2.50355067785668	0	12	11.8850871530019	5	27	1388	161	0.104	0.208628253638302	0.232827928592024	0.75	0	N	3878	0.27146736275739	2.05692243072615	0.59	0.896
k_m_all_freq	30	30	30	1.53712072304713	0	8	11.8850871530019	5	27	1180	369	0.238	0.128811193702049	0.16909198224108	0.571428571428571	0	N	2381	0.262139975051168	2.03977140286775	0.25	0.762
cf_inter	7.233699	0	17	0.508715300193673	0	5	11.8850871530019	5	27	598	951	0.614	0.042268635760317	0.109488489619952	0.375	0	N	788	0.18996688703154	2.01746853478986	0.06	0.386
cf_union	69.821821	52	92	2.50355067785668	0	12	11.8850871530019	5	27	1388	161	0.104	0.208628253638302	0.232827928592024	0.75	0	N	3878	0.27146736275739	2.05692243072615	0.59	0.896
cf_freq	30	30	30	1.53712072304713	0	8	11.8850871530019	5	27	1180	369	0.238	0.128811193702049	0.16909198224108	0.571428571428571	0	N	2381	0.262139975051168	2.03977140286775	0.25	0.762
all_intersect	0.000646	0	1	0	0	0	11.8850871530019	5	27	0	1549	1	0	NA	0	0	N	0	NA	NA	0	0
all_union	122.706262	100	150	3.10974822466107	0	13	11.8850871530019	5	27	1468	81	0.052	0.260288337659569	0.274650296345145	0.875	0	N	4817	0.27146736275739	2.06058835866008	1.03	0.948
all_freq	30	30	30	1.55842479018722	0	9	11.8850871530019	5	27	1181	368	0.238	0.130507373558078	0.171173515361103	0.571428571428571	0	N	2414	0.250981527649256	2.0284893249615	0.25	0.762

<분석 모델 결과 비교 테이블 (개선 전)>

Model Evaluating



<분석 모델 결과 비교 그래프 (개선 전)>

Model Evaluating

* 제한 사항 :

1) 실제 항목 대비 많은 영화 추천수

- 각 모델별로 추천 영화수가 많게는 120개에서 적게는 0개까지 불규칙하다. 실제 왓챠 사용 경험이 있는 사용자를 대상으로 focused group interview를 진행해본 결과 30~60개까지의 추천을 적정하다는 의견을 얻었다. 현재 30~60개에 해당하는 모델 중 가장 높은 정확도는 k-means를 사용한 k_m2_union 모델로 18.1%이다.

2) 정확도 개선의 가능성 존재

- 적정한 수의 영화를 추천하는 k_m2_union 모델은 0_ratio가 13.6, 정확도가 18.1%이다. 장르/국가/등급을 분석할 때 k-means 하나만 이용한 뒤 앙상블한 모델인데 비해, 의도적으로 다양한 기법으로 분석한 모델을 다시 앙상블할 경우 정확도가 향상될 가능성이 있다고 판단했다.

Model Improvement

*** Improved Ensemble model**

* sampling = 30

- 30_u_freq : frequency(union(kmeans), union(cf),union(uar))
- 30_c_freq : frequency(frequency(kmeans), frequency(cf),frequency(uar))
- 30_u_freq_wo_uar : frequency(union(kmeans), union(cf))
- 30_c_freq_wo_uar : frequency(frequency(kmeans), frequency(cf))

- 30_u_freq_wo_uar_km2 : frequency(union(kmeans), union(cf))
- 30_c_freq_wo_uar_km2 : frequency(frequency(kmeans), frequency(cf))

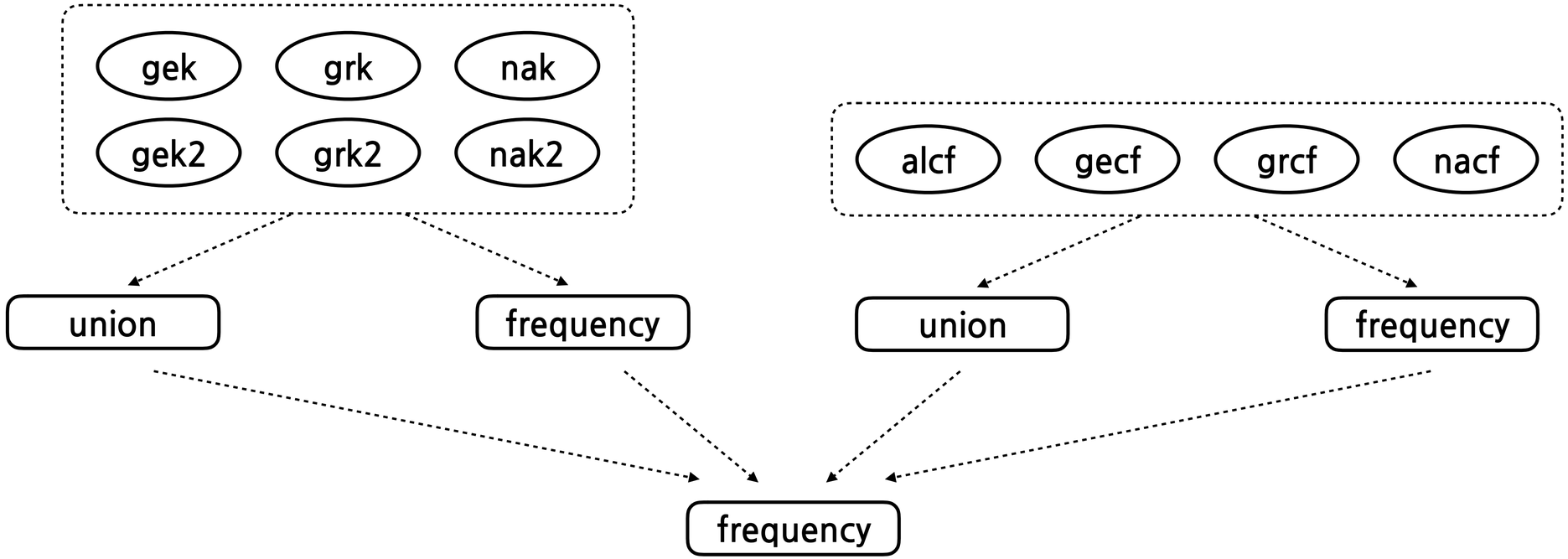
* sampling = 60

- 60_u_freq : frequency(union(kmeans), union(cf),union(uar))
- 60_c_freq : frequency(frequency(kmeans), frequency(cf),frequency(uar))
- 60_u_freq_wo_uar : frequency(union(kmeans), union(cf))
- 60_c_freq_wo_uar : frequency(frequency(kmeans), frequency(cf))

- 60_u_freq_wo_uar_km2 : frequency(union(kmeans), union(cf))
- 60_c_freq_wo_uar_km2 : frequency(frequency(kmeans), frequency(cf))

Model Improvement

* Improved Ensemble model

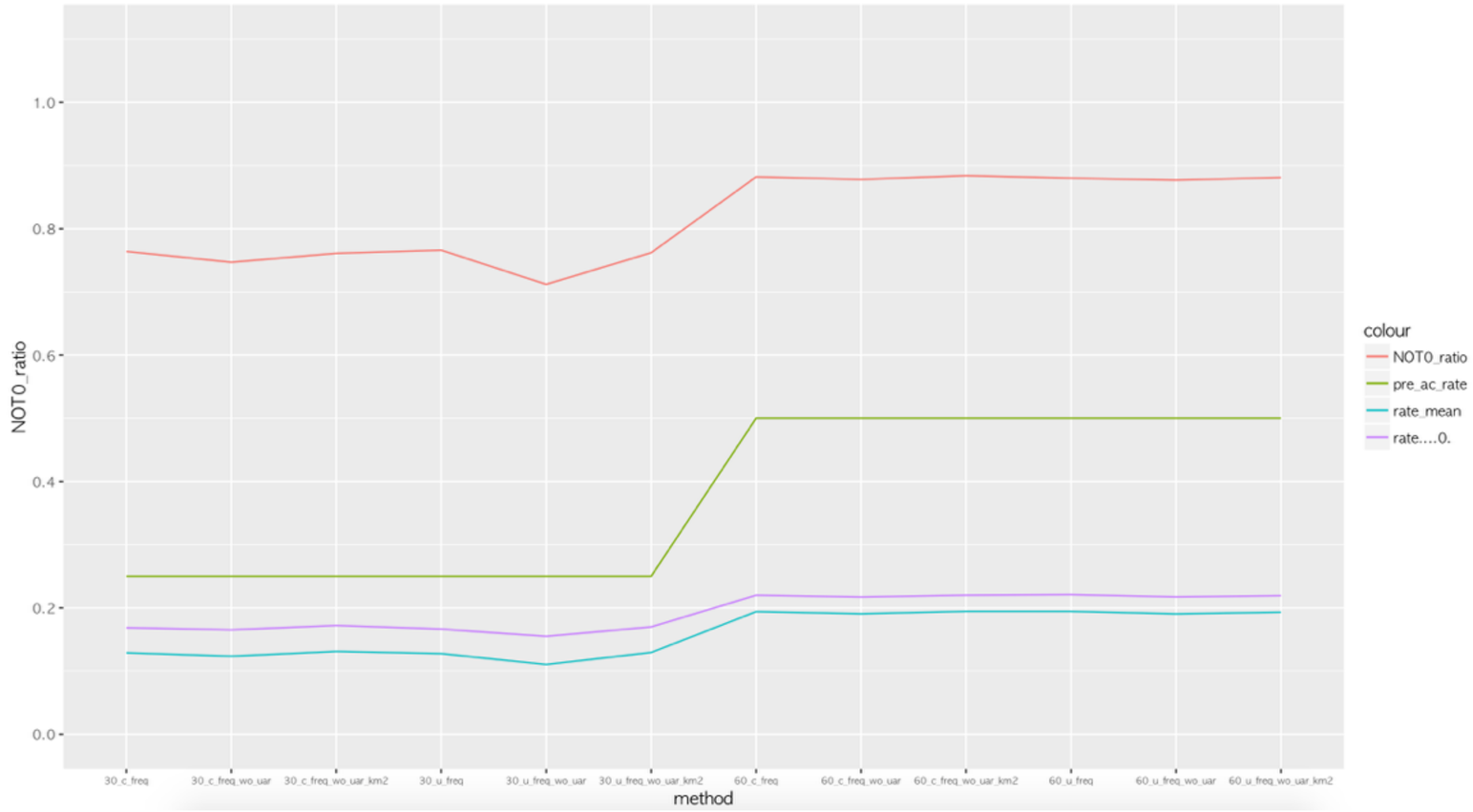


Model Improvement

method	predicted_mean	predicted_min	predicted_max	intersect_mean	intersect_min	intersect_max	actual_mean	actual_min	actual_max	length....0.	length.....0.	X0_ratio	rate_mean	rate....0.	rate_max	rate_min	NaN.	ints_length	pre_ac_cor	mean_abs	pre_ac_rate	NOT0_ratio
30_u_freq	30	30	30	1.52162685603615	0	10	11.8850871530019	5	27	1187	362	0.234	0.127494267518173	0.166376259802569	0.571428571428571	0	N	2357	0.257036243464243	2.01814758420525	0.25	0.766
30_c_freq	30	30	30	1.53389283408651	0	9	11.8850871530019	5	27	1184	365	0.236	0.128757994839652	0.168451126694781	0.571428571428571	0	N	2376	0.25852201691173	2.02809960823169	0.25	0.764
30_u_freq_wo_uar	30	30	30	1.30277598450613	0	9	11.8850871530019	5	27	1103	446	0.288	0.110343429044788	0.154960989655826	0.666666666666667	0	N	2018	0.241999255847356	2.0481015629717	0.25	0.712
30_c_freq_wo_uar	30	30	30	1.47127178825048	0	9	11.8850871530019	5	27	1157	392	0.253	0.123329242575126	0.165114085349067	0.571428571428571	0	N	2279	0.266363446307965	2.0466616080658	0.25	0.747
30_u_freq_wo_uar_km2	30	30	30	1.536475145255	0	8	11.8850871530019	5	27	1180	369	0.238	0.129318153673045	0.169757474609786	0.666666666666667	0	N	2380	0.256063666544546	2.02438677814395	0.25	0.762
30_c_freq_wo_uar_km2	30	30	30	1.56229825693996	0	8	11.8850871530019	5	27	1179	370	0.239	0.130945810569711	0.172039915667924	0.6	0	N	2420	0.272789094317536	2.00726666089393	0.25	0.761
60_u_freq	60	60	60	2.31762427372498	0	10	11.8850871530019	5	27	1363	186	0.12	0.194358757342144	0.220881669202481	0.666666666666667	0	N	3590	0.256390493443003	2.08204105510041	0.5	0.88
60_c_freq	60	60	60	2.31504196255649	0	10	11.8850871530019	5	27	1366	183	0.118	0.193924987161998	0.219904688956028	0.666666666666667	0	N	3586	0.254844776251969	2.09755432012893	0.5	0.882
60_u_freq_wo_uar	60	60	60	2.27437056165268	0	11	11.8850871530019	5	27	1358	191	0.123	0.190429282124811	0.217212782040746	0.666666666666667	0	N	3523	0.253662492886792	2.07577979088902	0.5	0.877
60_c_freq_wo_uar	60	60	60	2.27630729502905	0	10	11.8850871530019	5	27	1360	189	0.122	0.190571486452663	0.217055318025864	0.666666666666667	0	N	3526	0.252918405832102	2.10177398630005	0.5	0.878
60_u_freq_wo_uar_km2	60	60	60	2.29502905100065	0	11	11.8850871530019	5	27	1364	185	0.119	0.193012012865381	0.21919032839331	0.75	0	N	3555	0.263170686975229	2.1124354710454	0.5	0.881
60_c_freq_wo_uar_km2	60	60	60	2.31310522918012	0	10	11.8850871530019	5	27	1369	180	0.116	0.194423073949181	0.219986370743084	0.714285714285714	0	N	3583	0.266952652467159	2.10860852011409	0.5	0.884

<분석 모델 결과 비교 테이블 (개선 후)>

Model Improvement



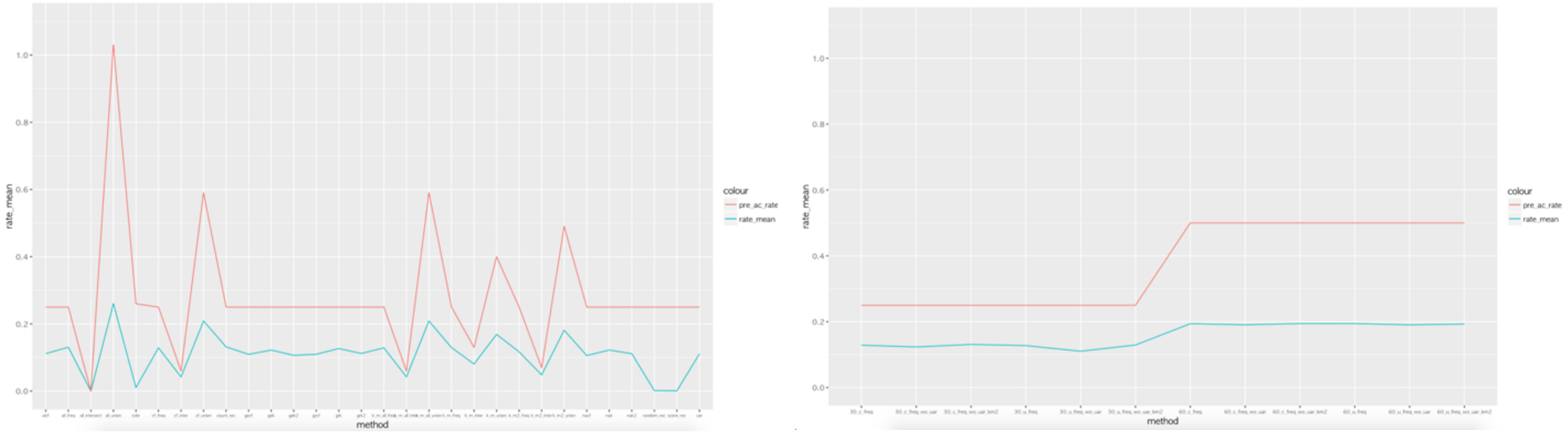
<분석 모델 결과 비교 그래프 (개선 후)>

Model Improvement

* 결과 비교 :

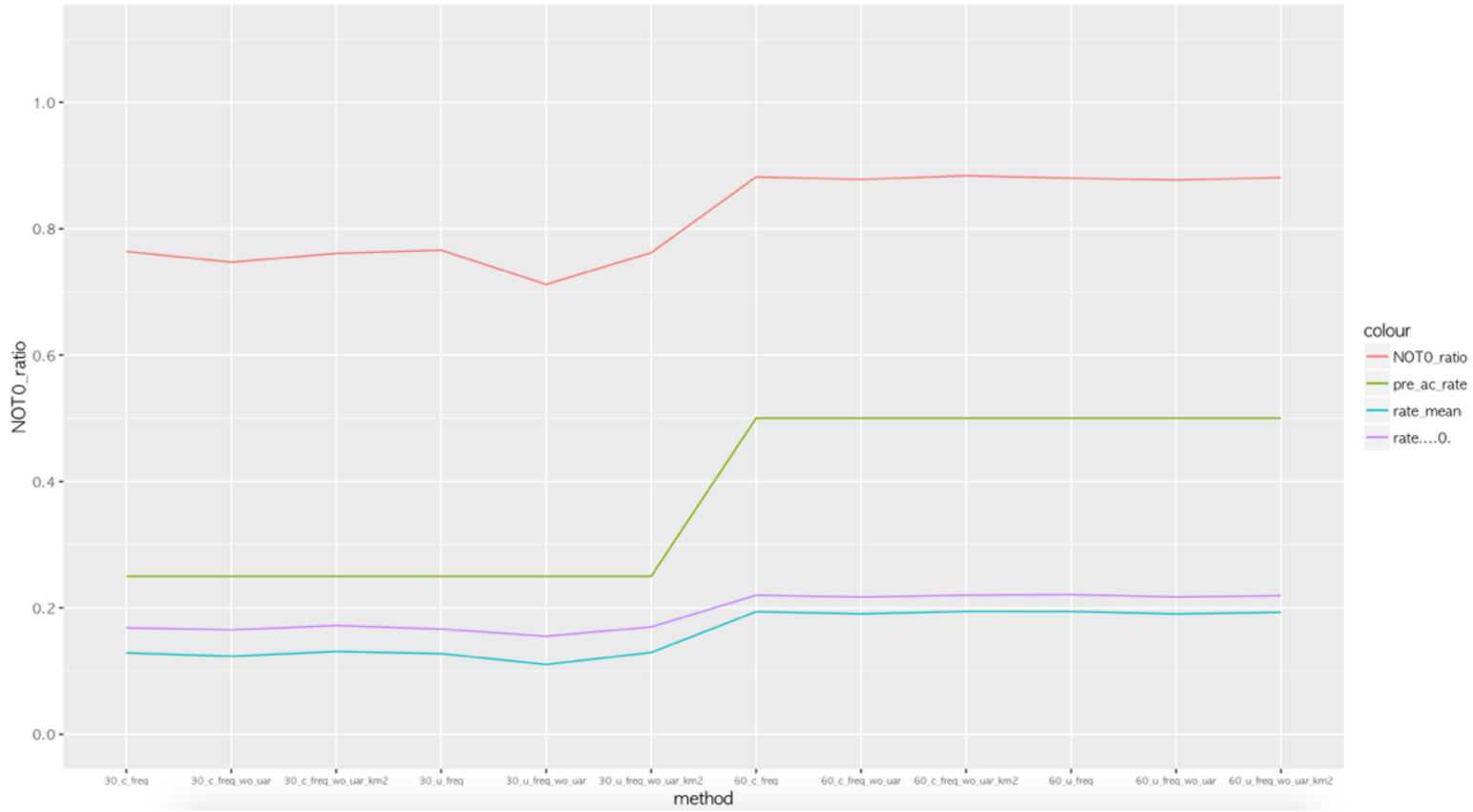
- 평균적인 적중율을 의미하는 rate_mean이 안정적으로 변했다. 30개를 추천하는(pre_ac_rate가 0.25) 경우 rate_mean이 11~13%가 되었고, 60개를 추천하는(pre_ac_rate가 0.5) 경우 rate_mean이 19%가 되었다.
- 개선전 모델에서 rate_mean이 개선 후 모델보다 높은 method는 pre_ac_rate 또한 0.6~1.0으로 rate_mean 증가분 대비 더 높았다.

=> 개선 후 모델의 경우 pre_ac_rate 대비 rate_mean을 증가시켰고, 결과적으로 더 안정적인 추천 모델이라고 판단할 수 있다.



<개선 전 & 개선 후 모델의 rate_mean과 pre_ac_rate 비교>

Model Improvement



<분석 모델 결과 비교 그래프 (개선 후)>

Conclusion

* 시사점 :

1) Naive recommendation model과의 비교

- 높은 평점의 영화, 랜덤 추천 등의 naive한 추천 모델과 비교했을 때 앙상블 모델의 정확도인 rate_mean이 평균적으로 0.1%에서 10%로 100배 이상 증가했으며, 개선된 앙상블 모델 중 60개 항목을 추천하는 모델의 경우 19%로 190배 가량 증가했다.
- review가 많은 영화를 기준으로 추천한 모델은 비교했을 때의 정확도인 rate_mean은 13%로 개선된 앙상블 모델 중 30개 항목을 추천하는 모델(11~12%)보다 높고 60개 항목을 추천하는 모델(19.4%)보다 낮았지만, 추천하는 영화가 주로 review가 많은 최신영화 위주로 편향되어 있었기 때문에 추천의 유의미함이 떨어진다.

-> 이러한 점을 고려했을 때, 개선된 앙상블 모델의 경우 한정된 정보와 제한 사항에도 불구하고 유의미한 performance를 보인다고 할 수 있다.

-> 높은 평점순, 랜덤, review순으로 추천하는 naive 추천 모델과 rate_mean이 높은 상위 7개 모델의 pre_ac_rate, NOT0_ratio, rate_mean을 종합적으로 비교했을 때, 개선된 앙상블 모델이 더 안정적인 추천 모델임을 알 수 있다.

* <상위 7 개 모델 결과 비교> 참고

2) Review_interval과 각 rate 지표의 관계

* Review_interval은 리뷰수를 각 구간별로 나눈 bin을 의미한다

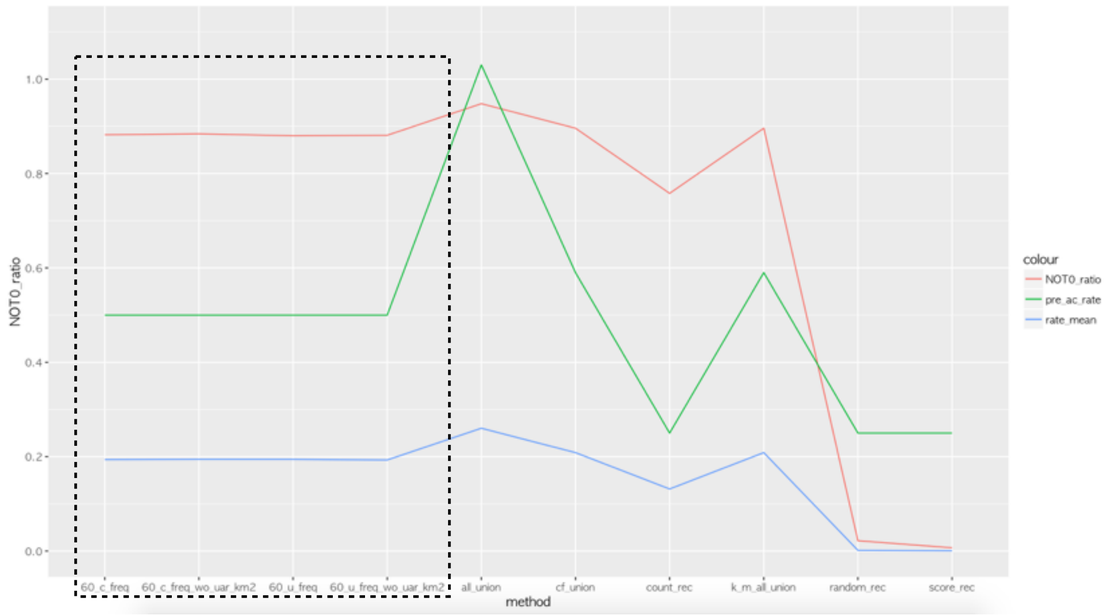
rate_mean이 15% 이상인 분석 모델에 대해

- Review_interval이 증가할수록, 대체로 rate_mean은 증가하는 경향을 보인다. 즉, accuracy를 높일 수 있다.
- Review_interval이 증가할수록, 0_rate는 낮아진다. 즉, accuracy를 높일 수 있다.

-> 데이터 확보의 중요성을 의미하며, 네이버 영화 페이지의 추천 서비스 개선이 필요한 이유를 뒷받침한다.

* <모델별 훈련 데이터양과 성과 비교> 참고

Conclusion



<상위 7개 모델 결과 비교>

method	interval&rate_cor	interval&0_rate_cor
k_m_union	0.5491324	-0.9173770
k_m2_union	0.7580306	-0.9040747
k_m_all_union	0.8064498	-0.8849938
cf_union	0.8064498	-0.8849938
all_union	0.7074732	-0.8516658
60_u_freq	0.5826639	-0.9046981
60_c_freq	0.6153044	-0.9062595
60_u_freq_wo_uar	0.5098509	-0.8957052
60_c_freq_wo_uar	0.5365103	-0.8943729
60_u_freq_wo_uar_km2	0.3328868	-0.9026882
60_c_freq_wo_uar_km2	0.2215479	-0.8978861

<모델별 훈련 데이터양과 성과 비교>

Conclusion

* 시사점 :

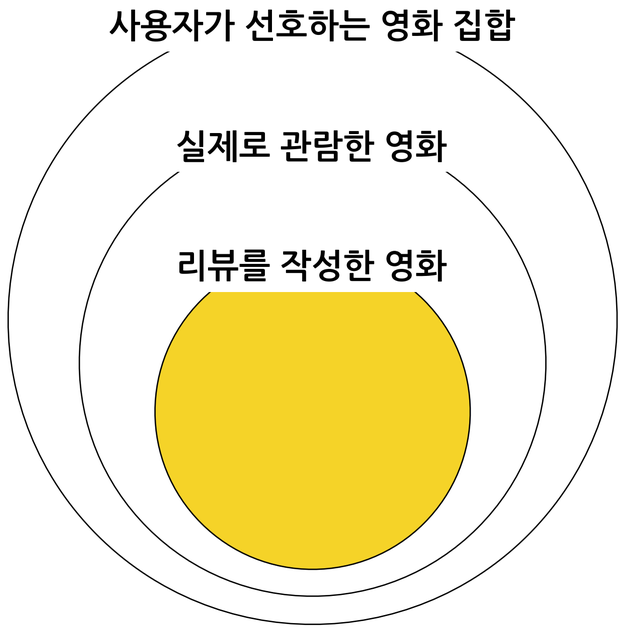
3) 성과 측정 모델의 불완전성과 가능성

A. 자체 설문조사 결과 사용자들은 일반적으로 추천 accuracy가 60% 이상이면 유의미하다고 느낀다.

B. 데이터 수집 단계에서 설계되지 않은 data만을 이용해야 했고 그렇기 때문에 주로 unsupervised learning을 활용한 knowledge discovery 작업이었다. 경쟁 서비스인 왓챠의 경우, 가입시 사용자 성향 분석을 위한 데이터 수집 단계가 존재하며, 사용자의 인구통계학적 정보도 함께 반영하는 반면, 단순한 영화 관람 기록만으로 보인 accuracy라는 점에서 서비스 구축의 가능성을 보인다고 할 수 있다.

C. 임의로 나눈 training set, test set으로 labeled data를 만들었지만, '사용자가 선호하는 영화의 추천'이라는 performance는 주관적인 수치이기 때문에 모델 평가에 있어서 가장 보수적인(안전한) 방법으로 성과를 측정했다. 현재 accuracy가 20%대이지만, 실사용에서 소비자 피드백은 더 높을 가능성이 크다는 것을 의미한다.

-> 초기 확보 가능한 데이터가 제한적이었음에도 불구하고, 위와 같은 사실을 고려하면 네이버가 서비스를 시작할 때 충분히 유의미한 performance를 보일 수 있다는 가능성을 보여준다.



Conclusion

* 시사점 :

4) 앙상블 모델의 설계와 모델 간의 성과 비교 및 유의성

- 분류율의 최대치는 모든 추천 항목을 더한 all_union 방법의 accuracy이다.
- uar과 ccbr은 크게 의미가 없었고, 전체적으로 accuracy를 높이는데 효과적인 영향을 끼쳤던 기법은 k=15일 때의 k-means와 collaborative filter를 사용한 기법이다.
- 한 번에 150개의 영화를 추천하는 것은 불필요하다. accuracy 감소율을 최소화하면서 추천 영화 수를 줄이기 위한 기법으로 freq는 유의미한 결과를 보였다. (150개 -> 60개, 예측율은 6% 감소)

-> 결과를 종합했을 때, 추천 항목을 30개로 제한했을 경우 k-means만을 사용하는 앙상블 모델이 18.1%로 가장 높았고, 추천 항목을 60개까지 확장했을 경우 collaborative filtering과 k-means의 앙상블 모델이 19.4%로 가장 높았다.

Conclusion

* 개선 사항 :

1) 날짜 정보 활용 :
- 회원이 리뷰를 작성한 날짜 정보를 알게 된다면, 1년을 주기로 각 장르별로 회원이 seasonality를 예측할 수 있을 것이다.
- 리뷰를 작성한 날짜와 영화를 관람한 날짜가 멀지 않다고 가정하고, 회원이 주로 영화를 관람하는 시점을 분석하고 예측할 수 있을 것이다.

2) 추천 포트폴리오 작성을 이용한 리스크 분배 :
- 추천 서비스에서 영화의 개봉 시점을 기준으로 '현재 상영 중인 영화'와 '상영하지 않는 영화', 혹은 단기간에 리뷰가 빠르게 증가한 영화군과 그렇지 않은 영화군 등으로 나누어 포트폴리오를 제작하여 리스크를 낮출 수 있다. 현재 상영 중인 영화는 영화의 선호보다 이슈성에 더 영향을 받으므로 회원 특성에 상대적으로 덜 영향을 받는 반면, 상영 중이지 않은 영화는 회원의 선호 성향에 더 크게 영향을 받는다.
  eg. 2017.4월 기준으로 예매율 1위는 '미녀와 야수'이다. 회원 성향이 확실하지 않은 경우, '미녀와 야수'를 추천하면 낮은 리스크로 적중률을 높일 수 있다.

3) 다른 수평적 서비스의 회원 정보 이용 :
- 회원의 네이버의 다른 서비스(웹툰, 음악)에서의 선호 성향을 포함하여 분석을 진행할 경우, 더 다채로운 추천 서비스와 분석이 가능할 것이다.