추알못의 슬기로운 추천생활 #1
prod. 추천 알고리즘 못 들어보신 분

글. 김희동 kt스카이라이프

언제부터인지는 잘 모르겠으나 드라마와 멀어진 지도 꽤 오래되었습니다. 아마도 어두 캄캄한 갱도의 끝자락, 빛이라고는 희미한 전등 하나에 의지한 채 목숨과 사투를 벌이며 행해지는 신성한 노동 현장의 의미가 왜곡되어 파생한 드라마가 난무하면서 부터인거 같습니다. 그렇다고 아예 안보는 것은 아닙니다. 작년 봄인 것 같습니다. 코로나가 특정 지역에서 점차 확산되고 있을 즈음, 오래된 친구 녀석과 만날지 말지 전화로 의견 조율을 하던 참에 요새 외출도 꺼려지는데 여가로 괜찮은 TV프로그램이 있다는 것이었습니다. 그래봤자 요즘 틀에 박힌 예능이나 식상한 드라마겠지 하는 생각에 무엇이냐고 물었더니 제목은 정확히 기억이 안 나고 예전 '응답하라' 시리즈를 만든 연출자와 작가가 다시 뭉쳐 제작한 비슷한 장르의 드라마라고 하더군요.

응x 시리즈는 학창시절 비슷한 또래가 겪어봤을 법한 내용이라 극중 전개가 지루하지만, 현실적이고 또 감동보다는 공감할 수 있는 부분이 와 닿았던 것 같습니다. 그래서 오래전 술자리서 말끝에 괜찮은 거 같다고 한 적이 있었는데 그걸 기억하고 이번 드라마를 추천해 주는 것이었습니다. 때마침 집콕으로 따분해하던 참에 그 말이 생각나 무심코 채널 튜닝을 하던 중 재방영되고 있는 '슬기로운 의사생활'을 시청하게 되었습니다. 소감이 궁금하시다고요? 음, 제 점수는요. 연출자와 작가 모두 저와 비슷한 연배라 그런지 몰라도 40대의 직장인, 여기서는 의사라는 전문 집단 소재를 따왔지만 결국에는 의사도 사람 사는 이야기고 인생사 생로병사 다 거기서 거기이기에, 결국 '슬기로운 의사생활' 자체도 '메디컬'이라 쓰고 '라이프'라 읽는 우리네 평범한 삶의 이야기를 아주 조금은 구성의 본능이 과장(드라마틱?)되었지

만 본성은 상선약수(上善若水)와 같이 자연스레 물 흐르듯 써 내려간 거 같아 많은 부분 공감할 수 있었습니다. 덕분에 20년 전, 즐겨듣던 쿨의 노래도 흥얼거릴 수 있어 좋았고 시청률 또한 괜찮아 올해에 시즌2가 나온다고 하더군요.
사실 드라마 자체를 얘기하고 싶었던 것이 아니라 글의 도입부로서 발단을 위해 필요했기에 여기까지만 하고 지금부터는 아까 그 친구가 제게 추천한 행위를 토대로 슬슬 오늘의 전개 부분으로 넘어가봐야 할 것 같습니다. 결론부터 말씀드리자면 그 친구는 제가 봤던 드라마(응답하라.)의 콘텐츠 정보(연출, 작가, 장르)를 기반으로 이와 유사한 드라마(슬기로운.)를 추천했기에 콘텐츠기반 필터링에 해당됩니다. 만약 연출과 작가는 같지만 새로운 장르를 시도한 다른 드라마를 추천하였다면 제가 재밌게 봤을까요? 결과를 단언할 수 없겠으나 본문에서 더 자세히 설명하겠지만 단순 콘텐츠 정보로는 한계가 분명해 보입니다. 그렇다면 다른 추천 방식이 필요할 거 같다는 생각이 듭니다.

구글이 아주 오래전 자사 검색 엔진의 시장지배력을 앞세워 Search를 대신하여 Googling을 만들었듯이, Netflixing 또한 Binge-watching처럼 어느샌가 장시간 동안 시리즈물을 시청하는 신조어가 돼버렸습니다. 이는 비단 넷플릭스의 막대한 오리지널 콘텐츠 투자 때문만은 아닐 거란 생각이 듭니다. 그도 그럴 것이 많은 포털에서 '넷플릭스'에 대한 연관 검색어를 살펴보면 어김없이 '추천'이란 단어가 이어집니다. 반대로 '추천'으로 검색해도 '넷플릭스'와 함께 등장하곤 합니다. 연관 검색어가 특정 단어 등을 검색창에 입력했을 때 이와 관련하여 나열되는 키워드이고 포털 측에서 그런 검색 패턴 등을 자동으로 분석하여 추출하기에 이미 많은 사람의 맘속엔 '추천'하면 '넷플릭스'가 각인되었다고 봐도 무방할 듯싶습니다. 주지하다시피 넷플릭스는 자신만의 추천시스템을 개발, 활용하여 기존가입자 만족 및 신규가입자를 지속적으로 유치하고 있습니다. 넷플릭스뿐만 아니라 유튜브, 아마존, 이베이 등 제품이든 서비스든 간에 고객의 소비를 절대적으로 필요 시 하는 모든 기업의 판매 전략에 있어 추천은 이미 선택이 아니라 필수로 자리 잡은 지 오래되었습니다.

따라서 추알못 시즌#1에서는 추천시스템의 종류에는 어떤 것들이 있고 어떻게 구분되는지에 관한 갈래부터 출발합니다. 그다음 추천의 고전이라 할 수 있는 베스트셀러나 설문방법을 통한 마케팅 1세대와 2세대 방법 중 연관관계분석(Association Rule Mining)의 개념과 이에 활용되는 Apriori, FPG(Frequent Pattern Growth) 알고리즘 등을 예제를 통해 알아봤습니다. 특히 실제 ARM이 콘텐츠 추천에 어떻게 활용될 수 있는지 사례를 바탕으로 살펴보고, 이론적 근거가 되는 함께 사용된 알고리즘들을 Python Code로 검증해 보았습니다.
시즌#2에서는 ARM과 큰 축을 이루는 정보필터링(Information Filtering) 방식이 이어집니다. 여기에는 앞서 잠깐 언급한 콘텐츠 기반 필터링(Content-based filtering), 협업 필터링(Collaborative Filtering), 그리고 이 둘을 합친 Hybrid Filtering 방식이 소개됩니다. 끝으로 협업 필터링을 메모리 기반(Memory based)의 아이템 기반(Item based)과 사용자 기반(User based)으로 나눠보고 3세대라 할 수 있는 머신러닝 기법의 모델 기반(Model based) 방식에는 어떤 것이 있는지 알아보겠습니다.
끝으로 여러 유사도 알고리즘 중에서 가장 기본이라 할 수 있는 거리기반의 Euclid Distance, 각도 기반의 Cosine Similarity, 그리고 이 유사도를 바탕으로 어떤 방식으로 분석이 가능하게 되는지 머신러닝의 비지도학습(Unsupervised Learning) 중 주성분 분석(Principal Component Analysis) PCA와 특이값 분해(Singular Value Decomposition) SVD 그리고 비음수행렬분해(Non-negative Matrix Factorization) NMF 등 시즌#1과 마찬가지로 다양한 사례와 함께 이를 Python으로 구현해 보았습니다. 이렇듯 추천시스템에는 다양한 방식이 존재하고 쓰이는 알고리즘 또한 무수히 많습니다. 따라서 추천 알고리즘을 못 들어보신 추알못 분들을 위하여 추천시스템은 무엇이고 어떤 방법으로 어떻게 사용하고 있는지를 이해하여 슬기로운 추천생활이 가능토록 기본 알고리즘부터 핵심 알고리즘까지 찬찬히 알아보겠습니다.

추천시스템이 뭔지를 제대로 알기 위해서는 시스템 알고리즘의 분석보다도 한 발짝 물러서서 우선 추천이란 단어의 뜻을 명확하게 집고 넘어갈 필요가 있습니다. 이는 머신러닝의 프레임웍에서 문제 정의를 먼저 확실히 파악하고 그에 맞는 정보 수집과 데이터 전처리 과정을 거쳐야 비로소 문제를 해결해 줄 수 있는 최적의 알고리즘 모델이 개발되는 것과도 같은 원리입니다. 혹시 퇴고(推敲)의 뜻이 뭔지 아시나요? 물론 작가가 본인의 시나 글 따위를 여러 번 고쳐 쓰는 것은 알겠는데 왜 하필이면 밀 퇴(推)와 두드릴 고(敲)가 쓰였을까요? 자세한 이야기는 시간 관계상 생략하기로 하고 시문의 쓰인 자구 중, 문을 밀다(推)와 두드리다(敲)를 두고 신중하게 고민한데서 유래합니다. 눈치가 빠르신 분은 아시겠지만 밀 퇴(推)는 밀 추(推)와 같은 자음(字音)이어서 추천(推薦) 또한 어떤 조건에 꼭 들어맞는 사람이나 물건을 들이밀어 권하는 것을 말합니다. 추천은 예로부터 동서양을 막론하고 책임지고 사람을 천거하거나 물건을 권할 때 쓰는 용어로 신중하게 행해지기에 단순히 알려주는 소개와는 확연히 다릅니다. 여기서부터가 문제를 정의하게 되는 기준이 됩니다. 따라서 추천을 하려면 대상(User)이 누군지에 대한 명확한 정보(Explicit Data)가 있어야 하고 또 무엇(Item)이 알맞은 것인 지에 대한 근거(Algorithm)도 타당해야 합니다.

몇 해 전까지만 해도 유명 스트리밍 콘텐츠 제공 회사의 추천 성공률이 75%를 넘어섰단 기사가 있었습니다. 단순 수치를 올리는 것이 중요한 걸까요? 물론 단기적으론 그럴 수 있지만 궁극적 목표는 완벽한 개인화이고 지금은 과도기에 있다고 보고 있습니다. 막연히 대중이 좋아하는 것이니까 먹히겠지 또는 인기 있는 아이템이니까 팔리겠지 하고 추천하였다간 점점 개인화가 뚜렷해지는 추세를 감안하면 낭패를 보기 십상입니다. 예전에야 고객이 누구인지에 대한 정확한 데이터도 부족하였고 또 있다고 하더라도 그로부터 Implication을 명시해 줄 수 있는 분산파일·처리 시스템이나 시각화 툴이 생소하였기에 알고리즘을 활용한 추천시스템의 도입은 요원하였는지도 모르겠습니다. 그래서 단순히 베스트셀러 또는 스테디셀러를 소개한다거나 대상이 누군지를 파악하기 위해 질의응답과 같은 설문이 실시되곤 하였습니다. 이런 방식이 고전이라 할 수 있는 마케팅 추천 1세대입니다. 2세대는 고객과 아이템을 구분하여 해당 빅데이터 정보를 처리하고 그 의미를 분석함에 있어 인공지능의 도움을 받는 정보필터링(IF)이나 연관성 분석(ARM) 등이 이에 해당합니다. 하지만 지금 기계학습이 주가 되어 하나의 Target이나 Outcome 변수가 존재하는 입력 데이터 독립변수와 출력 데이터 종속변수 사이의 관계함수를 규명해주는 지도학습(Supervised learning)의 분류(Classification)나 회귀(Regression) 알고리즘은 이미 신뢰할 수 있을 정도의 정확도를 증명해주고 있습니다. 이와 더불어 Target이나 Outcome 변수와 상관없이 입력 데이터 사이의 패턴으로부터 의미 있는 관계를 규명해 주거나 Training 데이터를 특징화(Featuring)하여 군집화(Clustering)나 차원축소(Dim. Reduction) 등에 활용되는 비지도 학습(Unsupervised learning) 또한 3세대 추천시스템의 주요 알고리즘으로 자리 잡았습니다. 아래 도표는 이런 추천시스템의 범주를 잘 도식화하고 있어 관련 사례연구에서 발췌해 보았습니다.

이제부터 본격적인 알고리즘에 대하여 알아보고자 합니다. 1세대는 [Figure 1]의 '기타'에 해당하는 부분으로 앞에서 대략 설명 드린 것이 전부이기에 다음으로 넘어가겠습니다. 2세대의 연관성 분석은 'What goes with what'을 규명하는 기법으로 고객이 어떤 상품(what)을 함께(with) 구매(goes)하

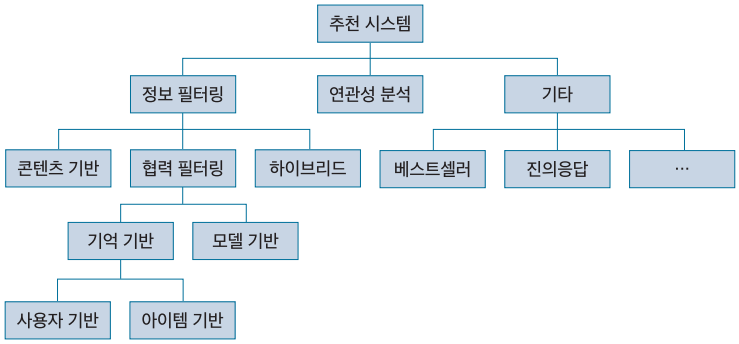

Figure 1. The Categorization of Recommender Systems, Son et al.,2015

는 지에 대하여 데이터 내부의 연관성, 즉 상품과 상품 간의 상호 관계 또는 종속 관계를 찾아내는 분석법을 뜻합니다. 대표적인 비지도학습 중 하나로 기계학습에서는 연관관계분석(Association Rule Mining, 이하 ARM)이라고도 부르며 지금도 개인화 추천 서비스에 널리 활용되고 있습니다. 이는 도메인에 따라 쓰이는 방식이 달라지는데 유통업에서는 흔히 장바구니 분석(Market Basket Analysis, 이하 MBA)으로 알려져 있습니다.

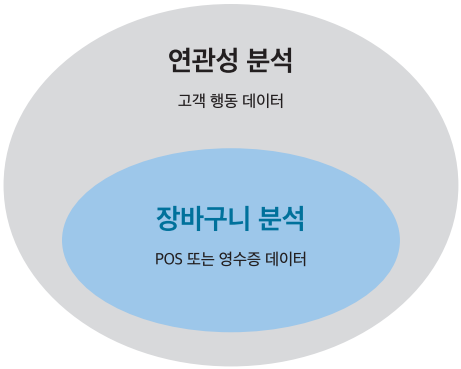

Figure 2. The Venn Diagram of ARM and MBA, Beusable

[Figure 2]는 ARM과 MBA의 관계를 잘 설명해주고 있는데, 이 MBA는 POS(Point-of-Sale) 시스템이 도입됨에 따라 전 세계적으로 확대되었습니다. 예를 들어 치킨을 산 고객이 맥주를 산다는 고객 영수증 데이터로부터 어떤 규칙이나 패턴을 발견하여 특정 상품을 구매한 고객에게 연관성이 높은 상품을 추천해주는 방식으로 사용됩니다. 상품 구매 예측 및 추천은 물론이거니와 매장에서 상품진열 방법, 사은품이나 패키지 구성 등 다양하게 활용되고 있습니다. 그 밖에도 여행, 호텔 숙박과 같은 서비스업에서는 고객들이 이용한 상품패키지를 분석하여 선호하는 서비스들이 어떤 방식으로 어떻게 이어지는지를 토대로 이를 새로운 고객에게 추천할 수도 있습니다. 이런 연관규칙에 대한 판단 기준은 전체 중 함께 발생하는 빈도수를 의미하는 지지도(Support)와 조건부 확률을 통해 항목 간 관련 정도를 측정하는 신뢰도(Confidence), 그리고 특정 상품이 함께 구매되는 확률 즉, 규칙과 상관없이 발생한 빈도수 가운데 함께 구매된 규칙이 얼마나 유효한지를 나타내주는 향상도(Lift) 값에 따라 상품 간의 연관성 의미를 분석해 낼 수 있습니다. [Figure 3]은 전체 중 두 상품 X와 Y가 있다고 가정했을 때, 이해를 돕기 위해 지지도, 신뢰도, 향상도를 벤다이어그램으로 나나낸 결과입니다.

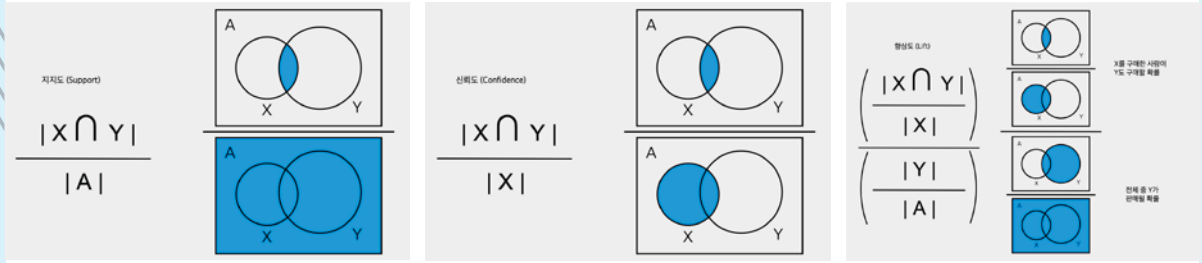

Figure 3. The Comparison among three Indications S., C., and L., Beusable

지지도는 직관적으로 전체 중에서 얼마나 두 상품 X와 Y가 함께 구매된 비율을 뜻하므로 별다른 설명이 필요 없을 것 같고 문제는 신뢰도와 향상도입니다. 신뢰도는 조건부 확률이라 말씀을 드렸는데 X가 구매되었을 때 X와 Y가 함께 구매된 비율을 뜻하기에 말 그대로 순서가 매우 중요합니다. 보통 Confidence(X→Y)와 같은 형태로 쓰이는데 이는 X의 구매가 선결 조건이 되므로 방향성이 중요합니다. 따라서 Confidence(Y→X)와는 전혀 다르다고 할 수 있습니다. 이제 중요한 것은 Lift인데 결론부터 말씀드리면 만약 이 향상도 값이 1보다 크면 X와 Y는 양의 상관관계를 갖는다는 뜻이 되고 이를 풀이하면 X를 사면 Y를 같이 구매할 확률이 그렇지 않은 경우보다 높다라는 결론을 내릴 수 있습니다. 왜냐하면 자세히 보시면 아시겠지만 향상도의 분자가 곧 신뢰도와 같음을 알 수 있고 분모는 X와의 구매 여부와 상관없이 전체 판매 중에서 Y가 차지하는 비율, 즉 Y의 인기 정도를 뜻하기에 그렇습니다. 다시 말해 Y 단독으로 팔리는 것(분모)보다 X가 구매되었을 때 Y가 함께 구매되는 신뢰도(분자) 값이 크다면 즉, 1보다 크다면 이 규칙이 유의미하다고 판단할 수 있기 때문입니다. 마찬가지로 이 값이 1보다 작다면 위 경우와 반대로 X와 Y는 음의 상관관계를 갖게 되어 함

께 구매되지 않을 확률이 그렇지 않은 경우보다 더 높다고 해석할 수 있게 되는 것입니다. 글로는 설명하기가 다소 난해할 수 있어 실제로 ARM이 콘텐츠 추천에 어떻게 활용될 수 있을지에 관한 예를 들어보겠습니다.

예 1) PPV 콘텐츠를 자사 네트워크로 제공하는 회사가 있습니다. 지난 한주 동안 영화를 3편 이상 구매한 상위 5명의 VIP들의 구매 행태를 토대로 아이템 간의 연관관계분석을 통해 추천시스템에 활용하여 매출을 증대시키려고 합니다. 유저-아이템 매트릭스는 아래와 같습니다.

User	PPV(Pay per view)
1	'담보', '검객', '국제수사'
2	'담보', '국제수사', '소리도 없이'
3	'검객', '국제수사', '소리도 없이'
4	'담보', '소리도 없이', '물란'
5	'검객', '국제수사', '물란'

Table 1. User-PPV Matrices

Rule 1) '담보'를 구매하면 '소리도 없이'를 구매

Rule 2) '국제수사'를 구매하면 '담보'를 구매

위와 같이 VIP 고객과 가장 많이 구매된 상위 5개의 콘텐츠로부터 어떤 유의미한 결론을 도출 시킬 수 있을까요? 먼저 Rule 1)을 살펴보겠습니다.('소리도 없이'는 이하 '소리'로 표기)

Individual Probability		Support	Confidence	Lift
P(담보)	P(소리)	P(담보∩소리)	P(담보∩소리)/P(담보)	P(담보∩소리)/P(담보)P(소리)
30/50	30/50	20/50	(20/50)/(30/50)	Confidence/P(소리)=0.67/0.6
0.6	0.6	0.4	0.67	1.11

Table 2. Rule 1's Support, Confidence, and Lift

먼저 '담보'와 '소리도 없이'의 개별 지지도 값을 구합니다. 각각 총 5번의 구매 리스트에서 3번씩 등장하였으므로 0.6이 되고 이중 동시에 구매된 경우는 2번이어서 공동 지지도 값은 0.4가 됩니다. 신뢰도는 '담보'를 시청한 고객이 '담보'와 '소리'를 함께 시청한 조건부 확률을 뜻하므로 위 계산식과 같이 0.67이 되고 이제 중요한 Implication을 얻을 수 있는 향상도 값만이 남았습니다. 앞에서 설명드린 바와 같이 '담보'와 '소리'가 함께 구매되는 규칙의 의미 즉, 전체 중에서 '소리'의 인기도 P(소리)와 비교하여 규칙이 얼마나 유효한지를 나타낸 값이 지지도이므로 계산식과 같이 P(담보∩소리)/P(담보)P(소리)가 되고 P(담보∩소리)/P(담보)값이 곧 신뢰도가 되므로 최종적으로 Confidence(담보→소리)/P(소리)와 같이 바꿔 쓸 수 있습니다. 이 값(1.11)이 1보다 크므로 우리는 아래와 같은 유의미한 결론을 내릴 수 있습니다.

"'담보'와 '소리'가 양의 상관관계를 갖는 것을 뜻한다.
즉, '담보'를 시청하면 '소리'를 시청할 확률이 시청하지 않을 확률보다 더 크다고 할 수 있다."

다음 Rule 2)도 위와 같이 풀이해 보겠습니다. ('국제수사'는 이하 '국제'로 표기)

Individual Probability		Support	Confidence	Lift
P(국제)	P(담보)	P(국제∩담보)	P(국제∩담보)/P(국제)	P(국제∩담보)/P(국제)P(담보)
40/50	30/50	20/50	(20/50)/(40/50)	Confidence/P(담보)=0.5/0.6
0.8	0.6	0.4	0.5	0.83

Table 3. Rule 2's Support, Confidence, and Lift

풀이 방법은 Rule 1)과 같은 방식으로 먼저 '국제'와 '담보'의 개별 지지도 값을 구합니다. 이 값들은 [Table 3]에서 보는 바와 같이 각각 0.8과 0.6이 되고 신뢰도는 '국제'를 시청한 고객이 '국제'와 '담보'를 함께 시청한 조건부 확률을 뜻하므로 위 계산식과 같이 0.5가 됩니다. 핵심인 향상도 값은 P(국제∩담보)/P(국제)P(담보)가 되고 P(국제∩담보)/P(국제)값이 곧 신뢰도가 되므로 최종적으로 Confidence(국제→담보)/P(담보)와 같이 바꿔 쓸 수 있습니다. 이 값(0.83)이 1보다 작으므로 이 역시 아래와 같은 유의미한 결론을 내릴 수 있습니다.

"'국제'와 '담보'가 음의 상관관계를 갖는 것을 뜻한다.
즉, '국제'를 시청하면 '담보'를 시청하지 않을 확률이 시청할 확률보다 더 크다고 할 수 있다."

이 두 가지 말고도 다양한 분석들이 있을 수 있겠지만 일단 아이템 간의 연관성 분석 자체만을 두고 추천시스템에 활용한다면 '담보'를 시청한 고객에게 '소리도 없이'를 추천해 주는 것이 더 효과적이라는 것을 알 수 있습니다. 이는 Machine learning에서 Classification Performance Evaluation에 이용하는 'Confusion Matrix'의 'Sensitivity(true positive or recall)' 값이 증가하게 되어 성능향상이 되는 것과 같은 원리입니다. 또한 '국제수사'를 시청한 고객에게 '담보'를 추천해 주지 않는 것이 위와 마찬가지로 'Confusion Matrix'의 'Specificity(true negative)' 값을 증가시킬 수 있습니다. 만약 '담보'를 시청한 고객에게 '소리도 없이'를 추전해 주지 않고 '국제수사'를 시청한 고객에게 '담보'를 추천해 준다면 각각 'Type Ⅰ error(false negative)'와 'Type Ⅱ error(false positive)' 값이 증가하게 되어 성능 저하를 초래하게 됩니다. 여기서 성능향상의 뜻은 매출 증대에 기여하게 됨을 뜻하고 성능 저하는 그렇지 않다는 뜻이 됩니다. 실제로 '담보'와 '소리도 없이'를 봤는데 어린 여자아이를 자의든 타의든 담보로 맡으면서 일어나는 상황을 하나는 웃음과 감동으로 다른 하나는 해학과 연민으로 풀어가는 영화입니다. 뻔하지만 웃음 속 감동을 주는 신파를 좋아하시는 분은 '담보'를, 기존의 분류된 장르를 넘어 새로운 블랙코미디를 즐겨 찾는 분은 '소리도 없이'를 추천합니다.

추천 얘기를 하다 보니 저도 모르게 영화추천을 하고 말았는데 그럼 지금부터는 위 계산이 맞는지 Python으로 구현해 보겠습니다. 아직 전 세계 개발자들이 가장 많이 사용하는 언어는 오랜 역사와 전통을 지닌 C와 Java이지만 인공지능, 특히 딥러닝에 특화된 언어는Python이 주류인 것만은 확실합니다. 대표적인 대화형 언어로 사용 가능한 플랫폼이 다양하고 Interpreter 형식이기 때문에 유저가 별도의 Compile 과정이 없이도 코드를 바로 실행하여 확인할 수 있고 또 기본으로 제공되는 Library가 많기에 초보자부터 전문가까지 두루 사용되고 있습니다. 처음에는 Python의 idle 창에서 간단한 코드를 입력하고 출력을 확인하는 방식으로 사용합니다. 그리고 다소 긴 코드들은 Script mode에서 코드를 작성하고 idle에서 출력을 확인하는 방법으로 사용합니다만 이런 방식은 개방을 중요시하는 코드환경과는 거리가 있어 보입니다. 만약 개발자들이 같은 UI로 코드를 확인하고 실행할 수 있다면? 또 코드를 수정하거나 그 결과를 실시간으로 볼 수 있다면? 이를 위해 Jupyter Notebook이 탄생하였습니다. Jupyter Notebook은 오픈 소스 소프트웨어, 개방형 표준, 그리고 여러 개의 프로그래밍 언어에 걸쳐 인터랙티브 컴퓨팅을 위한 서비스 개발을 위해 설립된 비영리 단체인 Project Jupyter에 의해 개발되었습니다. Jupyter가 목성을 뜻하는 Jupiter와 발음이 같은데 이는 Jupyter가 지원하는 세 개의 핵심 언어인 Julia, Python 그리고 R에서 유래했으며, Notebook은 목성의 위성의 발견이 기록된 갈릴레오 갈릴레이의 공책에 대한 존경의 의미도 내포하고 있다고 합니다. 따라서 Jupyter Notebook은 웹 브라우저를 통해 실행되므로 Jupyter Notebook 자체를 자신의 로컬 시스템이나 원격 서버에 호스팅 할 수도 있습니다. 그밖에 데이터 시각화나 대화형 코드 공유가 가능하여 많은 개발자들이 사용하고 있습니다.
그럼 Python에 관한 간략한 설명은 이것으로 마치고 다시 본문으로 돌아와 위 PPV 연관성 분석은 예제용이라 유저와 아이템을 현저히 줄여 놓았습니다. 실제로 분석을 위해서는 대용량의 유저-아이템 매트릭스 정보를 다양한 텍스

트 파일형태 등으로 불러들여 사용하는데 여기서는 앞의 분석에서 수기로 계산한 지지도, 신뢰도, 향상도 값의 결과가 맞는지 Jupyter Notebook 환경에서 Python 코드로 증명해 보겠습니다. 아래 [Code 1]에서는 향상도 값의 min_threshold 값이 0.5 이상인 PPV들을 추려보았습니다. 이 조건은 어떤 분석을 하느냐에 따라 달라질 수 있으며 만약 1 이상인 것들만 추천하고자 한다면 min_threshold 값을 1로 입력하면 됩니다. Rule 1)과 2)의 경우 각각 아래 코드 결과값의 7번과 2번에 해당하여 지지도, 신뢰도, 상승도 값이 수기로 풀었을 때와 같음을 확인할 수 있습니다. 그 밖에도 [Code 1]의 결과를 바탕으로 지지도값이 1이 넘는 0번 '국제'를 구매한 고객에게 '검객'을, 1번 '검객'을 구매한 고객에게도 '국제'를, 6번 '담보'를 구매한 고객에게 '소리'를 각각 추천해 주는 것이 기업 입장에서 매출 증대에 효과적이란 결론을 도출할 수 있습니다. 이처럼 아이템별로 일일이 계산할 필요가 없고 분석에 필요한 조건에 맞춰 출력을 해주니 Python이 얼마나 유의미한 결과를 추론하고자 할 때 유용한지 확인할 수 있었습니다.

In [1]:
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori,association_rules
dataset = [('담보','검객','국제'),
           ('담보','국제','소리'),
           ('검객','국제','소리'),
           ('담보','소리','물란'),
           ('검객','국제','물란')]
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.3, use_colnames=True)
from mlxtend.frequent_patterns import association_rules
association_rules(frequent_itemsets, metric="lift", min_threshold=0.5)
```

Out[1]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(국제)	(검객)	0.8	0.6	0.6	0.750000	1.250000	0.12	1.6
1	(검객)	(국제)	0.6	0.8	0.6	1.000000	1.250000	0.12	inf
2	(국제)	(담보)	0.8	0.6	0.4	0.500000	0.833333	-0.08	0.8
3	(담보)	(국제)	0.6	0.8	0.4	0.666667	0.833333	-0.08	0.6
4	(국제)	(소리)	0.8	0.6	0.4	0.500000	0.833333	-0.08	0.8
5	(소리)	(국제)	0.6	0.8	0.4	0.666667	0.833333	-0.08	0.6
6	(담보)	(소리)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
7	(소리)	(담보)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2

Code 1. The Analysis of three Indications-Support, Confidence and Lift

이런 ARM은 데이터베이스에 빈번하게 등장하는 세트 중, 신뢰도를 기준으로 연관이 있는 아이템 간의 규칙이나 패턴을 찾아내는 방법이므로 사용자나 아이템 수가 증가하면 할수록 많은 문제점이 동반됩니다. 첫째로 계산량이 기하급수적으로 늘어나게 됩니다. 만약 아이템이 n개라면 검토해야 할 경우의 수는 n*(n-1)로 증가하게 됩니다. 둘째로 데이터 희소성(Sparsity) 문제입니다. 데이터 희소성은 유저-아이템 매트릭스에서 0의 값을 갖는 희소행렬로 이를 어떻게 처리하는지에 따라 협업 필터링의 성능에 큰 영향을 미치게 됩니다. 두 번째 문제를 해결하기 위해서는 시즌#2에서 자세히 설명하겠지만 차원축소에 활용되는 알고리즘인 주성분 분석 PCA, 특이값 분해 SVD, 비음수행렬분해 NMF 등이 있습니다. 그렇다면 첫 번째 문제를 해결하기 위해 다른 항목에 비해 보다 더 자주 등장하는 아이템에 초점을 맞춰 '효과적'으로 연관규칙을 찾아주는 Apriori algorithm이 제안되었습니다. 여기서 '효과적'이란 단어의 의미는 만약 임의의 아이템 집합의 지지도가 일정 기준(ex: Minimum Support, Minsup)을 넘지 못한다면 이 아이템 집합을 포함하는 초월집합(Superset)의 지지도 또한 해당 기준보다 작아 이런 아이템 집합의 규칙은 배제하게 되어 앞으로 계산할 필요가 없어지기에 그렇습니다. 이 역시 글로는 이해하기가 어려울 수 있어 아주 쉬운 예를 들어보겠습니다.

예 2) VOD 콘텐츠를 스트리밍으로 제공하는 회사가 개인별 유사도를 측정하여 분류된 집합에 가장 비슷한 신규가입자를 맞이하였습니다. 그런데 이 신규가입자의 정보가 빈약하여 이를 테스트하기 위해 군집화된 기존가입자들의 최근 구매한 영화 패턴 행태를 분석하여 가장 연관성이 높은 콘텐츠를 차례대로 추천하고자 합니다. 이 클러스터링에는 총 10명이 있고 지난 일주일간 구매된 콘텐츠는 '오!문희', '오케이마담', '죽지 않는 인간들의 밤', '디바', '정무문', '다만 악에서 구하소서' 총 6편이었습니다. 이를 바탕으로 Apriori를 실행해 보겠습니다.

1) Minsup을 정함.(전체 Database의 지지도를 바탕으로 최소 3번 구매로 기준 정함)

Minsup=3

User	VOD
1	'오!문희', '오케이', '죽지 않는'
2	'오케이', '디바'
3	'오케이', '정무문'
4	'오!문희', '오케이', '디바'
5	'오!문희', '정무문'
6	'오케이', '정무문'
7	'오케이', '디바'
8	'오!문희', '오케이', '정무문', '죽지 않는'
9	'오!문희', '오케이', '정무문'
10	'다만악에서'

Table 4. User-VOD Matrices

k=1

VOD	Support
'오문희'	5
'오케이'	8
'죽지않는'	2
'디바'	3
'정무문'	5
'다만악에서'	1

: 붉게 표시된 VOD가 Minsup을 만족

2) 전체 지지도를 고려하여 자주 등장하는 VOD '오문희', '오케이', '디바', '정무문'이 Minsup 기준인 3보다 높음. 따라서 Minsup을 만족하는 두 개의 VOD 셋으로 확장

k=1

VOD	Support
'오문희'	5
'오케이'	8
'죽지않는'	2
'디바'	3
'정무문'	5
'다만악에서'	1

k=2

VOD	Support
'오!문희'& '오케이'	4
'오문희'& '디바'	1
'오!문희'& '정무문'	3
'오케이'& '디바'	3
'오케이'& '정무문'	4
'디바'& '정무문'	0

3) VOD 셋 '오문희'&'오케이', '오문희'&'정무문', '오케이'&'디바', '오케이'&'정무문'이 Minsup 기준인 3보다 높음. 따라서 Minsup을 만족하는 세 개의 아이템 셋으로 확장

k=2

VOD	Support
'오!문희'& '오케이'	4
'오문희'& '디바'	1
'오!문희'& '정무문'	3
'오케이'& '디바'	3
'오케이'& '정무문'	4
'디바'& '정무문'	0

k=3

VOD	Support
'오!문희', '오케이', '정무문'	2
~~'오!문희', '오케이', '죽지않는'~~	2

: '죽지않는'의 개별 항목 지지도 2<3

4) 3개의 VOD 셋으로 확장하였으나 Minsup 3의 기준을 넘지 못하므로 여기서 중단. ~~'오문희', '오케이', '죽지 않는'~~은 3개의 VOD 셋 중 2라는 가장 높은값을 받았지만 이미 개별항목 '죽지 않는'의 지지도가 Minsup 3

의 기준을 충족하지 못하므로 제외. 아래 [Figure 4]에서 확인할 수 있듯 개별 항목의 서포트를 계산하여 특정 minsup 기준에 미달하는 a, b 그리고 이를 포함하는 Superset(ex: ab, ac, abc 등)은 배제(Pruned)하고 자주 등장하는 c, d, e의 슈퍼셋(ex: cd, ce, cde)을 조사하여 minsup 기준을 만족 하는 Superset이 나오지 않을 때까지 k값을 늘여 가며 Apriori 실행.

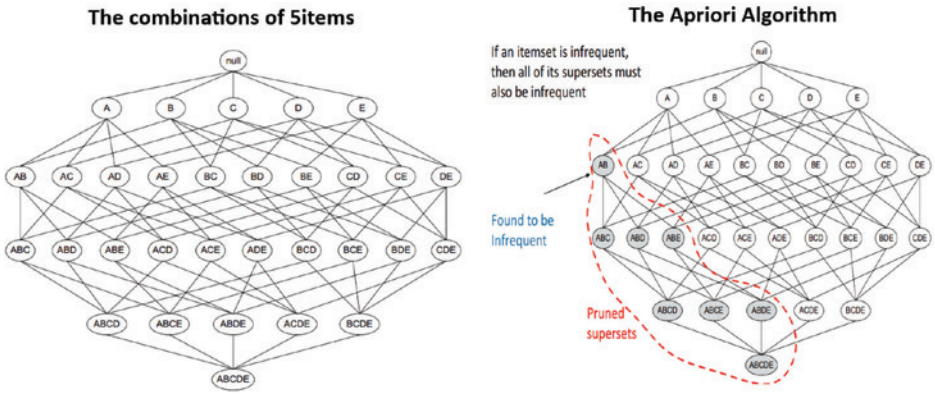

Figure 4. The Principle of Apriori Algorithm, WIKI

```
In [2]: import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
vod = [('오문희', '오케이', '죽지않는'),
       ('오케이', '디바'),
       ('오케이', '정무문'),
       ('오문희', '오케이', '디바'),
       ('오문희', '정무문'),
       ('오케이', '정무문'),
       ('오케이', '디바'),
       ('오문희', '오케이', '정무문', '죽지않는'),
       ('오문희', '오케이', '정무문'),
       ('다만악')]
te = TransactionEncoder()
te_ary = te.fit(vod).transform(vod)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_vodsets = apriori(df, min_support=0.3, use_colnames=True)
frequent_vodsets
```

Out[2]:

	support	itemsets
0	0.3	(디바)
1	0.5	(오문희)
2	0.8	(오케이)
3	0.5	(정무문)
4	0.3	(디바, 오케이)
5	0.4	(오문희, 오케이)
6	0.3	(오문희, 정무문)
7	0.4	(정무문, 오케이)

Code 2. The Implementation of Apriori Algorithm

수기로 계산한 Apriori Algorithm이 맞는지 마찬가지로 Python Code로 확인해 봤습니다. Minsup 0.3을 기준으로 잡았기에 위 결과와 같이 4번부터 7번의 값은 k=2일 때의 값과 같음을 알 수 있습니다. 최소 지지도인 Minsup의 기준을 충족하는 빈발항목집합을 찾은 후 그것들에 대해서만 연관규칙 계산하는 Apriori는 모든 가능한 아이템의 부분집합의 개수를 줄이는 1세대 방식인데 반해 2세대 방식인 Frequent Pattern Growth(이하 FPG)는 거래 내역 안에 포함된 품목의 개수를 줄임으로써 비교하는 횟수를 줄이는 방식으로 작동하기에 보다 더 효율적이라 할 수 있습니다. 다시 말해, FPG는 Apriori와 달리 후보 빈발항목 집합을 생성하지 않고 FP-Tree를 만든 후 분할정복(Divide-and-conquer)방식을 사용하기에 상대적으로 효율적인 연관분석이 가능하여 Commercial 한 응용 프로그램에 더 적합하다 할 수 있습니다. 다음 장의 [Figure 5]는 1세대와 2세대 연관분석 알고리즘이 Dataset size가 증가할수록 처리하는 속도가 어떻게 달라지는 지를 보여주는 성능비교 그래프입니다.

· 1세대 : Apriori(빈발항목 집합으로부터 연관 규칙이나 패턴 찾음)
· 2세대 : FP-Growth(Apriori 단점 보완, 분할정복 방식을 통해 빠르게)
· 3세대 : FPV(분산의 관점에서 2세대의 단점 수정 보완, 메모리 효율적 사용, 비교 횟수 단축)

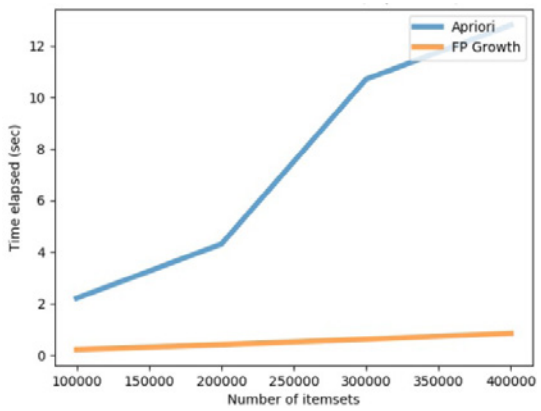

Figure 5. Performance Comparison between Apriori and FPG, WIKI

FPG는 그래프에서 보는 바와 같이 Itemset의 증가가 처리 속도에 별다른 영향을 끼치지 못하는데 반해 Apriori는 Itemset이 늘어남에 따라 우상향하고 있음을 확인할 수가 있습니다. FPG Algorithm에 대해 앞서 보여 드린 VOD 예로 간략히 설명하자면 [Table 5]와 같이 우선 개별 VOD의 지지도를 구하고, VOD Itemset을 정렬할 때 지지도 순서로 나열하여 이 값이 같을 때는 알파벳 순서로 우선순위를 정해줍니다. 그다음 [Figure 6]과 같이 FPG의 핵심이라고도 할 수 있는 분할정복(Divide-and-conquer)방식을 활용하여 FP-tree를 건설하게 됩니다. 분할정복은 해결하고자 하는 문제를 동일한 유형의 하위 문제로 분할하고 가장 작은 단위의 하위 문제를 해결하여 정복하는 과정을 말합니다. 그리고 마지막으로 하위 문제에 대한 결과를 원래 문제에 대한 결과로 조합하는 개념으로 여러 알고리즘의 기본이 되는 해결방법입니다. FP-conditional tree로부터 Conditional Pattern Base와 Conditional FP tree, 그리고 Frequent VOD Sets를 구하는 방법은 아래 [Table 6]과 같습니다. 편의상 '오!문희'부터 '다만 악에서 구하소서'까지를 알파벳 A~F로 대체하였습니다. Minsup 3을 기준으로 하였기에 '죽지 않는 인간들의 밤' C(지지도 2)와 '다만 악에서 구하소서' F(지지도 1)는 제외되었습니다.

User	VOD sets (Original)
1	A, B, C
2	B, D
3	B, E
4	A, B, D
5	A, E
6	B, E
7	B, D
8	A, B, E, C
9	A, B, E
10	F

User	VOD sets (Threshold=3)
1	B, A, ∈(Below Threshold)
2	B, D
3	B, E
4	B, A, D
5	A, E
6	B, E
7	B, D
8	B, A, E
9	B, A, E
10	F(below Threshold)

Table 5. Cleaning and Sorting according to Minsup and Alphabetical Order

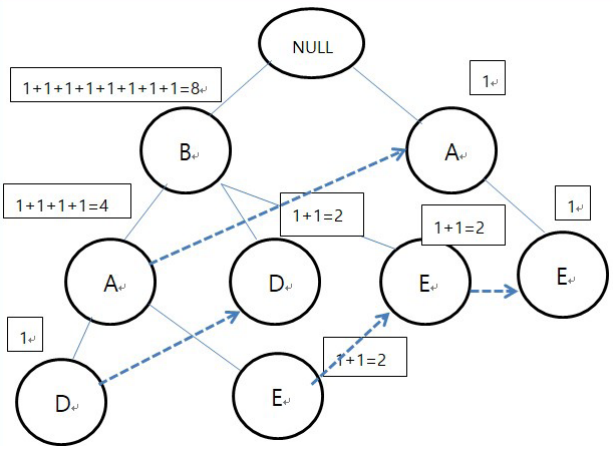

Figure 6. Construction of FP-tree, Author

알파벳 B는 '오!문희'를 나타내고 있고 그 위 숫자 8의 의미는 한번 등장할 때마다 Counting되어 총 8번, 그러니까 지지도 값과 같습니다. 조합은 맨 아래에 있는 D부터 시작하게 되는데 Conditional Pattern Base의 {B,A:1}는 B-A-D가 1번 발생했다는 뜻이고 {B:2}는 B-D가 2번 발생했다는 뜻입니다. Conditional FP Tree의 {B:3}은 Conditional Pattern Base에서의 B의 값을 더한 값입니다. {B,A:1}에서 한 번, {B:2}에서 두 번 나왔기에 최종값은 3이 됩니다. 그럼 최종적으로 Frequent VOD Sets를 도출하기 위해서 D와 조합을 하면 되는데,

Minsup을 3으로 정했기에 나올 수 있는 조합은 {B,D}:3이 됩니다. 같은 방법으로 E와 A를 계산하게 되는데 결과는 아래 [Table 6]과 같고 위에서 구한 Apriori와 같음을 확인할 수 있습니다.

	Conditional Pattern Base	Conditional FP Tree	Frequent VOD Sets
D	{B,A:1}, {B:2}	{B:3}	{B,D}:3 ←B&D's Support is 3
E	{B,A:2}, {B:2}, {A:1}	{B:4}, {A:3}	{B,E}:4, {A,E}:3, {B,A,E}:2
A	{B:4}	{B:4}	{B,A}4
B	-	-	

Table 6. FP-conditional tree

실제로 FP-tree를 그리고 FP-conditional tree로부터 Frequent Item-set을 구하는 방법은 해보시면 아시겠지만 직접 풀기에는 어려움이 따를 수 있습니다. 그럼 검증을 해보겠습니다. [Code 3]은 코드가 다소 길어 결과 부분만 보여드렸습니다. 확실히 Apriori보다 길지만 Dataset을 보다 효율적으로 분석하는데 용이한 방법입니다. Apriori 결과와 같이 {B,E}:4, {A,B}4, {A,E}:3, {B,D}:3이 Minsup 조건을 만족하였고 3개로 확장한 Dataset은 모두 제외되었습니다.

```
            fqset[1]=n['wordcc']
            result.append(fqset)
            condtran = self.condtreetran(n['linknode'])
            contree= fptree(condtran,sup)
            conwords = contree.findfqt(fqset)
            if conwords is not None:
                for words in conwords:
                    result.append(words)
        return result
item_sets = [('ABC'),
             ('BD'),
             ('BE'),
             ('ABD'),
             ('AE'),
             ('BE'),
             ('BD'),
             ('ABEC'),
             ('ABE'),
             ('F')]
fp_tree = fptree(item_sets, 2)
frequentwordset = fp_tree.findfqt()
frequentwordset=sorted(frequentwordset,key = lambda k: -k[1])
frequentwordset
```

```
Out[3]: [[{'B'}, 8],
         [{'E'}, 5],
         [{'A'}, 5],
         [{'B', 'E'}, 4],
         [{'A', 'B'}, 4],
         [{'D'}, 3],
         [{'B', 'D'}, 3],
         [{'A', 'E'}, 3],
         [{'C'}, 2],
         [{'B', 'C'}, 2],
         [{'A', 'B', 'C'}, 2],
         [{'A', 'C'}, 2],
         [{'A', 'B', 'E'}, 2]]
```

Code 3. The Implementation of FPG Algorithm

이것으로 시즌#1에서는 추천시스템의 고전인 1세대부터 2세대 연관관계분석(Association Rule Mining)의 개념과 다양한 사례를 통해 여러 알고리즘에 대하여 알아보았습니다. 어떤가요? 이제 슬슬 감이 오시나요? 이 기세를 몰아 다음 시즌#2에서는 본격적으로 2세대 정보필터링(Information Filtering) 방식에 대해 살펴보면서 함께 사용된 Euclid Distance, Cosine Similarity, PCA, SVD, NMF 등의 알고리즘과 Python Code에 대해 설명드릴 예정입니다. 특히 협업필터링에서 추천시스템 알고리즘의 핵심이라 할 수 있는 행렬분해(Matrix Factorization)의 의미와 그 쓰임새에 대하여 실제 활용사례를 시즌#1과 마찬가지 방법으로 소개해 드릴 예정이오니 많은 관심 부탁드립니다. 그럼 『추알못의 슬기로운 추천생활』#2에서 다시 만나요.