2019 January 9

조선대학교 IT융합대학 컴퓨터공학과

20144800 천승현

**2019동계 계절학기 OSSP   
중간 보고서**

영화 추천 시스템 구현

내용

[**1. 주제** 2](#_Toc534828276)

[**2. 필요성 및 기대효과** 2](#_Toc534828277)

[**3. 기존 연구** 3](#_Toc534828278)

[**4. 제안방법** 6](#_Toc534828280)

[- 문제 파악 6](#_Toc534828281)

[- 데이터베이스 다운로드 7](#_Toc534828282)

[- 필요 데이터 선별 9](#_Toc534828283)

[- 예측 모델에 맞게 데이터 변형 9](#_Toc534828284)

[**5. 실험 및 결과** 9](#_Toc534828285)

[- 실험 데이터에 대한 설명(자료형태, 구조) 9](#_Toc534828286)

[- 개발환경(H/W 스펙, SW 버전) 10](#_Toc534828287)

[- 모델 개선 10](#_Toc534828288)

[- 모델 테스트 10](#_Toc534828289)

[**6. 느낀점** 10](#_Toc534828290)

[**7. 참고** 10](#_Toc534828291)

# **1. 주제**

* **영화의 정보들을 모아둔 데이터를 사용하여 사용자에게 영화를 추천해주는 프로그램 구현**

# **2. 필요성 및 기대효과**

온라인 컨텐츠 시장이 성장함에 따라 소비자들은 더욱 많은 선택권을 가지게 되었다. 하지만 그만큼 모든 컨텐츠를 소모하기엔 시간적 여유가 따르지 않기 때문에 소비자들은 제한된 시간을 효율적으로 소모하기 위해 제한적으로 컨텐츠를 고를 수 밖에 없다. 대부분의 멀티미디어 서비스 웹사이트들은 이런 소비심리를 전략적으로 이용하기 위해 대부분 컨텐츠 추천 알고리즘을 사용하고 있다.

최근 가장 높은 성장률을 보이고 있는 넷플릭스(Netflix)는 대표적인 멀티미디어 컨텐츠 서비스 사이트이다. 넷플릭스는 강력한 컨텐츠 추천알고리즘을 사용하여 사용자(User)가 시청한 영상물들을 분석하여 트렌드를 파악하고, 취향에 맞는 컨텐츠를 추천해준다. 또는 다른 유저들이 많이 즐긴 컨텐츠들을 리스트로 상단에 노출시키는 마케팅을 하기도 한다.

* **Netflix : 대여되는 영화의 2/3가 추천으로부터 발생**
* **Google News : 38% 이상의 조회가 추천에 의해 발생**
* **Amazon : 판매의 35% 가 추천으로부터 발생**
* **Netflix Prize (~2009) Netflix에서 주관하는 경연대회로, 영화 선호도를 가장 잘 예측하는 협업 필터링 알고리즘에 수상 (US$1,000,000)**

이런 컨텐츠 추천 알고리즘은 소비산업에서 가장 필수적이면서 기본적인 마케팅 기법이 되었다. 미디어 산업뿐만 아니라 쇼핑, 스포츠, 부동산등 여러 분야에서 활발히 사용되고 있고 소비산업 뿐만 아니라 사용자들의 니즈를 분석하여 새로운 데이터를 수집할 수 있다는 점에서 다양하게 사용될 수 있다. 앞으로 이런 추천 알고리즘을 개량 시키고 발전시킴으로써 궁극적으로 소비자의 마음을 읽어 완벽하게 필요로 하는 요소들만 추천하는 인공지능 분야에도 사용될 수 있다.

# **3. 기존 연구**

# ìí ì¶ì² ìê³ ë¦¬ì¦ì ëí ì´ë¯¸ì§ ê²ìê²°ê³¼

현재 가장 활발히 사용되고 있는 추천 알고리즘은 협업 필터링(Collaborative filtering)이다. 협업 필터링은 사용자들로부터 정보를 받아 사용자들의 관심사들을 자동적으로 예측하여 추천해주는 필터링 기법이다. 협업 필터링은 크게 유저 기반 협업 필터링(User-based-CF), 아이템 기반 필터링(Item-based-CF)로 나누어지고, 유저 기반 협업 필터링은 다시 능동적, 수동적 필터링으로 분류 된다. 이런 필터링 알고리즘들은 데이터를 분석하여 유사도 계산을 통해 추천하게 된다.

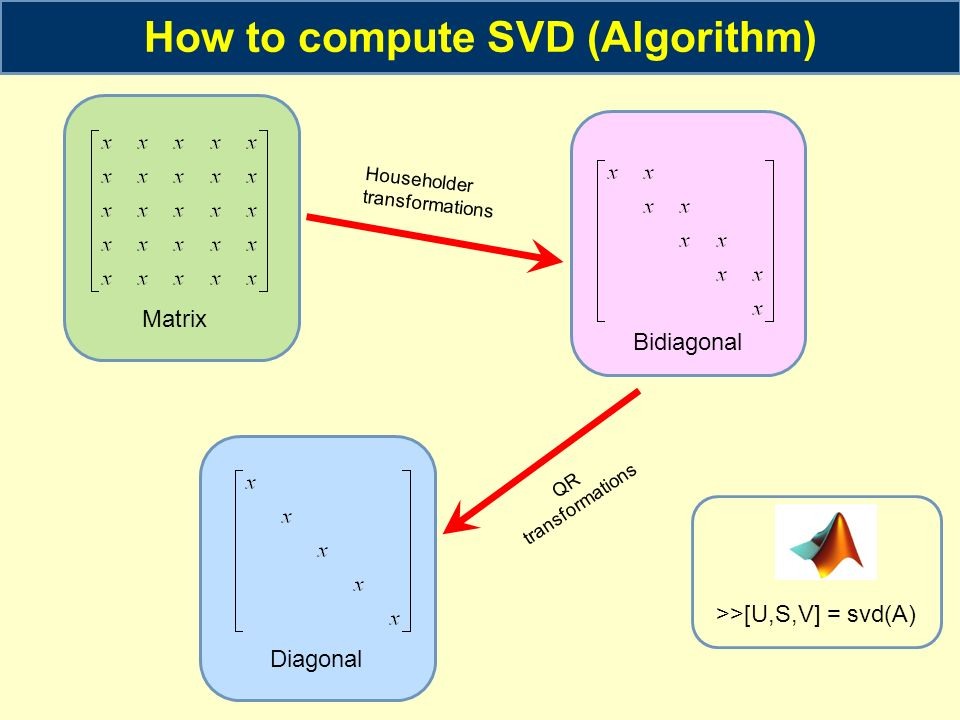
유저 기반 필터링(UBCF)는 사용자들간 공통된 아이템을 기준으로 얼마나 선호도가 일치하는지를 수치화 하여 그 점수를 사용한다. 예를 들어 A가 Z라는 블록버스터 영화에 평점5점을 주고, B가 5점을 준다면 둘의 선호도도 점수의 거리는 0이라고 할 수 있다. 만약 B가 3점을 줬다면, 두 사용자의 거리는 증가하게 된다. UBCF는 이것을 수치화 하여 더 가까운 유사도를 활용한 추천 방식이라고 볼 수 있다. 그러나 UBCF는 단점을 가지고 있는데, 만약 사용자가 최초로 시스템을 이용하는 거라면, 필요한 초기 데이터가 없기 때문에 추천을 하기가 어렵고, 또한 이런 데이터를 수집하는데 많은 시간이 소요 될 수 있다. 이러한 단점을 보완하는 것이 아이템 기반 필터링 이다.

아이템 기반 필터링(IBCF)는 추천 알고리즘에 사용자 데이터 대신 평가된 아이템 객체들이 데이터로 사용된다. 오늘날 대부분의 추천 시스템은 IBCF를 사용하고 있는데, 가장 대표적인 곳이 넷플릭스와 아마존이다. 아마존에서는 사용자가 구매한 상품들과 유사한 상품들을 선호할 것이라는 예측을 기반으로 하고 있다. IBCF는 UBCF와 달리 초기에 등록한 객체들의 데이터에 의존하기 때문에, 처음 시스템을 접하는 사용자라도 원하는 상품과 유사한 상품들을 쉽게 추천 받을 수 있다. 그러나 IBCF역시 상품들과의 유사도를 고려하였지만, 사용자들간의 선호도가 전혀 고려되지 않았기 때문에 특정 고객과 선호도가 매우 다르다면 상품들간의 추천 정확도가 떨어지고, 추천 시스템의 추천 능력이 떨어질 수 있다.

이외에도 최근에 인기가 증가하고 있는 능동적 필터링 방법이나, 미래에 가장 잠재력이 있다고 여겨지는 수동적 필터링 방법들도 존재한다. 이러한 필터링 방법들은 모두 각각 장단점이 있고, 완전하다고 볼 수 없다. 해당 필터링 방법론들의 장단점을 보기 쉽게 아래 표로 정리했다.

|  |  |  |
| --- | --- | --- |
|  | 장점 | 단점 |
| 유저 기반 필터링 | 알고리즘이 간단함, 아이템 정보 없이 추천이 가능 | 데이터가 많아질수록 연산이 복잡해짐, 신규 사용자의 데이터 공백 |
| 아이템 기반 필터링 | 아이템 자체의 정보 없이 추천 가능, 신규 사용자 추천 가능 | 데이터가 커질수록 연산이 복잡해짐, 초기 서비스 추천 정확도 떨어짐 |
| 능동적 필터링 | 신뢰성이 높음 | 편협 적인 유사도, 적은 피드백 |
| 수동적 필터링 | 능동적 필터링의 변수 제거 | 사용자 의존적 |

그렇다면 이런 협업 필터링들의 단점들을 어떻게 보완할 것인가? 가장 각광받는 것은 차원 축소 알고리즘이다. 차원 축소 알고리즘은 대표적으로 PCA(Principal component analysis), SVD(Singular Value Decomposition), NMF(Non-negative Matrix Factorization) 3가지가 있다. 이중에서 SVD(특이값 분해)라는 알고리즘을 사용하여 추천 시스템을 보완한다. SVD는 추천 시스템에 쓰이는 가장 대표적인 차원 축소 알고리즘으로, 매트릭스(행렬)을 이용하여 차원 축소를 수행한다.



SVD를 매우 핵심적이고 간략하게 설명하자면, 사용자와 상품간의 선호도를 수많은 데이터셋으로 정리하고 매트릭스로 표현한다. 이때 SVD를 사용하여, 대각 행렬을 축소하여 전체 처리해야 하는 데이터의 양을 줄일 수 있고, 아직 평가하지 않은 데이터에 대해서 평균 값을 이용하여 예상점수를 구할 수 있다. 추천 시스템은 이 점수를 이용하여 추천 점수가 높은 아이템을 사용자에게 추천하게 된다. 이런 차원 축소 알고리즘들은 다양한 상품에 대해 차원 축소를 적용함으로써 일반화 현상이 발생되고, 차원 축소가 많이 적용될수록 추천시스템은 더 효율적으로 작동한다.

# **4. 제안방법**

스크린샷이(가) 표시된 사진

자동 생성된 설명

본격적으로 프로그램을 구현하기 전에 플로우 차트를 그려 전체적인 흐름을 파악한다. 프로젝트는 크게 5개의 섹션으로 진행되며, 섹션마다 각각의 결과물이 도출된다. 최종 결과물은 구현이 완료된 프로그램이며, 테스트와 개선을 마친 상태를 의미한다.

문제 파악: Kaggle(http://kaggle.com)에서 완료된 기존 kernel의 문제점을 파악하고 개선점 파악

->**데이터 베이스 다운로드**

필요 데이터 선별: 프로젝트를 진행하며 필요한 데이터만 추출하여 데이터베이스를 가공

->**가공된 데이터 셋**

데이터 변형: 코드를 작성하며 데이터구조를 필요에 따라 변형시킨다.

->**예측 모델**

프로그램 구현: 원하는 데이터구조로 변형된 데이터셋을 이용하여 코드 작성

->**완성된 초기 모델**

테스트: 모델을 테스트 및 개선하여 최종 결과물 도출

->**구현 완료된 최종 결과물**

## - 문제 파악

협업 필터링의 단점에서 문제점을 파악한다.

단점들을 종합하여 가장 큰 두가지의 문제점을 도출할 수 있다.

**1. 사용자의 취향은 항상 고정되어 있지 않고 축적된 데이터라도 일관성은 변동이 가능하다.**

**2. 사용자들의 성향을 파악할 수 없다. 단순한 1차원적 추천시스템만 구현 가능하다.**

이러한 사용자들의 자세한 성향이나 유사취향 문제는 차원축소 알고리즘을 사용하여 해결할 수 있다. 이방법은 기존의 사용자, 아이템 기반 필터링보다 많은 계산량이 요구되기 때문에 실제 적용하기까지 시간이 소요됐지만, 현재 컴퓨터의 기술력이 발전하고 사양이 상승함에 따라 적용 가능하게 되었다.

## - 데이터베이스 다운로드

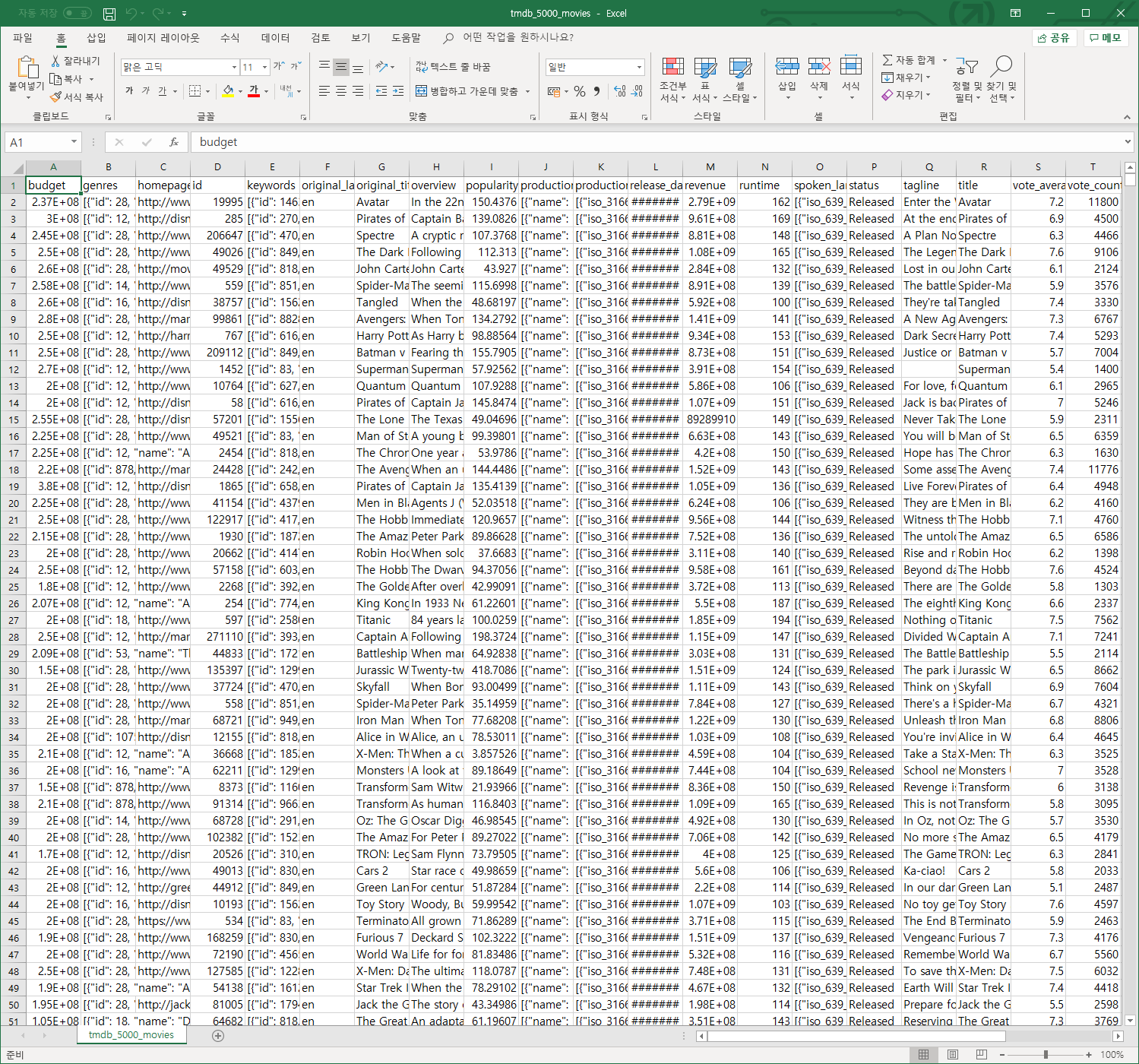
kaggle에서 제공하는 데이터 셋 다운로드

사람, 사진, 건물이(가) 표시된 사진

자동 생성된 설명

<https://www.kaggle.com/tmdb/tmdb-movie-metadata>

아마존닷컴의 자회사인 IMDB는 세계에서 가장 규모가 거대한 영화사이트를 운영한다. IMDB에서는 세계 모든 영화의 다양한 정보들을 제공하고 있다. 주연, 감독, 장르 뿐만 아니라 영화에 등장하는 장소, 음악, 수익, 심지어 모든 엑스트라 배우들의 크레딧도 제공한다. 이 데이터베이스를 선별하여 약 5000여편의 영화에 대한 데이터를 정리한 영화 데이터를 사용하기로 한다. 어떤 영화가 상업적으로 성공할 수 있는지, 어떤 영화가 높은 평가를 받는지 파악하기에 매우 좋은 자료이다.



데이터는 쉼표로 구분된 csv 텍스트 파일로 구성되어 있으며, 컬럼과 로우가 존재하는 데이터시트형태로 편집되어 있다. 컬럼들의 목록들은 다음과 같다.

* budget - 영화가 만들어진 예산.
* genre - 영화의 장르
* homepage - 영화 홈페이지의 링크
* id - movie\_id
* keywords -. 동영상과 관련된 키워드 또는 태그
* original\_language - 영화가 만들어진 언어.
* original\_title - 번역 또는 개봉하기 전의 영화 제목
* overview - 영화에 대한 간략한 설명
* popularity 영화 인기도
* production\_companies - 영화의 제작소
* production\_countries - 생산된 국가.
* release\_date - 출시 날짜.
* revenue - 창출되는 전 세계 매출액
* runtime - 영화의 상영 시간
* status - "개봉" 또는 "루머".
* tagline - 영화의 태그라인
* title 영화 제목.
* vote\_average - 영화가 받은 평균 등급
* vote\_count - 득표수

## - 필요 데이터 선별

영화를 추천해주기 위해서는 측정지표가 필요하다. 모든 영화의 점수를 계산하고 정렬한 뒤에, 사용자에게 적합한 가장 좋은 점수의 영화를 추천해준다. 하지만 현재 데이터에서 제공하는 기준치는 평균 기준치로 삼을 수 없다. 평점이 3표 밖에 없지만 8.9점인 영화와, 40표이상인 7.8점 영화영 비교할 때 3표를 받은 영화가 더 뛰어난 영화라고 볼 수 있지 않기 때문이다. 이러한 문제때문에 정확한 기준치를 계산하기 위해 IMDB 가중치를 계산법을 이용한다.

https://image.ibb.co/jYWZp9/wr.png

각각의 변수는 다음과 같다.

* v 영화가 받은 추천수
* m 차트에 적용 해야하는 최소 추천수
* R 영화의 평균 등급
* C 전체 보고서에서의 평균 추천수

위와 같은 계산법을 함수로 정의하여 데이터시트의 모든 영화별로 적용해서 컬럼에 추가한다.

## - 예측 모델에 맞게 데이터 변형

[주피터 노트북 실행 화면]

-> 최종 보고서 추가

# **5. 실험 및 결과**

## - 실험 데이터에 대한 설명(자료형태, 구조)

[주피터 노트북 실행 화면]

->최종 보고서 추가

## - 개발환경(H/W 스펙, SW 버전)

H/W

|  |  |
| --- | --- |
| 하드웨어 | 스펙 |
| CPU | i7-8700K |
| GPU | GTX-1060C |
| RAM | 8G |
| STORAGE | SSD 250GB, HDD 2TB |

SW

|  |  |
| --- | --- |
| 소프트웨어 | 버전 |
| 윈도우10 | Edu 1709 |
| Anaconda Navigator | 1.9.6 |
| Jupyter Notebook | 5.5.0 |
| Python | 3.6.5 |

## - 모델 개선

->최종 보고서 추가

## - 모델 테스트

->최종 보고서 추가

# **6. 느낀점**

현재 중간보고서를 작성하며 프로젝트를 중반까지 진행하며 많은 걸 배웠습니다. 또 많은 걸 더 배워야 한다는 걸 느끼고 있습니다. 주제로 정한 영화 추천 프로그램을 작성하기 위해서 알고리즘을 적용해야 하는데, 이 알고리즘이 상당히 많은 부분이 수학과 연관되어 있었습니다. 개선시키기로 선택한 kernel에서는 코사인 유사도를 이용하여 추천프로그램을 구현하고, 개선점으로 SVD 알고리즘을 제안하고 있었습니다. 이 부분을 직접 구현하기 위해 남은 기간동안 SVD에 대해 많이 공부하고, 또 파이선 라이브러리를 다루는 법과 문법을 많이 알아봐야 겠다고 생각했습니다.

추가적으로 Github를 활용하여 프로젝트를 진행하는 방법을 익혀 매우 효율적으로 사용하고 있습니다.

최종보고서에는 완성된 코드와 코드해석, 주피터 노트북 실행화면을 캡처하여 추가하고 더 많은 정보를 설명할 예정입니다. 또한 완성된 코드의 개선 및 테스트 과정을 진행하겠습니다.

# 

# **7. 참고**

http://darkpgmr.tistory.com/106

＃SVD(Singular value Decomposition) 설명

https://www.kaggle.com/ibtesama/getting-started-with-a-movie-recommendation-system/notebook

＃영화 추천 시스템 시작하기 kernel

https://ko.wikipedia.org/wiki/%EC%BD%94%EC%82%AC%EC%9D%B8\_%EC%9C%A0%EC%82%AC%EB%8F%84

＃코사인 유사도 위키백과

http://docs.likejazz.com/cosine-sim/

#코사인 유사도의 의미

https://ko.wikipedia.org/wiki/%ED%98%91%EC%97%85\_%ED%95%84%ED%84%B0%EB%A7%81

＃협업 필터링(collaborative filtering) 위키백과

https://m.blog.naver.com/a2zygote/220922530433

＃선형성과 비선형성의 개념

http://blog.naver.com/PostView.nhn?blogId=tjdudwo93&logNo=221051481147

＃SVD와 Kernel의 의미

https://proinlab.com/archives/2103

＃User-based CF, Item-based CF

http://leebaro.tistory.com/entry/SVD%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%9C-%EC%B6%94%EC%B2%9C-%EC%8B%9C%EC%8A%A4%ED%85%9C-%EA%B5%AC%ED%98%84%ED%95%98%EA%B8%B0

＃SVD를 이용한 추천 시스템 구현

https://www.draw.io/

＃플로우 차트 그리기

http://blog.kthdaisy.com:8080/recommendation\_system\_kthdaisy/

＃협업 필터링 알고리즘과 SVD

\*깃허브 히스토리

스크린샷, 노트북, 컴퓨터이(가) 표시된 사진

자동 생성된 설명