최종 보고서

DeepLearning을 이용한 뉴스 요약 서비스

Newsum Application

Ver. 2.0

2019.12.21

한국외국어대학교 정보통신공학과

Team name: MUD(Meeting Using Deeplearning)

무단 복제/전제 금지(MUD)

문서 정보

구분	소속	성 명	날 짜
작성자	한국외국어대학교	박주영(T)	2019.12.21
	한국외국어대학교	김아연	2019.12.21
	한국외국어대학교	김혜원	2019.12.21
검토자	한국외국어대학교	홍승환	2019.12.21
	한국외국어대학교	이산가 비두샤	2019.12.21
	한국외국어대학교	박주영(T)	2019.12.21
사용자			
승인자	한국외국어대학교	홍진표	2019.12.23

무단 복제/전제 금지(MUD)

머리말

본 문서는 묶음 뉴스 요약 본 제공 서비스인 Newsum Application시스템의 요구사항 정의서에 대한 상세 설계서를 모두 구현시킨 후 작성 된 최종 보고서이다. 요구사항 정의서와 상세 설계에서 명시 된 Deep Learning을 이용한 헤드라인 생성 서비스, Lexrank 알고리즘을 이용한 컨텐츠 생성 서비스 개인화 맞춤 뉴스 추천 서비스 등 구현하였으며 이에 대한 소개와 설계 방법, 적용 방안에 대해 자세히 기술하고 있다.

무단 복제/전제 금지(MUD)

개정 이력

이력	작성자	개정일자	개정내역
1.0	김아연	2019.11.25	초안 작성
	검토자	박주영	
1.1	홍승환	2019.11.27	서비스 개요, 기대효과 작성
	김아연		
	검토자	박주영	
1.2	박주영	2019.11.30	시스템 설명, 기능 설명
	홍승환		
	김아연		
	김혜원		
	이산가비두샤		
	검토자	박주영,	
1.3	박주영	2019.12.02	최종 보완 및 수정
	홍승환		
	김아연		
	김혜원		
	이산가비두샤		
	검토자	박주영	
2.0	김아연	2019.12.21	2차 보완

무단 복제/전제 금지(MUD)

목 차

무단 복제/전제 금지(MUD)

1. 개요

뉴스 어플리케이션 'Today news'는 인공지능 기술을 이용하여 오늘의 뉴스 요약본을 제공하고, 사용자별로 뉴스를 추천하여 준다. 본 장에서는 해당 시스템에 대한 목적과 범위, 참고문서를 소개한다.

1.1 목적

본 프로젝트에서는 인공지능 뉴스 편집 기술을 구현하였다. 기존의 사람이 직접 편집하는 방식보다 정보를 신속하게 제공하고, 사용자의 관심사를 파악하여 원하는 뉴스를 개인별로 추천해주는 시스템을 구축하는데 목적을 둔다. 이 프로젝트를 진행하기 위해 다음 사항을 구체적으로 명시하고 구현하였다.

K-means 클러스터링을 이용하여 실시간 뉴스 군집화

Lexrank 알고리즘(추출적 요약)과 Attention mechanism RNN 모델(추상적 요약)을 이용한 본문 요약과 헤드라인 추출

content-based filtering과 collaborative filtering 기술을 사용하여 사용자에게 뉴스 추천

1.2 참고 문서

문서	문헌 제목
한국정보과학회	Self-attention 기반의 다중 문서 인코더를 통한 추상적 다중 문서 요약 생성
한국정보과학회	자가 주의 메커니즘을 활용한 seq-to-seq 기반 문서 생성 요약
고려대학교 산업경영공학과/서울대학교 산업공학과	추천 시스템 기법 연구동향 분석Review and Analysis of Recommender Systems
한국정보과학회	lexrankr: LexRank 기반 한국어 다중 문서 요약
한국정보과학회	그래프 군집화기반의 다중문서요약 기법 Multi-Document Summarization Based on Graph Clustering

무단 복제/전제 금지(MUD)

2. 서비스 개요

Today News 서비스는 뉴스를 보고 싶어 하나, 뉴스를 찾아보거나 뉴스 기사 하나하나를 읽어보기 귀찮아하는 사람들을 위해 기존 뉴스들보다 접근성 있고, 시각적으로 한눈에 오늘의 뉴스를 알려주는 어플리케이션이다. 즉, 뉴스를 봐야겠다 생각은 하나, 뉴스를 보지 않는 타깃에게 오늘 하루의 뉴스를 브리핑해 주는 어플리케이션이다.

[Figure 1] 서비스 기획 배경

2.1 서비스 기획 배경

최근 Time-Poor족 즉, 시간이 부족하다고 생각하는 사람들이 많이 늘어나고 있다. 그래서 자신이 원하는 정보만을 빠른 시간 내 습득할 수 있는 요약 시장 또는 개인화 맞춤 서비스 시장이 활발해 지고 있다. 사람들은 빅데이터 시대 속에서 빠른 시간 내에 자신이 원하는 정보만을 효율적으로 얻기를 바란다. 사람들은 뉴스를 통해 가장 많은 정보를 얻고 있는데 통계에 따르면 하루에 67개의 매체에서 평균 25,866개의 기사가 생성된다. 이러한 기사들을 전부 읽을 수는 없으며 간략화 하여 접근성을 낮출 필요가 있다고 생각하여 개인 맞춤 요약 뉴스를 제공하는 Today News 어플리케이션을 기획하고 구현하게 되었

무단 복제/전제 금지(MUD)

다.

무단 복제/전제 금지(MUD)

2.2 서비스 기능 소개

본 장에서는 Newsum이 제공하는 주요 기능들을 간단하게 소개하고 세부 설명은 '5장 기능 설명'에서 이어 한다.

(1) 뉴스 요약본 제공

Today News는 사회, 정치, 경제, IT 총 4개의 카테고리로 분류된 뉴스들을 실시간으로 보여준다. 뉴스의 전문을 보여주는 것이 아닌 요약해서 단 3줄만 보여준다.

(2) 주제별 뉴스

실시간으로 제공되는 뉴스들은 비슷하거나 같은 내용의 뉴스들이 많으므로 사용자가 한 번에 주제를 파악하고 그 주제의 다양한 뉴스들을 제공받기 위해 뉴스를 주제별로 군집하여 묶어서 보여준다. 같은 주제로 묶인 뉴스들이 하나의 내용으로 요약되어 새로운 요약본을 사용자에게 보여주며 그 군집에 맞는 헤드라인을 제공하여 사용자에게 한 눈에 주제를 파악할 수 있도록 보여준다. .

(3) 개인화 추천 기능

Today News는 사용자의 경험을 중요시하고 편리함을 보장하기 위해 개인화 맞춤 추천 기능을 제공한다. 뉴스를 제공하는 홈 피드와 카테고리 피드는 추천 알고리즘을 통해 사용자의 취향에 맞는 뉴스들로 채워지게 된다. 사용자가 뉴스에 평점을 매기면 그 점수를 기반으로 추천 알고리즘이 돌아간다.

(4) 북마크 기능

사용자는 나중에 다시 읽고 싶은 기사를 북마크(스크랩)기능을 통해 저장해 둘 수 있다. 스크랩한 뉴스는 프로필창에서 확인 할 수 있다.

(5) 뉴스 브리핑 기능

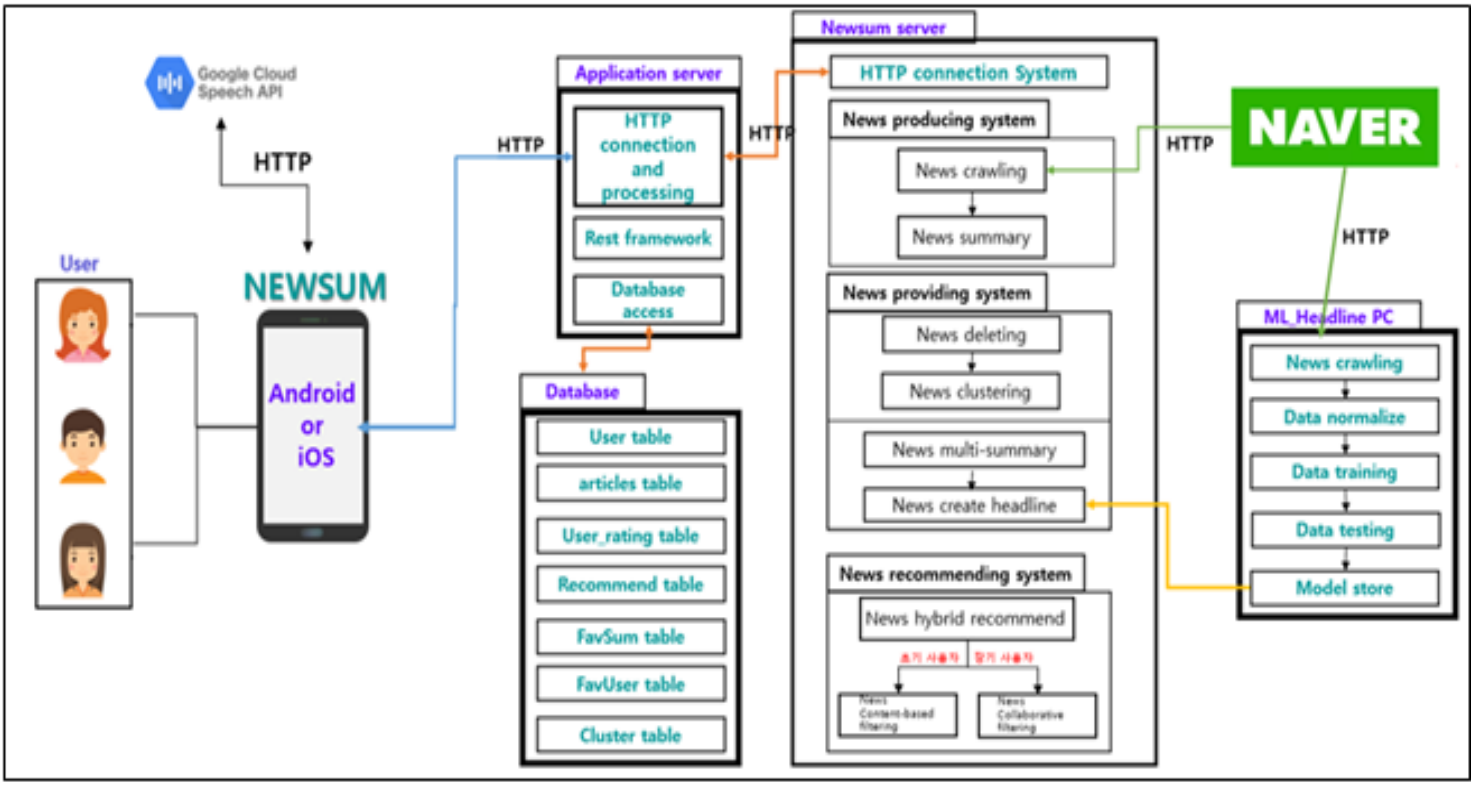
홈 피드 상단에 스피커 아이콘을 누르면 오늘 뉴스의 주제들을 브리핑 기능을 통하여 음성으로 들을 수 있다.

무단 복제/전제 금지(MUD)

3. 시스템 설명

3.1 전체 시스템 구성

[Figure 2] NewSum 시스템 구성도



무단 복제/전제 금지(MUD)

NewSum System 구성요소				
user	**App(android or iOS)**	**Application Server**	**Database**	**ML_Headline PC**
user는 NewSum System Service를 받는 대상이다.	Application Server에서 운용되는 application을 사용할 수 있는 역할을 담당한다.	application server로 App과 Database와 Newsum server 사이에 존재하여 이들 사이에서 정보를 제공하고 제공받는다.	application server에서 운용되는 database로 Newsum Server와 App 사이에서 data language를 이용해 정보를 주고 받는다.	Naver에서 제공하는 뉴스들을 토대로 dataset을 만들고 train한 모델을 Newsum server로 headline을 보낸다.

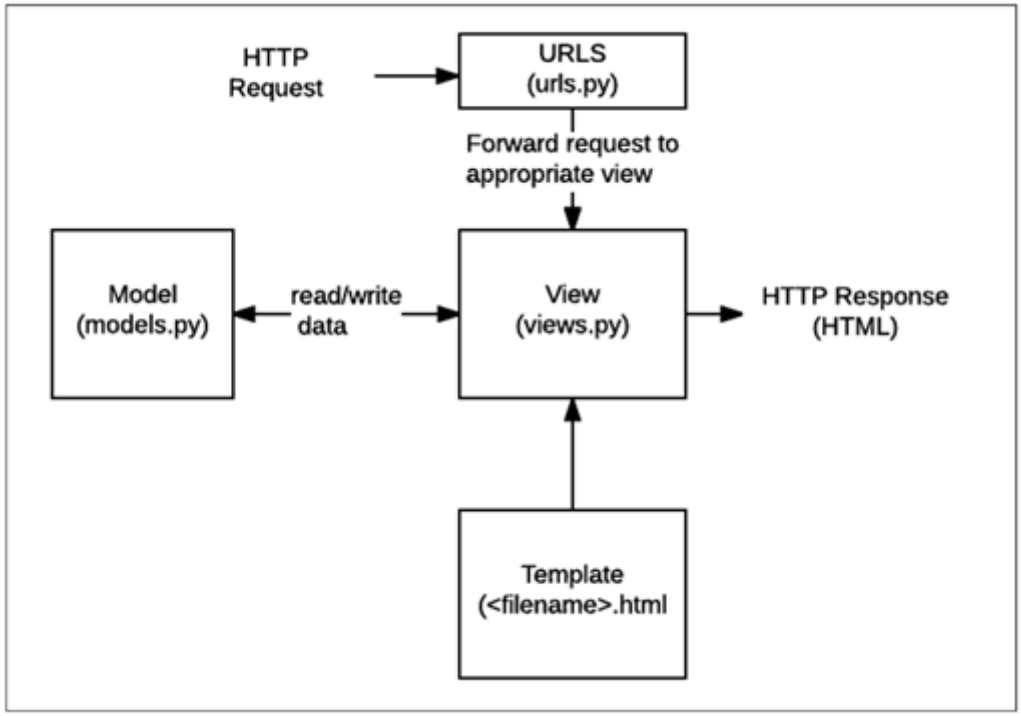
Today News는 크게 3개의 서버로 구성된다. 클라이언트와 서버(Newsum server)간 중계역할을 하는 REST API 서버인 Application server, Newsum의 전체적인 서비스 및 기능을 제공하는 Newsum server, 딥 러닝 모델을 생성하는 ml_headline PC가 있다.

무단 복제/전제 금지(MUD)

3.2 Application Server

3.2.1 Django REST Framework & REST API

Django는 python을 활용해 쉽게 웹서비스를 만들 수 있도록 하는 프레임 워크이다. Django Rest Framework는 Django를 활용하여 REST API를 구현하는데 필요한 다양한 기능들을 제공한다. REST는 웹의 장점과 HTTP의 우수성을 잘 활용하고 있는 아키텍처이며 Resource, Verb, Representation으로 구성된 아키텍처이다. HTTP URI를 통해 자원을 명시하고 HTTP Method(POST, GET, PUT, DELETE)를 통해 자원의 CURD 연산을 적용한다. 따라서 HTTP 표준 프로토콜을 따르는 모든 플랫폼에서 사용가능히다. REST API를 사용하면 프론트엔드와 백엔드를 완전히 분리할 수 있으며, 코드의 재사용성을 높일 수 있다.



3.2.2 REST API 구성

Django 웹 어플리케이션은 위의 그림과 같이 분류된 파일들에 대해 일련의 단계를 수행하는 코드로 구성되어 있다. MVC아키텍처를 따르고 있다.

무단 복제/전제 금지(MUD)

3.3 Newsum Server

Newsum server는 News Producing System, News providing System(News clustering System, Headline & Multi-summary System), News Recommend System 총 4개의 시스템으로 구성된다. 아래는 각 시스템들에 대한 간단한 설명을 표로 보였으며 시스템을 구현하는데 사용된 주요 기술과 그 적용 방법은 4장에서 기술한다.

System	설명
News Produce System	News Produce System은 실시간으로 뉴스를 크롤링 한 후 본문을 요약하여 DB에 저장하는 시스템이다. 즉, 이용자에게 제공할 낱개 뉴스를 생성하는 시스템이다. News Crawling을 통해 네이버 뉴스 속보의 정치/경제/IT/사회란에서 실시간으로 뉴스를 크롤링 해온다. 크롤링 해 온 뉴스들은 News Summary를 통해 실시간으로 다양한 뉴스들을 크롤링하여 가져와 뉴스 본문을 3줄로 요약한다. 각 뉴스마다 '뉴스 생성 시간, 헤드라인, 요약 내용, 언론사, 카테고리' 정보를 가지고 있으며 이 데이터는 linux server를 거쳐 DB에 저장된다.
News Cluster System	News Cluster System의 News clustering은 요약된 후 DB에 저장된 뉴스들을 군집화 한다. 군집이 완료된News delete를 통해 지워진다. 클러스터링이 완료된 뉴스들은 DB의 Cluster table에 저장된다.
Headline & Multi-summary System	Headline & Multi-summary에서 News multi-summary를 통해 같은 주제로 군집화 된 뉴스들을 대표하는 새로운 요약 뉴스를 생성한다. News headline processing에는 딥러닝 학습을 거쳐 미리 저장된 모델이 있으며 이 모델은 새로운 요약 뉴스에 맞는 새로운 제목을 생성하여 준다.
News Recommend System	뉴스 추천 시스템은 앞의 시스템들을 거쳐 재가공 되어진 뉴스를 사용자에게 개인 맞춤으로 추천해주는 시스템이다. 뉴스 데이터 양에 따라 contents-based recommendation과 Collaborative recommendation을 번갈아 사용하게 된다.

무단 복제/전제 금지(MUD)

무단 복제/전제 금지(MUD)

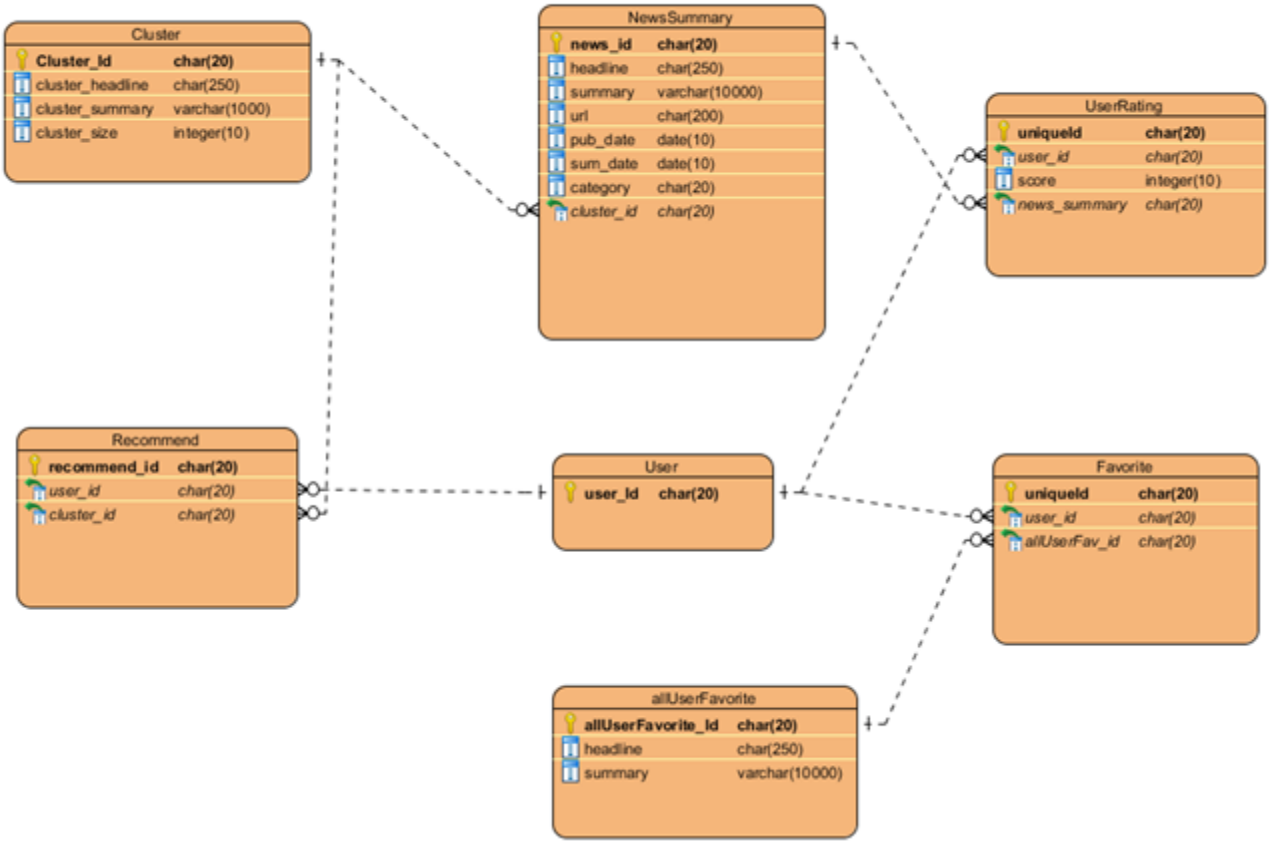
3.4 ML_Headline PC

ML_Headline PC는 머신러닝 컴퓨터로 군집 뉴스의 Headline을 만드는 모델을 생성한다. 모델 학습을 실행시키고 모델을 자주 업그레이드 하기 위해 딥 러닝 모델을 처리할 정도의 GPU사양을 가진 컴퓨터를 사용하였다. NewSum만의 딥러닝 시스템 ML_Headline PC를 두어 비용 효율성을 높였다.

무단 복제/전제 금지(MUD)

3.5 Database 구성

3.3.1 Database 설계도(MySQL)



[Figure 12] 시스템 구성 요소 4.6: Database 설계도

news crawling과 news clustering, ML processing, news recommending 각 REST API를 통해 받은 data를 수집하여, Database에 갱신한다. Django의 models.py에서 위의 table들을 생성한다.

무단 복제/전제 금지(MUD)

3.3.2 Database 구성 표

models.py에서 table과 field들의 attribute를 정의하고 python manage.py migrate를 통해 Database에 table을 생성한다.

Table Name	Attribute	
articles	News_id(PK)	Crawling한 뉴스 고유의 uuid4부여
	Headline	Crawling한 뉴스의 제목
	Summary	Crawling한 뉴스의 본문을 lexrankr 알고리즘을 이용하여 본문 요약 content
	url	Crawling한 뉴스의 URL주소
	Pub_date	Crawling한 뉴스의 생성 날짜[뉴스기사에 포함 되어있는 날짜를 의미함]
	Sum_date	Crawling한 뉴스의 본문 요약 완료된 날짜
	Category	Crawling한 뉴스의 카테고리 이름[economy, politics, IT/science, society]
	Cluster_id(FK)	Cluster table의 cluster_id값을 참조한다[FK] [clustering이 실행되는 시간은 정해져 있기 때문에 cluster되기전에 저장된 news들의 값은 default값으로 부여 받는다] [또한 cluster에 참여하지 못한 뉴스들도 default값으로 부여 받는다]
Cluster	Cluster_id(PK)	개별 뉴스들이 cluster(군집화)되어 군집화된 뉴스들에게 고유의 ID값(uuid4)를 부여한다
	Cluster_headline	Cluster 다중 문서 요약된 content를 통해 ml_headline_model에 input 값으로 넣어 output으로 headline을 얻는다. 얻은 headline은 cluster_headline에 저장된다.
	Cluster_summary	Cluster된 뉴스들을 다중 문서요약에 특화된 lexrankr를 통해 다중 문서 요약 content를 cluster_summary에 저장한다
	Cluster_size	Cluster된 뉴스들의 크기를 측정하여 순위를 매겨준다. Cluster 내 존재 가능 개별뉴스는 최대 40개이므로 Cluster_size 또한 1부터 40까지의 범위를 가진다.
Recommend	Recommend_id(PK)	Recommending system의 output으로 cluster의 id를 저장한다.
	User_id(FK)	User table의 user_id를 참조한다

무단 복제/전제 금지(MUD)

	Cluster_id(FK)	Cluster의 cluster_id를 참조한다.
User	User_id	FirebaseDatabase에서 받은 user_id에 대한 고유의 id값(uuid4)이다
FavSum	AllUserFavorite_id(PK)	모든 회원들이 스크랩한 뉴스들의 Newssummary의 news_id값을 참조하지 않고 복사하여 저장한다[이 AllUserFavorite_id _id의 값은 영원히 지워지지 않는 data이다]
	Headline	모든 회원들이 스크랩한 뉴스들의 Newssummary의 headline을 참조하지 않고 복사하여 저장한다[이 headline의 값은 영원히 지워지지 않는 data이다]
	summary	모든 회원들이 스크랩한 뉴스들의 Newssummary의 summary를 참조하지 않고 복사하여 저장한다[이 summary의 값은 영원히 지워지지 않는 data이다]
FavUser	uniqueId(PK)	User가 뉴스를 스크랩할 시 부여 받는 고유한 id이다.
	allUserFav_id(FK)	allUserFavorite table의 allUserFav_id를 참조한다.
	User_id(FK)	User table의 User_id값을 참조한다.
UserRating	uniqueId(PK)	User가 개별 각각 뉴스에 대해 평점을 매길 시 부여 받는 고유한 id이다
	User_id(FK)	User table의 usr_id를 참조하고 있다.
	Score	회원이 개별 각각 뉴스에 대해 1~5점사이의 점수를 부여하면, 그 값이 score에 저장된다[이 score는 recommending system에서 사용된다]
	News_summary(FK)	Newssummary table의 news_id를 참조하고 있다

models.py에서 table과 field들의 attribute를 정의하고 python manage.py migrate를 통해 Database에 table을 생성한다.

무단 복제/전제 금지(MUD)

3.3.3 Database 저장 형태

Table Name	저장 형태
newssummary	
Users	

무단 복제/전제 금지(MUD)

최종보고서: Machine Learning을 이용한 뉴스 요약 및 추천 어플리케이션(Today News)

Cluster


```
{
    "cluster_id": "1f4f3d79-192a-409c-b824-091ae97bfccd",
    "cluster_headline": "This is a default Cluster Headline",
    "cluster_summary": "This is a default Cluster Headline"
},
{
    "cluster_id": "3e93d95e-543a-49e7-9a63-ae604fb535a8",
    "cluster_headline": "세월호 유가족 '폭식투쟁' 참가자 분기소 처분에 반발",
    "cluster_summary": "권도현 기자 단식농성을 하던 세월호참사 유가족 앞에서 '폭식 투쟁'을 벌여 유가족과 시민단체로부터 고발당한 참가자를 검찰이
},
{
    "cluster_id": "58d0aa77-1664-4bed-9f45-0e28e8b11171",
    "cluster_headline": "한미 방위비 분담금 3차 회의...본격 협상시작",
    "cluster_summary": "한미 대표단 4시간여 회의… 본격적인 협상 3차 회의 내일까지 진행…연말 타결 미지수 앵커 우리나라와 미국이 내년부터 적용할 방
},
```


UserRating


```
{
    "rating_id": "055c8196-9b47-4289-8592-a586a7625696",
    "score": 5,
    "user_id": "pFbDe7m55DYtXMgcar18FgEFhYr1",
    "news_summary": "b47b1456-d816-4f30-8d5c-9d5846a79bb9"
},
{
    "rating_id": "4f71ee52-e260-4382-8b30-5e9847765b6b",
    "score": 2,
    "user_id": "R7TdA4s1DiO5QwN7iTJL1ad6e2V2",
    "news_summary": "6fd0f744-5758-45c1-94c5-84a69b893429"
},
{
    "rating_id": "6d501bd3-5078-4e36-ab70-401a2bf8e87f",
    "score": 4,
    "user_id": "GNwo1CGPbOdxumpO4uDYXkNGL1H2",
    "news_summary": "6fd0f744-5758-45c1-94c5-84a69b893429"
},
```


Recommend


```
{
    "recommend_id": "862c007f-d49d-4265-8bce-814f7d2e18df",
    "user_id": "pFbDe7m55DYtXMgcar18FgEFhYr1",
    "cluster_id": "58d0aa77-1664-4bed-9f45-0e28e8b11171"
},
{
    "recommend_id": "b77947da-6355-48da-b2b1-7c0a7f729e10",
    "user_id": "pFbDe7m55DYtXMgcar18FgEFhYr1",
    "cluster_id": "dbd18b3f-6ae2-4a2c-820e-5c815cd2662c"
},
{
    "recommend_id": "e97eb3b4-5876-4bc1-8079-2897b25a0aac",
    "user_id": "pFbDe7m55DYtXMgcar18FgEFhYr1",
    "cluster_id": "3e93d95e-543a-49e7-9a63-ae604fb535a8"
}
```


무단 복제/전제 금지(MUD)

4. 주요 기술 설명 및 적용 방법

4.1 Crawling

크롤링이란 컴퓨터 소프트웨어 기술로 웹 사이트에서 원하는 정보를 추출하는 것을 의미하며, 크롤러는 인터넷의 웹페이지를 방문해서 수많은 자료들을 수집하는 일을 하는 프로그램을 말한다. 사람이 모든 웹 사이트를 방문하며 데이터를 수집하는데 한계가 있기 때문에 크롤러는 사람이 일일이 하는 대신 이를 자동으로 수행한다.

Newsum은 실시간으로 뉴스 정보를 수집하는 자체 크롤러를 구현하며, 이는 News producing System의 클래스로 정의된다. 아래의 4.1.1장에서 Newsum Crawler 설계 방법에 대해 자세히 기술한다.

4.1.1 Newsum Crawler 구현

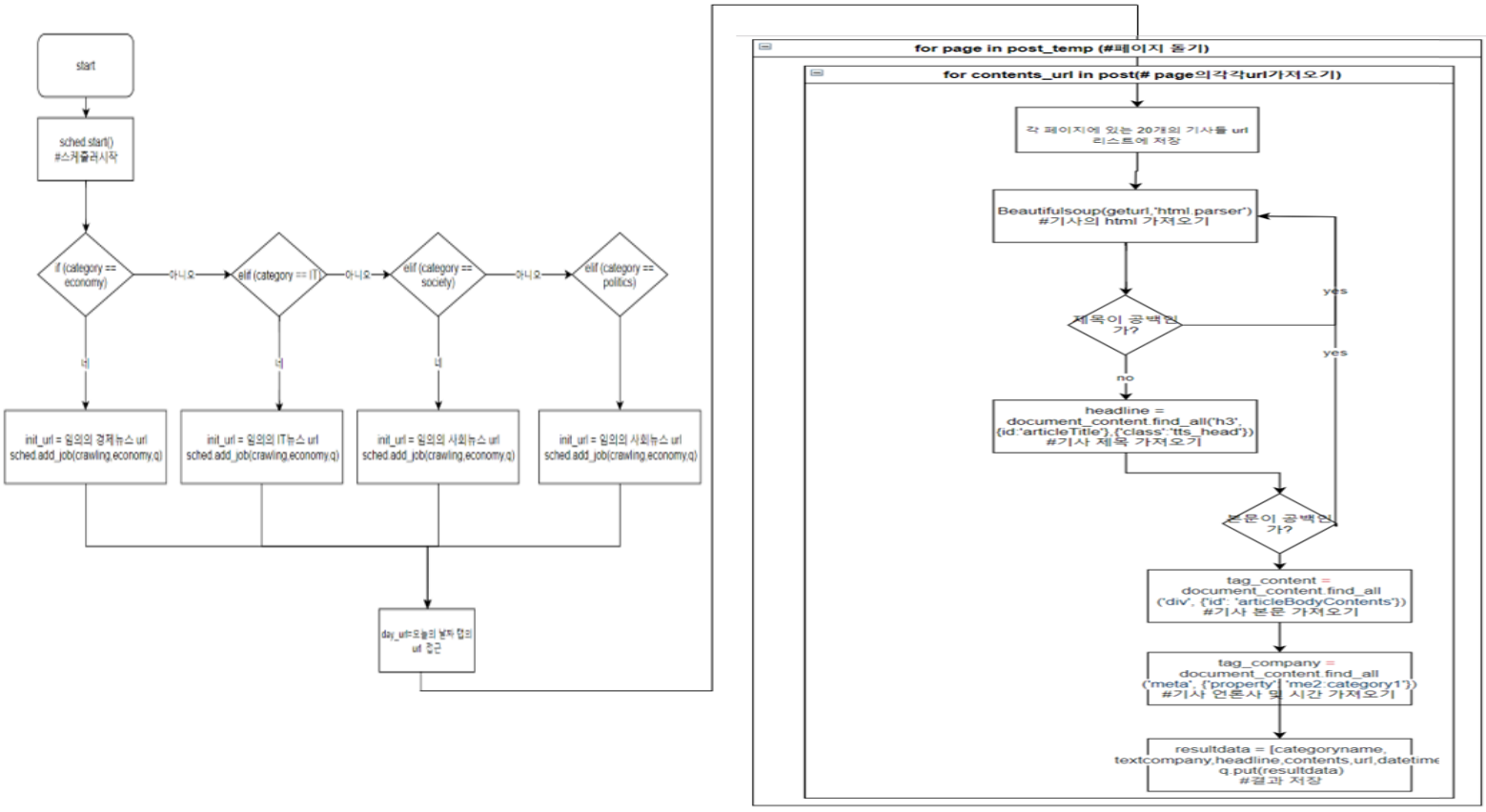


Figure 11 [알고리즘 2-1 _News crawling]

무단 복제/전제 금지(MUD)

① 네이버뉴스의 속보란에서 경제, 사회, 정치, IT 카테고리내의 오늘의 뉴스를 수집한다.

② 스케줄러를 이용하여 일정 시간 간격으로 계속하여 crawler함수를 실행시킨다. 경제: 5분, 사회: 3분, 정치: 7분, IT:15분 간격으로 크롤링을 실행한다.

```
##경제 :5분/ 사회:3분/ 정치:7분/IT:15분.
if "economy" in category:
    old.append(category["economy"])
    sched.add_job(Crawler.crawling, 'cron', minute="5", id='test_1',args=["economy", q])  # argssms 배열로 넣어주어야한다.
if "IT_science" in category:
    old.append(category["IT_science"])
    sched.add_job(Crawler.crawling, 'cron', minute="15", id='test_2', args=["IT_science", q])  # argssms 배열로 넣어주어야한다.
if "society" in category:
    old.append(category["society"],)
    sched.add_job(Crawler.crawling, 'cron', minute="3", id='test_3', args=["society", q])  # argssms 배열로 넣어주어야한다.
if "politics" in category:
    old.append(category["politics"])
    sched.add_job(Crawler.crawling, 'cron', minute="7", id='test_4', args=["politics", q])  # argssms 배열로 넣어주어야한다.
```

③ 크롤러의 url 접근 순서 : 오늘의 날짜 탭 url 가져오기 > 페이지 url 가져오기>페이지 내 각 기사의 url 가져오기 > html을 이용하여 기사 제목,본문,날짜,언론사 가져오기

④ 이때 크롤링된 내용은 resultdata = [news title, news contents, category, date, URL, publication]로 리스트 형식으로 저장되며 이 값은 q.put(resultdata)로 큐에 저장된다.

무단 복제/전제 금지(MUD)

4.2 Lexrankr을 이용한 뉴스 요약

Lexrank는 Textrank기법에 그래프 클러스터링과 새로운 유사도 함수를 적용한 알고리즘으로 대용량 문서 요약과 다중 문서 요약에 적용 가능하다. tf-idf벡터 기반으로 미리 클러스터링 가능하지만 lexrank는 그래프를 구성 후 그래프 클러스터링 알고리즘을 적용한다. NewSum은 추출적 요약 기법인 Lexrank 알고리즘을 이용하여 기사 원문에서 3줄 요약문을 만들어 낸다. (설문조사에 의하면 모바일에서 이용자가 좋은 뉴스라고 생각하는 뉴스의 길이는 3줄-5줄이다.)

Newsum은 Lexrankr을 이용하여 2번의 요약을 한다. 각각의 요약을 News Summary(1차 요약), News multi-Summary(2차 요약)이라 부른다. 1차 요약은 News Producing system에서 정의되며 크롤링 후 가공이 되지 않은 원본 뉴스들의 본문을 3줄로 요약하여 낱개 뉴스의 요약본을 생성하는 것이다.

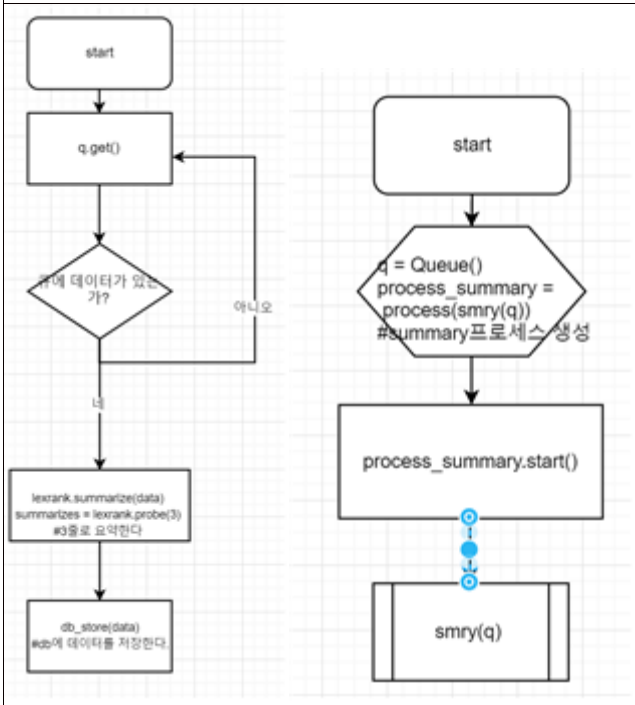
2차 요약은 News Providing System에서 정의되며 낱개 뉴스들을 클러스터링 한 군집 뉴스들의 3줄 요약본을 생성하는 것이다. 군집 내 낱개 뉴스 4개를 랜덤으로 선택하여 요약하는데 이 때(다중 문서 요약을 할 때) Lexrankr이 textrank나 그 외 다른 요약 알고리즘에 비해 성능이 뛰어난 것을 밑의 결과를 통해 알 수 있다.

방법 - 데이터셋	F1	ROUGE-1	ROUGE-2
기준 - 짧은 문서	50.4	61.1	51.5
제안 - 짧은 문서	**59.2**	**75.8**	**67.0**
기준 - 긴 문서	36.9	51.8	37.6
제안 - 긴 문서	**47.6**	**68.0**	**56.8**

위 도표에서 기준은 기존에 존재하는 요약 알고리즘 texkrankr와 제안하는 lexrankr 알고리즘 둘 중에서 어느 것이 더 한국어 다중문서 요약에 적합한지 보여지고 있다. 결과적으로 lexrankr 알고리즘이 다중 문서요약에 적합하다는 것을 알 수 있었고 따라서 Newsum은 Lexrank 알고리즘을 통해 단일 문서 요약 뿐만아니라 다중 문서요약도 진행하였다.

무단 복제/전제 금지(MUD)

4.2.1 News Summary 구현 (1차 요약)

News Summary 구현 과정	
	① 멀티프로세싱을 이용하여 crawler()와 smry()함수를 동시에 실행시킨다. 큐를 사용하여 데이터를 전달할 것이기 때문에 q = Queue()를 생성한 후 summary 프로세스를 생성한다. ② Queue에 데이터가 있는 경우, 데이터를 data = q.get() 으로 가져온다. Lexrank를 이용하여 3줄로 요약한다. 그 후 db_store()함수를 통해 데이터를 summarynews 디비 테이블에 [category_name, text_company, text_headline, summary,url, news_datetime]값을 업데이트 한다.

4.2.2 News Multi-Summary 구현 (2차 요약)

다중 뉴스 요약 구현 과정

```
if __name__ == "__main__":
    q1 = Queue()
    q2 = Queue()
    sched = BackgroundScheduler()

    sched.start()
    sched.add_job(random_four_news, 'cron', hour='1', id="s1", args=[df, cluster_id, q1])
    sched.add_job(random_four_news, 'cron', hour='12', id="s2", args=[df, cluster_id, q1])
    sched.add_job(random_four_news, 'cron', hour='15', id="s3", args=[df, cluster_id, q1])
    sched.add_job(random_four_news, 'cron', hour='18', id="s4", args=[df, cluster_id, q1])
    sched.add_job(random_four_news, 'cron', hour='21', id="s5", args=[df, cluster_id, q1])
    sched.add_job(random_four_news, 'cron', hour='24', id="s6", args=[df, cluster_id, q1])
    #process_four = Process(target=random_four_news, args=(q1,))
    process_summary = Process(target=summary, args=(q1,q2,))
    process_mlheadline = Process(target=ml_headline, args=(q2,"test",))

    #process_four.start()
    process_summary.start()
    process_mlheadline.start()

    q1.close()
    q2.close()
    q1.join_thread()

    #process_four.join()
    process_summary.join()
    process_mlheadline.join()
```

① -1. random_four_news는 클러스터링 된 후 진행되야 하기 때문에 (1시,12시,15시,18시,21시,24시)에 진행된다. random_four_news함수와 summary함수는 queue q1을 공유하고 병렬로 진행된다(동시에 진행) ➔ 위 그림에서 파란색 박스 부분에 해당된다

① -2. summary함수와 ml_headline함수는 queue q2를 공유하고, 이 두 함수는 병렬로 진행된다(동시에 진행) ➔ 위 그림에서 빨간색 박스 부분에 해당된다

```
print("random_four_news start")
articles = requests.get(url="http://34.84.147.192:8000/news/articles/?format=json&limit=1").json()
count = articles['count']
print(count)

articles = requests.get(url="http://34.84.147.192:8000/news/articles/?format=json&limit=" + str(count)).json()
d = pd.DataFrame(articles['results'])

df = pd.DataFrame(columns=['news_id', 'summary', 'cluster_id'])
df = d[['news_id', 'summary', 'cluster_id']]
cluster = requests.get(url="http://34.84.147.192:8000/news/clusters/").json()
#cluster = df['cluster_id']
#print(cluster)
cluster_id = list([i["cluster_id"] for i in cluster if i["cluster_id"] != '07f269a8-3ae6-4994-abfd-e2cb2d4633f3'])
```

② 다중 뉴스 요약을 하기 위해 DB cluster table에서 default cluster_id값을 제외한 cluster_id를 가져오고, article table에서 news_id, summary, cluster_id 세부분만 pd 형식으로 저장한다.

무단 복제/전제 금지(MUD)

```
for cluster_id_1 in cluster_id:
    print("x",cluster_id_1)
    df2 = df[df["cluster_id"] == cluster_id_1]
    df_news = df2['news_id'].tolist()
    news_id = random.sample(df_news, 4)
    for news in news_id:
        df3 = df[df["news_id"] == news]
        df_summary += df3['summary'].tolist()

    tmp.append(cluster_id_1)
    tmp.append(" ".join(df_summary))
    #print("TTTTTTtmp",tmp)
    q1.put(tmp)
```

③-1. 위에서 article table을 pd형식을 저장한 df에서 cluster_id와 같은 news_id들만 뽑는다.

③-2. 뽑은 news_id들 중에서 random으로 4개 뉴스를 뽑아, 4개의 news_id에 해당하는 summary(뉴스 1차 요약본)을 가져와 하나의 string으로 만들어 queue q1에 put한다 →q1.put(tmp)

```
def summary(q1,q2):
    while True:
        print("smry start")
        multi_summary = q1.get()
        lexrank = LexRank()
        lexrank.summarize(multi_summary[1])
        summaries = lexrank.probe(3)  # 3줄
        summaries = '. '.join(summaries)+'.'
        print("multi-summary= ",summaries)
        multi_summary[1] = summaries
        q2.put(multi_summary) # db에 저장되어
```

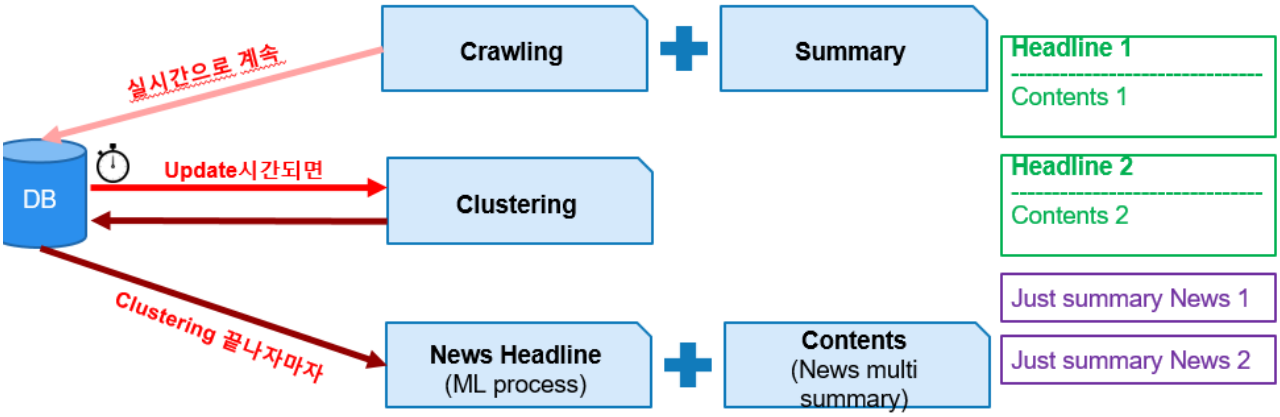
④-1. summary함수에서 queue가 차 있으면 q1.get()한다

④-2. q1.get한 data에서 content부분인(multi_summary[1])을 다중 문서 요약하기 위해 lexrank 알고리즘을 이용하여 12문장을 3문장으로 요약한다. 요약한 3문장은 다시 multi_summary[1]로 지정하고 queue q2에 put한다 → q2.put(multi_summary)

무단 복제/전제 금지(MUD)

4.3 K-means Clustering을 이용한 News Clustering

k-평균 알고리즘(K-means algorithm)은 주어진 데이터를 k개의 클러스터로 묶는 알고리즘으로, 각 클러스터와 거리 차이의 분산을 최소화하는 방식으로 동작한다. 이 알고리즘은 label입 없는 데이터의 입력을 받아 각 데이터에 label을 할당하여 군집을 수행한다. 개념이 매우 간단하며 실행속도가 빠르고 특정한 데이터에는 좋은 성능을 보인다. 하지만 k를 개발자가 직접 설정해주어야 하며 최적의 k를 구하는데는 많은 시간이 걸린다는 단점이 있다. NewSum에서는 최적의 k를 구하는 시간을 줄여 성능을 높이는 법을 실험하고 적용해보았다.



이때 Newsum은 24시간의 실시간 뉴스를 군집 뉴스로 브리핑 해주므로 실시간으로 계속 크롤링 된 것 중 24시간 분량의 정보만 있으면 된다. 따라서 News producing System에서 제공된 뉴스 data를 News providing System에서 24시간 양의 data로 줄여서 clustering해준다. 따라서 News providing System에서는 1차적으로 News deleting을 하고, 2차적으로 News clustering을 하여 군집 뉴스의 군집의 기준을 지정해준다. 또한 이후 과정에서 나눠진 군집에 대한 Headline과 Contents를 다루도록 하겠다.

이때 News clusering은 K_class.py라는 Kmeans module을 활용한 자체 class로 만들어 실제 코드에서 class를 import 하여 사용하게 구현했다.

4.3.1 News deleting 기능에 대한 상세 설계

스케줄러에 의해 News providing System이 시작된다. 그 안에서 News deleting이 먼저 시작된다. 즉, News clustering이 이루어 지기 전에 전달받은 articles tables의 data 중 pub_date시간을 기준으로 업데이트 시간보다 24시간 전 뉴스data는 모두 삭제 시킨다.

무단 복제/전제 금지(MUD)

4.3.2 전처리 과정

News deleting 이 모두 완료되면 순차적으로 News clustering 이 시작된다. News clustering 은 크게 3.4.2 전처리 과정, 3.4.3 최적의 K 값 찾기, 3.4.4 근접 기사 찾기, 3.4.5 DB 저장으로 설계했다.

요구사항에서 명시한 것과 같이 군집화(Clustering)은 자연어 처리 기술로 전처리 과정에 따라 성능이 좌우된다. News clustering 기능에서 중요한 것은 빠른 속도와 정확성이다. 따라서 이 설계 목표에 맞춰 전처리 과정에서 변수를 조절하여 실험 후, 성능이 좋은 방법을 채택하여 구현하였다.

전처리과정은 '①형태소 분석 & 토큰화 처리 ②처리 불용어 제거 ③Vectorization ④정규화'로 prorecessing class 에 있다

```
#토큰화함수
def tokenizer(self,raw,pos=["Noun"], stopword ='[^가-힣ㄱ-ㅎㅏ-ㅣa-zA-Z]'):
    import pandas as pd
    pd.options.mode.chained_assignment = None
    import numpy as np
    np.random.seed(0)
    from konlpy.tag import Okt
    twitter = Okt()
    return [
        word for word, tag in twitter.pos(
            raw,
            norm=True,  # normalize 그랰ㅋㅋ -> 그래ㅋㅋ
            stem=True  # stemming 바뀌나->바뀌다
        )
        if len(word) > 1 and tag in pos and word not in stopword
```

Equation 1 [News clustering_전처리 과정 1]

Equation 1 tokenizer 함수에서 ①형태소 분석 & 토큰화 처리 ②처리 불용어 제거 과정을 다룬다 Konlpy module 의 twitter(현재 Okt() 이름이 바뀜)로 한국어 형태소 분석을 하고 pose =["Noun"]으로 토큰화 처리를 한다. 또한 stopword 로 특수문자 및 영어 숫자를 제거하여 불용어 제거를 한다.

```
def result(self):#rawDatas는 call_Dataset Class의 data_in_Category 함수로 사용
    from sklearn.feature_extraction.text import CountVectorizer
    #CountVectorizer로 벡터화, 이때 토큰화는 Class 내 tokenizer 사용
    vectorizer = CountVectorizer(tokenizer = self.tokenizer)
    X=vectorizer.fit_transform(self.data)
    #벡터 정규화처리
    X=normalize(X)
    # X=X.toarray() : distoration구할때 !
    return X
```

Equation 2 [News clustering_전처리 과정 2]

Equation 2 result 함수에서 앞서 말한 tokenizer()함수를 불러 데이터를 한번 가공하고 이후 CounterVectorizer 라는 Tfidfvectorizer 모듈로 벡터화를 한 후 정규화 처리를 해준다. 이 결과로 X 라는 전처리 과정을 마친 Dataset 으로 바뀌진다.

무단 복제/전제 금지(MUD)

4.3.3 최적의 K값찾기

clustering에서 활용하는 Kmeans module의 최대 단점은 K값 즉, cluster의 개수를 정해줘야만 하는 것이다. 이때 이 K값에 따라 정확성이 달라지므로 최적의 K를 빠른 시간 내에 찾는 알고리즘을 설계하였다.

최적의 K값을 찾는 방법에는 요구사항에서 명시한 elbow 기법, 반복문을 써서 가장 성능(SSE) 차이가 많이 나는 지점을 적절한 K라고 지정한다. 이를 위해 optimal_K()라는 함수를 만들어 elbow 기법을 적용시킨다. 하지만 Newsum의 뉴스기사 Dataset은 24시간의 4가지 카테고리의 정보량으로 약 2만5천개의 기사들로 용량이 크므로 dataset을 모두 반복문을 돌리면서 K means module을 실행시키기에는 처리속도가 낮다. 또한 Dataset의 용량이 클수록 K means module 자체적으로도 수렴하는 적절한 clustering이 이루어 지기가 힘들다.

여기서 문제가 되었던 정확도란 Kmeans module의 자체 인자 값인 n_init을 통해 변화를 줄 수 있다. n_init이란 Kmeans module이 자체적으로 dataset 내의 임의의 점을 시작으로 지정해준 K만큼 군집을 만들기 시작하는데 이대 임의의 점들의 정해주는 횟수를 말한다. 즉 n_init의 값이 높을수록 랜덤으로 주는 시작점이 많아지고 즉 반복학습이 많아지므로 더 적절한 군집들로 수렴한다. 하지만 반복학습이 많아짐에 따라 또한 속도 역시 느려지는 단점이 있다.

따라서 이를 해결하기 위해 Newsum의 News clustering에서는 '구간 별 정확도 나누기' 방법으로 시간은 단축시키되 성능은 좋게 유지할 수 있는 elow 기법 최적의 K를 찾는 optimal_K class를 구현하였다.

```
#2단계 구간 나누기 --> 단기간 최적의 K c찾는함수
def final_find_K(self):
    #1차 구간나누기 : 6칸씩 300으로
    first_K, first_index,first_length = self.find_K(self.sK,self.eK,6,200)

    #2차 구간나누기 : 3칸씩 400으로
    if (first_index == first_length): #1차구간중 마지막 구간에 해당한다면
        second_K,second_index,second_length = self.find_K(first_K-5,self.eK,3,300)
    else:
        second_K,second_index,second_length = self.find_K(first_K-5,first_K+5,3,300)

    #3차 구간나누기 :
    thrid_K =self.find_K(second_K,second_K+3,1,600)[0]
    print('final_optimal_K',thrid_K)
    return thrid_K
```

Final_find_K()합수에서 1차,2차,3차 구간을 나누어 n_init값을 200, 300, 600으로 다르게 주었고 속도 향상을 위해 구간 내에서도 step 값으로 6, 3, 1칸씩 건너뛰어 값을 구했다.

SSE 값 차이를 이용한 elbow 기법으로 K를 찾는 자체 함수 find_K()에 이렇게 구간별 다른 변수 값을 인자로 주어 적절한 최정 K인 third_K를 구하였다.

무단 복제/전제 금지(MUD)

4.3.4 근접 기사 찾기

3.4.3까지의 과정을 통해 Database에서 가져온 24시간 Dataset을 preprocessing class를 통해 전처리 과정 , optimal_K class를 통해 최적의 K값까지 찾았다. 이를 main class 인 run_Kmeans class의 play함수에서 실행 시킨 후 가공처리 된 Dataset인 X와 최적의 K를 kmeans module을 통해 만든 finalkmeans인 kmeans model의 인자로 넣는다. 이의 결과로 finalkmeans은 적절한 K값으로 전처리한 X를 clustering 한 모델을 제공한다.

```
def play(self):
    p =preprocessing(self.rawData)
    preprocessed_Data =p.result()
    o =optimal_K(preprocessed_Data,self.sK,self.eK)
    k = o.final_find_K() #최적의 K
    finalkmeans =KMeans(n_clusters=K, init='k-means++',n_init=600).fit(preprocessed_Data)

    label =finalkmeans.labels_
    labelsize ={}
    for i in set(label):
        labelsize[i] = label.count(i)

    distance = finalkmeans.transform(preprocessed_Data)
    c=closed_news(finalkmeans,label,distance)
    default_index = c.default_index_news() #default를 줘야하는 뉴스기사 index를 리스트
```

이때 Kmean module의 자체 기능인 labels_을 이용하여 전처리된 dataset인 X의 인덱스 별로 cluster number을 알 수 있다.

이 label을 통하여 자체 근접 기사 찾기 closed_news class를 통해 근접 기사가 아닌 default 값을 지정 해 줘야하는 뉴스 index를 돌려 받는다.

```
    default_cluster=[]
    for i in range (len(self.label)):
        dis_list_in_cluster = self.dis_list[base:base+self.label.count(i)]
        dis_list_in_cluster.sort(key=lambda x: x[2])
        base += self.label.count(i)
        # cluster_id 당 기사 개수가 40개가 크면 (default값으로 부여해줌)
        if(self.label.count(i)>40):
            for m in range(40 , self.label.count(i)):
#거리 정렬순으로 했을때 해당 cluster_id 속 40번째가 초과되는 뉴스기사들의 index를 default_cluster 리스테 넣어주기
                default_cluster.append(dis_list_in_cluster[m][0])
    return (default_cluster)
```

이때 근접 기사란 요구사항에서 명시한 것과 같이 클러스터 내부 화면에서 보여지는 개별 뉴스를 의미 한다.

따라서 한 개의 클러스터에서 존재하는 개별 기사는 최대 40개로, centroid와 가까운 순으로 순서를 매겨 40번째 까지의 뉴스data만 근접 기사로 생성된 cluster id를 부여하고, 나머지 값들은 모두 default cluster id를 주어 카테고리 화면에서 보여지는 개별 뉴스로 보여진다.

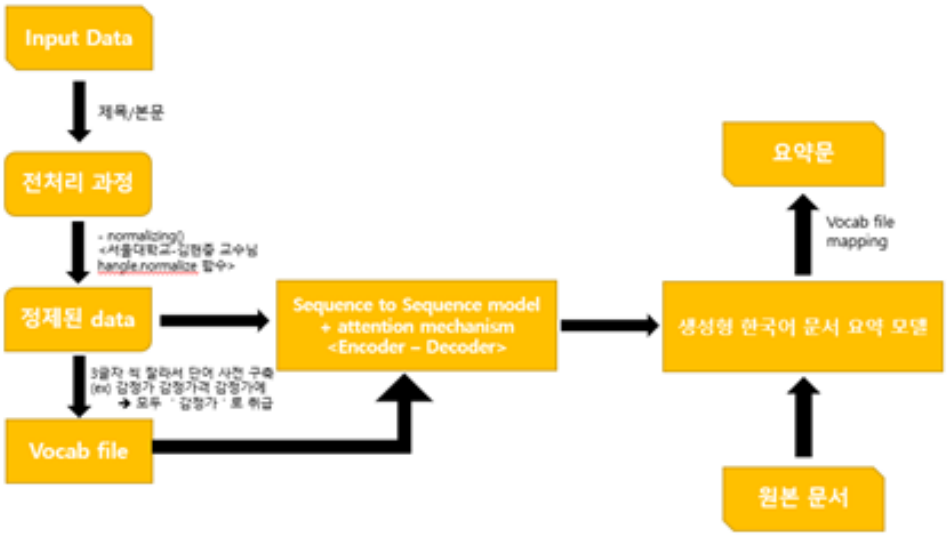
무단 복제/전제 금지(MUD)

4.4 Attention Mechanism을 이용한 뉴스 헤드라인 재생성

구현

진행 순서	
Model manual update	ml_headline server에서 model 생성되면 Today server와 HTTP 통신 하여 Today server의 ml_headline 모델을 manual update 한다.
Cluster의 headline 추출	1. 다중 문서 요약된 내용이 queue에 q.put()하면, ml_headline은 queue에서 q.get()하여 다중 문서 요약 content를 가져와 ml_headline_model()에 input 값으로 넣어 output 값으로 headline을 도출한다. 2. 다중 문서 요약된 content와 headline을 DB[Cluster table의 headline과 content를 저장한다]에 접근하여 json형태로 전송한다. 이때, DB에 접근하기 위해서 Today server는 Linux server에 HTTP통신한다.

4.4.2 headline ml_headline_processing 구현



무단 복제/전제 금지(MUD)

무단 복제/전제 금지(MUD)

4.5 Hybrid Filtering을 이용한 뉴스 추천

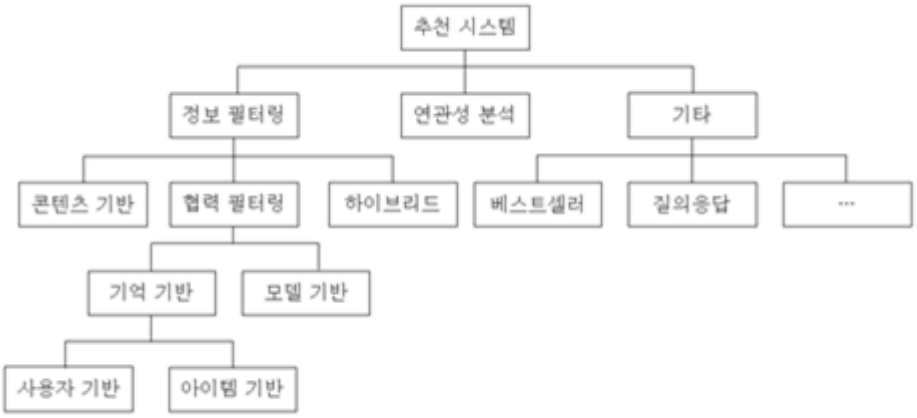
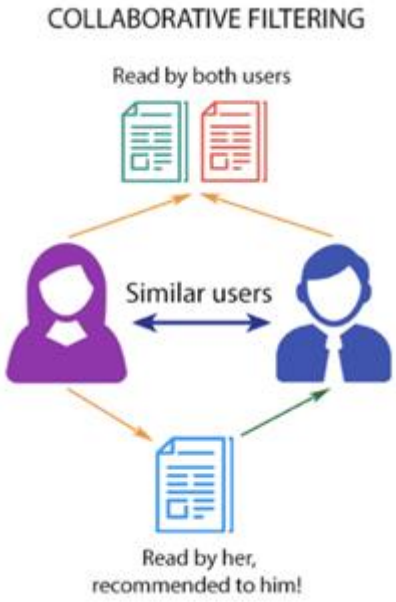


Figure 1. The categorization of recommender systems

추천이란 사용자가 아직 소비하지 않은 아이템 중 선호도가 높을 것으로 추정되는 아이템을 예측하는 것이다. NewSum에서는 사용자의 데이터를 기반으로 사용자가 아직 보지 않은 뉴스 중 선호도가 높을 만할 뉴스를 추천한다. 대표적인 추천 알고리즘으로는 Contents Based Filtering(콘텐츠 기반)과 Collaborative Filtering(협력 필터링)이 있으며 NewsSum은 각각의 장단점을 보완한 하이브리드 추천 시스템을 채택한다. 사용자 데이터가 부족한 초기에는 콘텐츠 기반 알고리즘을 사용하고 일정량의 사용자 데이터가 모이면 협업 필터링 알고리즘을 사용한다.

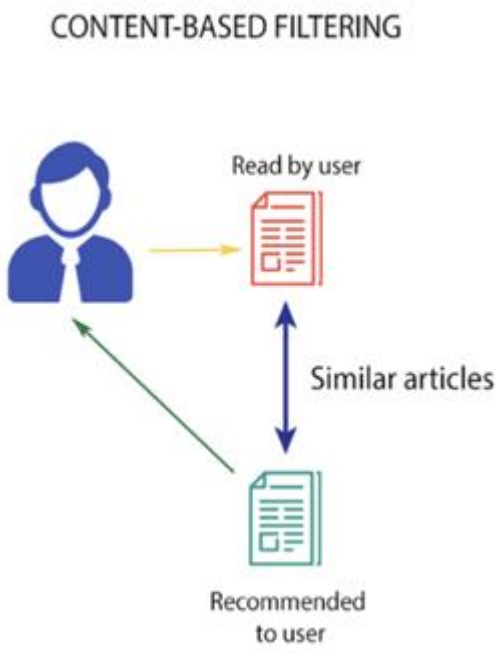
무단 복제/전제 금지(MUD)

4.5.1 Collaborative Flitering – User based



협업 필터링은 유사도를 기반으로 동작한다. 사용자간의 유사도를 기준으로 하는 경우는 사용자 기반(User-Based), 아이템 간의 유사도를 기준으로 하는 경우는 아이템 기반 (Item-Based) 추천이라 한다. 사용자 기반 알고리즘은 유사도를 구하여 선호 성향이 비슷한 사용자들을 그룹화하고 동일 그룹에 비슷한 선호 상품 추천한다. 이 때 두 사용자 간의 유사도는 두 벡터 간의 유사도로 정의할 수 있다. 두 벡터 간의 유사도를 구하기 위해서 다양한 방법이 사용될 수 있는데 그 중 피어슨 유사도를 사용하였다. NewSum은 아이템 기반 알고리즘을 채택하였으며 유사도 측정으로는 피어슨 상관계수를 사용하였다.

4.5.2 Contents Based Flitering – Item based



사용자의 선호도가 높은 아이템과 비슷한 아이템을 추천하기 때문에 아이템의 속성이 매우 중요하며 항목 자체를 분석해야 한다. 항목을 분석한 프로필과 사용자의 선호도를 추출한 프로필의 유사성을 계산한다. 콘텐츠 기반 알고리즘은 내용 자체를 분석하기 때문에 cf에서 발생하는 콜드스타트 문제를 해결 할 수 있다. 프로필을 생성하고 분석하는 데는 군집분석, 인공신경망, tf-idf등의 기술을 사용한다. 뉴스 기사를 추천 할 때는 tf-idf와 같은 기능을 이용하여 항목에 대한 점수를 추출하기 때문에 NewSum은 tf-idf 기술로 만든 행렬을 이용하여 코사인 유사도를 구하는 방식을 채택하였다.

무단 복제/전제 금지(MUD)

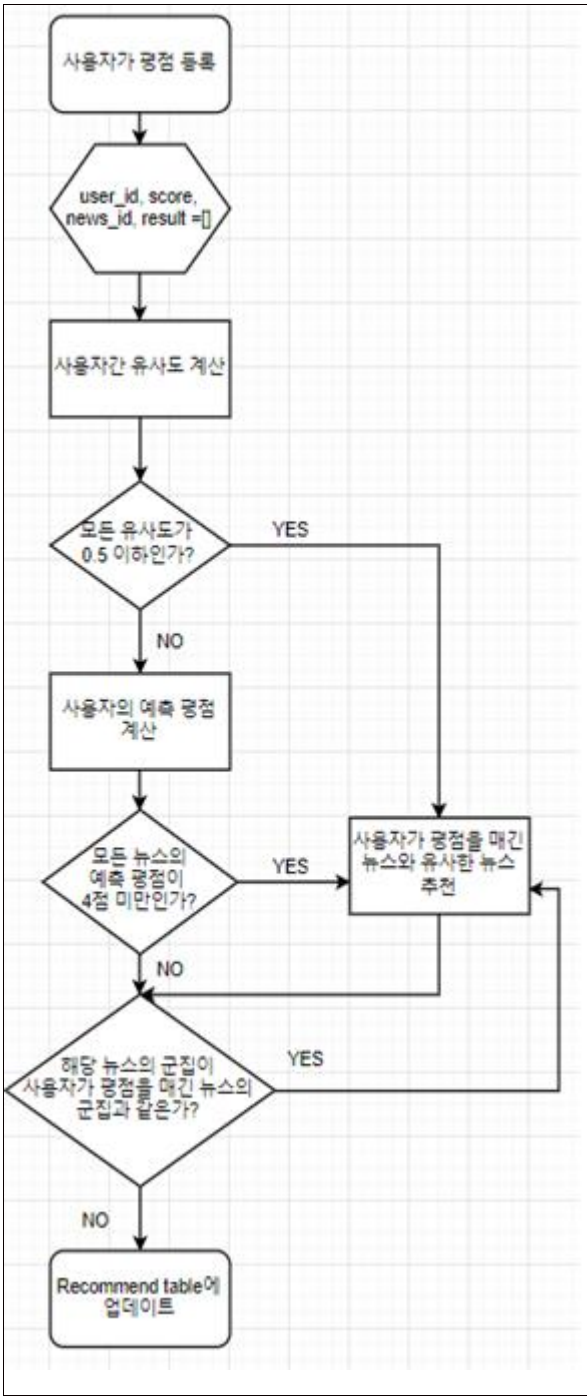
4.5.3 Hybird Filtering 추천 시스템 구현

초기의 사용자가 평가한 뉴스 기사 데이터가 없을 시 Collaborative filtering을 사용할 수 없기 때문에 사용자의 뉴스 기사 평점 데이터가 어느정도 쌓이기 전까지는 Content-based recommend 중 아이템 기반 추천 알고리즘을 사용하여 사용자가 평점을 매긴 뉴스와 유사한 뉴스 찾아 그 뉴스가 해당하는 Cluster와 그 Cluster 안의 뉴스 기사들을 보여준다.

사용자 데이터가 모인 이후, 협업 필터링 (Collaboration Filtering)중 사용자 기반 추천 알고리즘(User based)을 사용하며 개인 맞춤 뉴스 추천 기능을 제공한다. 초기에 평점에 대한 데이터가 전무한 사용자는 군집해서 묶인 개체가 많은 순으로 보여준다.

무단 복제/전제 금지(MUD)

최종보고서: Machine Learning을 이용한 뉴스 요약 및 추천 어플리케이션(Today News)



① 사용자에 맞는 뉴스 추천을 하기 위해 DB Rating Table에서 user_id, score와 news_id를 가져온다.

② 가져온 데이터를 이용해 추천 받을 사용자와 타 사용자 간의 피어슨 상관계수를 구해 0.5 이상인 사용자만을 리스트에 [사용자, 상관계수] 형식으로 저장한다.

③ 추천 받을 사용자가 평점을 매긴 뉴스를 제외하고 리스트에 있는 사용자들이 평점을 매긴 뉴스들에 대한 평점과 유사도를 곱해 모두 더하여 딕셔너리 형태로 score_dic[news_id] = news_id : "점수" 저장하고 그 뉴스를 본 사용자의 유사도도 모두 더해 딕셔너리 형태로 sim_dic[news_id] = news_id : "유사도" 형태로 저장한 후 score_dic[news_id] / sim_dic[news_id]를 통해 해당 뉴스에 대한 추천 받을 사용자의 예측 평점 값을 구하고 그 값이 4점 이상인 것만 result_list에 저장한다.

④result_list에 있는 뉴스들의 cluster_id가 DB Recommend Table에 있고 그에 해당하는 user_id가 추천 받을 사용자이면 result_list에서 삭제한다.

⑤ 만약 result_list가 비었다면 사용자가 본 뉴스 중 가장 높은 평점을 매긴 뉴스들을 이용해 그와 유사한 뉴스를 가져오고 result_list에 저장한 후 위와 같이 DB Recommend Table을 검사한다.

⑤ result_list에 남아있는 뉴스의 cluster_id를 DB Recommend Table에 user_id와 함께 저장한다.

무단 복제/전제 금지(MUD)

4.5.4 추천 시스템 구현 시 고려 사항

▌TF-IDF와 코사인 유사도

```
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=5, max_df=0.80, stop_words= stopword)
tfidf_matrix = tf.fit_transform(ds['content'])
```

[tf-idf]

```
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
for idx, row in ds.iterrows():
    similar_indices = cosine_similarities[idx].argsort()[:-100:-1]
    similar_items = [(cosine_similarities[idx][i], ds['id'][i]) for i in similar_indices]
    results[row['id']] = similar_items[1:]
```

[코사인 유사도]

TF-IDF로 만든 행렬로 코사인 유사도를 검사하여 뉴스 하나에 대해 유사한 정도가 높은 순으로 각 뉴스들이 대응 할 수 있게 리스트에 저장한다. 사용자가 해당 뉴스에 평점을 매기면 그 뉴스와 유사도가 높은 순으로 뉴스들의 Cluster를 검사해 평점을 매긴 뉴스와 Cluster가 같지 않으면 그 뉴스의 cluster_id를 recommend DB에 저장한다.

▌피어슨 상관계수

사용자간의 유사도 측정으로는 피어슨 상관계수를 사용한다. 두 사용자가 공통으로 평점을 매긴 뉴스의 점수를 사용해 측정한다. 유사도는 -1에서 1까지 표현 되며 0.5 이하로 나온다면 유사도가 없다고 판단하여 사용하지 않는다.

▌평점 예측

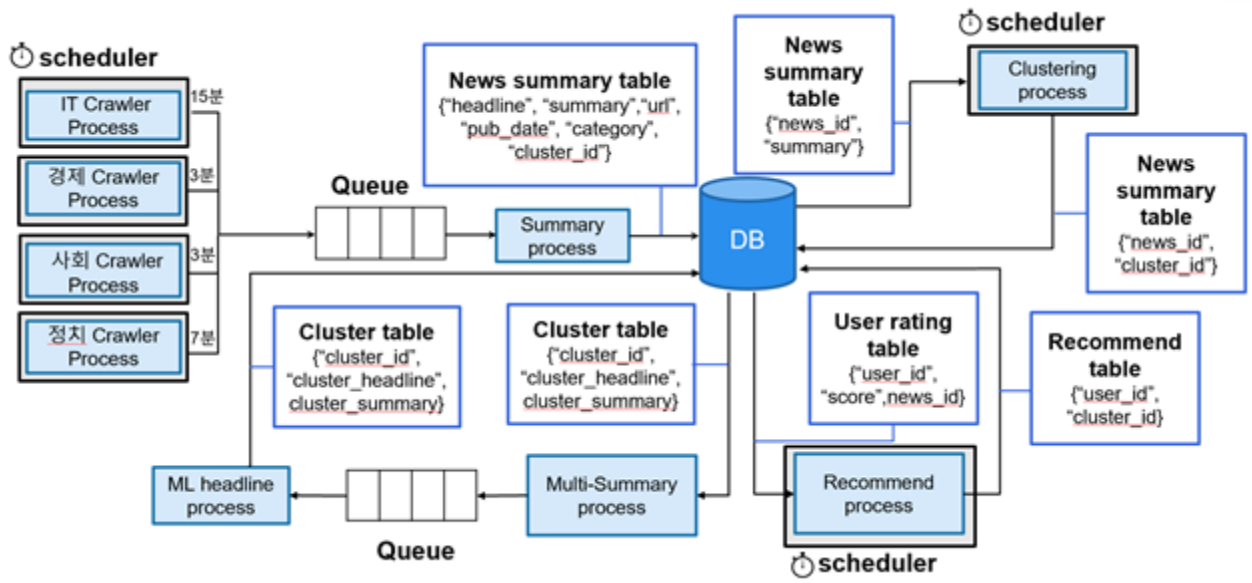
평점 예측은 피어슨 상관계수가 가장 높은 사람의 평점으로 계산 하는것보다 전체 사용자 중 유사도가 0.5 이상인 사람들의 평점들로 계산 하는것이 더 추천하는데 정확하다고 판단해 뉴스 추천 받을 사용자를 제외하고 그 사람과의 피어슨 상관계수가 0.5이상인 사람들의 뉴스평점과 유사도를 곱해 추측 평점을 구한 후 모두 더한 다음 유사도 총합을 나눠 나온 점수로 사용자의 평점을 예측한다.

▌뉴스 추천 주기

각 뉴스의 추천이 아닌 그 뉴스가 속한 클러스터를 추천해 주고 뉴스에 대한 평점 데이터가 일정 수준 쌓여야 하기 때문에 클러스터링 된 후 1 시간 뒤 실행하며 스케쥴러 모듈을 이용하여 1 시, 7 시, 13 시, 16 시, 19 시, 22 시에 추천 시스템을 사용한다.

무단 복제/전제 금지(MUD)

4.6 Scheduling과 Multiprocessing



NewSum은 사용자에게 뉴스를 실시간으로 제공하며 주기적으로 뉴스를 군집화하고 추천해 주기 위해 Scheduling 기능을 사용한다. 또한 News Summary와 News Crawling을 병렬처리, Headline Summary와 Contents Summary, News Clustering기능을 병렬처리 하기 위해 multiprocessing 기능을 사용한다.

스케줄러로는 APScheduler를 사용하며 이는 함수의 주기적 수행을 도와주는 라이브러리이다. Newsum은 일정 주기로 함수를 수행시키는 Interval 수행방식을 사용한다. 또한 스케줄 종류로는 다수 수행에 사용되는 BackgroundScheduler를 사용한다.

또한 멀티 프로세싱을 활용하여 시간이 오래 걸리는 크롤링등의 작업을 별도의 프로세스를 생성 한 후 병렬로 처리하여 성능을 높였다. Multiprocessing 패키지는 스레드 대신 서브 프로세스를 사용하여 파이썬의 GIL을 피할 수 있다.

무단 복제/전제 금지(MUD)

4.6.1 News Producing System에 적용

① News Crawler와 News Summary 간의 객체 교환이 있기 때문에 멀티 프로세싱의 Queue를 생성한다.

② 스케줄러를 생성하여 crawler를 제어한다. 카테고리 별로 총 4개의 프로세스를 생성한다. 스케줄러를 이용하여 일정 시간 간격으로 계속하여 crawler함수를 실행시킨다. 경제: 3분, 사회: 3분, 정치: 7분, IT:15분 간격으로 크롤링을 실행한다.

③ 크롤링 된 데이터는 큐에 넣어지고 Summary 프로세스가 그 데이터를 큐에서 꺼내와 처리한다. Crawler와 Summary프로세스는 병렬적으로 실행되어 처리속도가 빨라진다.

```
if "economy" in category:
    old.append(category["economy"])
    sched.add_job(Crawler.crawling, 'interval', seconds=180, id='test_1',args=["economy", q])  #
if "IT_science" in category:
    old.append(category["IT_science"])
    sched.add_job(Crawler.crawling, 'interval', seconds=900, id='test_2', args=["IT_science", q])
if "society" in category:
    old.append(category["society"],)
    sched.add_job(Crawler.crawling, 'interval', seconds=180, id='test_3', args=["society", q])  #
if "politics" in category:
    old.append(category["politics"])
    sched.add_job(Crawler.crawling, 'interval', seconds=420, id='test_4', args=["politics", q])
```

4.6.2 News Providing System에 적용

```
q1 = Queue()
q2 = Queue()
sched = BackgroundScheduler()

sched.start()
sched.add_job(random_four_news, 'cron', hour='7', id="s1", args=[df, cluster_id, q1])
sched.add_job(random_four_news, 'cron', hour='13', id="s2", args=[df, cluster_id, q1])
sched.add_job(random_four_news, 'cron', hour='16', id="s3", args=[df, cluster_id, q1])
sched.add_job(random_four_news, 'cron', hour='19', id="s4", args=[df, cluster_id, q1])
sched.add_job(random_four_news, 'cron', hour='22', id="s5", args=[df, cluster_id, q1])
sched.add_job(random_four_news, 'cron', hour='1', id="s6", args=[df, cluster_id, q1])


process_summary = Process(target=summary, args=(q1,q2,))
process_mlheadline = Process(target=ml_headline, args=(q2,"test",))
```

① 스케줄러를 생성하여 News Cluster를 제어한다. 카테고리 별로 총 4개의 프로세스를 생성한다. 스케줄러를 이용하여 일정 시간 간격으로 계속하여 Cluster함수를 실행시킨다.

② news clustering(군집화)가 완료된 후 1시간의 텀을 두고 스케줄러가 실행된다. 군집화된 뉴스들을 다중 요약(multi-summary)한다. 다중 요약과 ml_headline_processing을 병렬적으로 진행하기 위해서 둘 사이에 queue를 이용한다. 다중 요약된 cotents들은 queue에 put하고 ml_headline_processing처리할 때 queue 에서 다중 요약된 contents들을 get하여 처리한다.

4.6.3 News Recommend System에 적용

무단 복제/전제 금지(MUD)

```
sched.start()
for i in user:
    sched.add_job(Recommender.getRecommendation, 'cron', hour='7', minute'30', id=None, args=[i])
    sched.add_job(Recommender.getRecommendation, 'cron', hour='13', minute'30', id=None, args=[i])
    sched.add_job(Recommender.getRecommendation, 'cron', hour='16', minute'30', id=None, args=[i])
    sched.add_job(Recommender.getRecommendation, 'cron', hour='19', minute'30', id=None, args=[i])
    sched.add_job(Recommender.getRecommendation, 'cron', hour='22', minute'30', id=None, args=[i])
    sched.add_job(Recommender.getRecommendation, 'cron', hour='1',, minute'30' id=None, args=[i])
```

① 클러스터링된 이후 일정 시간이 지난 후 즉 1시간 반 이후로 정했다.

무단 복제/전제 금지(MUD)

무단 복제/전제 금지(MUD)