|  |
| --- |
| **최종 보고서**  DeepLearning을 이용한 뉴스 요약 서비스  **Newsum Application** |

Ver. 3.0

2019.12.21

한국외국어대학교 정보통신공학과

**Team name: MUD(Meeting Using Deeplearning)**

**문서 정보**

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **소속** | **성 명** | **날 짜** |
| **작성자** | **한국외국어대학교** | **박주영(T)** | **2019.12. 22** |
| **한국외국어대학교** | **김아연** | **2019.12. 22** |
| **한국외국어대학교** | **김혜원** | **2019.12. 22** |
| **한국외국어대학교** | **홍승환** | **2019.12. 21** |
| **한국외국어대학교** | **이산가 비두샤** | **2019.12. 21** |
| **검토자** | **한국외국어대학교** | **박주영(T)** | **2019.12. 22** |
|  |  |  |
|  |  |  |
| **사용자** |  |  |  |
|  |  |  |
|  |  |  |
| **승인자** | **한국외국어대학교** | **홍진표** | **2019.12.23** |

**머리말**

본 문서는 묶음 뉴스 요약 본 제공 서비스인 Newsum Application시스템의 요구사항정의서에 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.

**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** |
| **검토자** | **박주영** | | |
| **3.0** | **김아연** | **2019.12.22** | | **3차 보완** |
| **김혜원** | **2019.12.22** | | **3차 보완** |
| **검토자** | **박주영** | | |

**목 차**

[**문서 정보** 2](#_Toc28007906)

[**머리말** 3](#_Toc28007907)

[**개정 이력** 4](#_Toc28007908)

[**목 차** 5](#_Toc28007909)

[**1. 개요** 8](#_Toc28007910)

[**1.1 목적** 8](#_Toc28007911)

[**1.2 참고 문서** 8](#_Toc28007912)

[**2. 서비스 개요** 9](#_Toc28007913)

[**2.1 서비스 기획 배경** 10](#_Toc28007914)

[**2.2 서비스 기능 소개** 11](#_Toc28007915)

[**3. 시스템 설명** 13](#_Toc28007916)

[**3.1 전체 시스템 구성** 13](#_Toc28007917)

[**3.2 Application Server** 15](#_Toc28007918)

[**3.2.1 Django REST Framework &　REST API** 15](#_Toc28007919)

[**3.2.2 REST API 구성** 15](#_Toc28007920)

[**3.3 Newsum Server** 16](#_Toc28007921)

[**News Produce System** 16](#_Toc28007922)

[**News Cluster System** 16](#_Toc28007923)

[**Headline & Multi-summary System** 16](#_Toc28007924)

[**News Recommend System** 16](#_Toc28007925)

[**3.4 ML\_Headline PC** 17](#_Toc28007926)

[**3.5 Database 구성** 18](#_Toc28007927)

[**3.3.1 Database 설계도(MySQL)** 18](#_Toc28007928)

[**3.3.2 Database 구성 표** 19](#_Toc28007929)

[**3.3.3 Database 저장 형태** 21](#_Toc28007930)

[**4. 주요 기술 설명 및 적용 방법** 23](#_Toc28007931)

[**4.1 Crawling** 24](#_Toc28007932)

[**4.1.1 Newsum Crawler 구현** 25](#_Toc28007933)

[**4.2 Lexrankr을 이용한 뉴스 요약** 27](#_Toc28007934)

[**4.2.1 News Summary 구현 (1차 요약)** 28](#_Toc28007935)

[**4.2.2 News Multi-Summary 구현 (2차 요약)** 29](#_Toc28007936)

[**4.3 K-means Clustering을 이용한 News Clustering** 31](#_Toc28007937)

[**4.3.1 News deleting 기능에 대한 상세 설계** 32](#_Toc28007938)

[**4.3.2 전처리 과정** 32](#_Toc28007939)

[**4.3.3 최적의 K값찾기** 34](#_Toc28007940)

[**4.3.4 근접 기사 찾기** 35](#_Toc28007941)

[**4.4 Attention Mechanism을 이용한 뉴스 헤드라인 재생성** 36](#_Toc28007942)

[**4.4.1 Headline Summary Model 생성 구현** 37](#_Toc28007943)

[**4.4.2 ML\_Headline 동작 관련에 대한 설계** 40](#_Toc28007944)

[**4.5 Hybrid Filtering을 이용한 뉴스 추천** 41](#_Toc28007945)

[**4.5.1 Collaborative Flitering – User based** 42](#_Toc28007946)

[**4.5.2 Contents Based Flitering – Item based** 42](#_Toc28007947)

[**4.5.3 Hybird Filtering 추천 시스템 구현** 43](#_Toc28007948)

[**4.5.4 추천 시스템 구현 시 고려 사항** 45](#_Toc28007949)

[**4.6 Scheduling과 Multiprocessing** 46](#_Toc28007950)

[**4.6.1 News Producing System에 적용** 47](#_Toc28007951)

[**4.6.2 News Providing System에 적용** 48](#_Toc28007952)

[**4.6.3 News Recommend System에 적용** 49](#_Toc28007953)

[**5. NewSum 기능 설명** 50](#_Toc28007954)

[**5.1 인터페이스 및 기능 설명** 50](#_Toc28007955)

[**5.1.1 로그인/회원가입 기능** 50](#_Toc28007956)

[**5.1.2 북마크 기능** 51](#_Toc28007957)

[**5.1.3 홈 메인 화면** 52](#_Toc28007958)

[**5.1.4 카테고리 화면** 53](#_Toc28007959)

[**5.1.5 군집 뉴스 보여주기** 54](#_Toc28007960)

[**5.1.6 개별 뉴스 보여주기** 55](#_Toc28007961)

[**5.1.7 평점 기능** 56](#_Toc28007962)

[**5.1.8 TTS 기능** 57](#_Toc28007963)

[**5.2 Sequence Diagram** 58](#_Toc28007964)

[**5.2.1 News Provide System Sequence Diagram** 58](#_Toc28007965)

[**5.2.2 News Cluster System Sequence Diagram** 58](#_Toc28007966)

[**5.2.3 Headline & Multi-summary System Sequence Diagram** 60](#_Toc28007967)

[**5.2.4 News Recommend System Sequence Diagram** 60](#_Toc28007968)

[**5.3 APP flow chart** 61](#_Toc28007969)

[**6. 적용방안 및 기대효과** 62](#_Toc28007970)

[**7. 프로젝트 세부 추친 일정** 63](#_Toc28007971)

[**8. 팀원 담당 업무** 64](#_Toc28007972)

**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, 참고 문서를 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

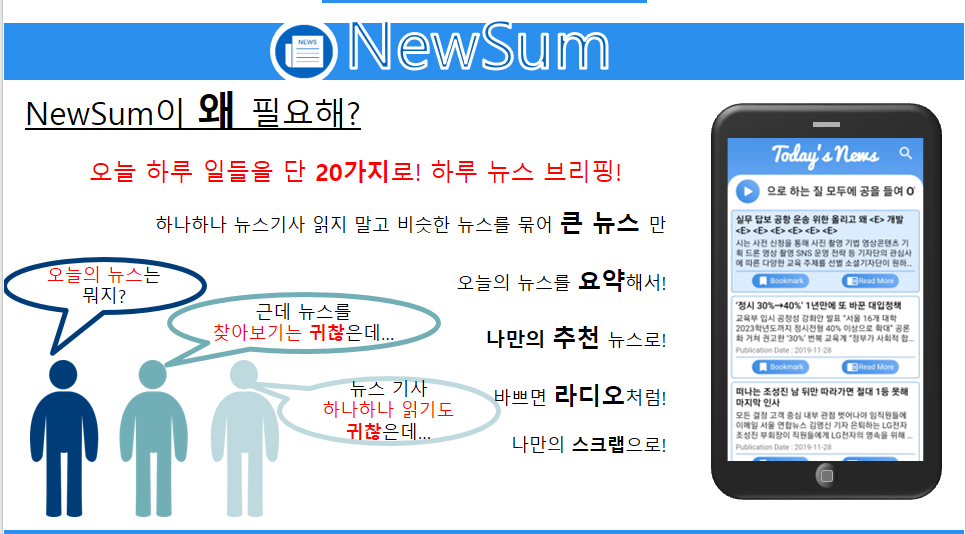
▶ Lexrank 알고리즘(추출적 요약)과 Attention mechanism RNN 모델(추상적 요약)을 이용한 본문 요약과 헤드라인 추출

▶ 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 |

# **2. 서비스 개요**

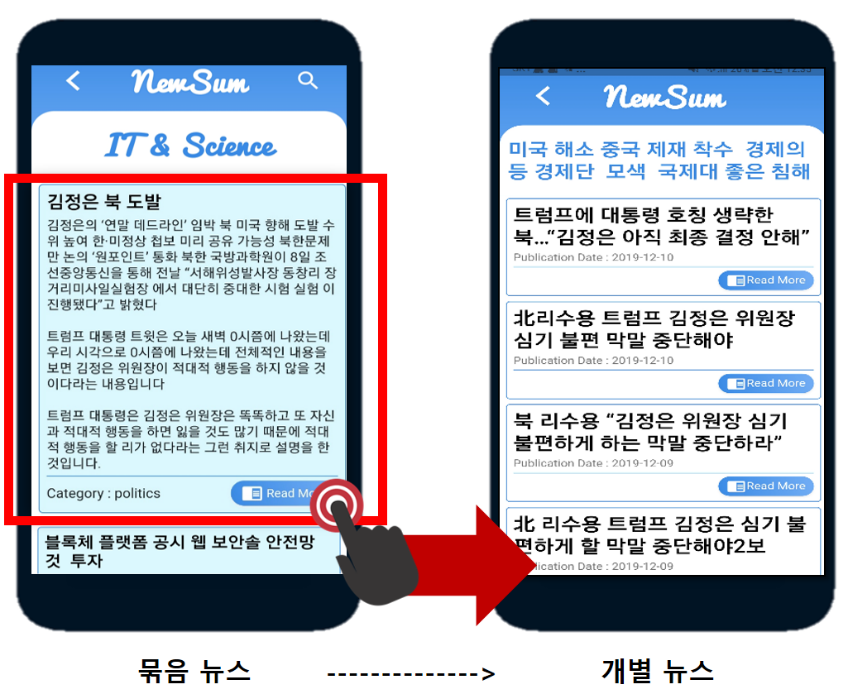
Today News 서비스는 뉴스를 보고 싶어 하나, 뉴스를 찾아보거나 뉴스 기사 하나하나를 읽어보기 귀찮아하는 사람들을 위해 기존 뉴스들보다 접근성 있고, 시각적으로 한눈에 오늘의 뉴스를 알려주는 어플리케이션이다. 즉, 뉴스를 봐야겠다 생각은 하나, 뉴스를 보지 않는 사람들에게 오늘 하루의 뉴스를 단 몇가지의 묶음 뉴스로 간략하게 브리핑해주는 어플리케이션이다.

[Figure 1] Newsum 서비스 개요

## **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 Newsum 어플리케이션을 기획하고 구현하게 되었다.

## **2.2 서비스 기능 소개**

본 장에서는 Newsum이 제공하는 주요 기능들을 간단하게 소개하고 세부 설명은 ‘3장 기능 설명’에서 이어 한다. 해당 서비스 기능에 대한 사용자 관점에서의 사용법은 ‘5장 사용자 인터페이스 및 APP 가이드’에서 명시한다.

**(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. 그림에서 기사 내용을 보여주는 빨간 네모와 노란 네모를 보면 3줄만에 내용이 끝남을 볼 수 있다.

**(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. .그림에서 빨간 네모를 보면 주제별 뉴스 즉, 묶음 뉴스가 제공되며 Read more버튼을 누르면 해당 주제의 개별 뉴스 또한 제공된다.

**(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. 그림에서 노란부분에서 색깔 이모티콘이 있는 버튼에 평점을 매기면 이를 토대로 노란 네모처럼 개인화 추천순으로 뉴스가 사용자에게 제공된다. (2)와 마찬가지로 추천순으로 보여주는 뉴스 또한 묶음 뉴스로 제공된다.

**(4) 북마크 기능**

사용자는 나중에 다시 읽고 싶은 기사를 북마크(스크랩)기능을 통해 저장해 둘 수 있다. 스크랩한 뉴스는 프로필창에서 확인 할 수 있다. 그림에서 초록 네모를 보면 각 개별 뉴스마다 북마크 버튼을 누르면 프로필 창의 북마크 카테고리에서 스크랩 된 뉴스들을 확인할 수 있다.

**(5) 뉴스 브리핑 기능**

홈 피드 상단에 스피커 아이콘을 누르면 오늘 뉴스의 주제들을 브리핑 기능을 통하여 음성으로 들을 수 있다. 핑크 네모를 보면 play 버튼을 누르면 오늘의 추천 뉴스를 재생 할 수 있다.

# **3. 시스템 설명**

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

## **3.1 전체 시스템 구성**

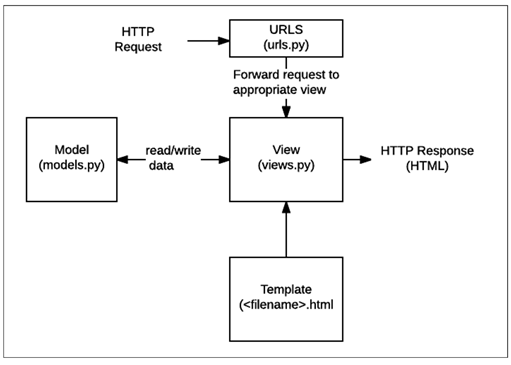
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NewSum System 구성요소** | | | | |
| **user** | **App(android or iOS)** | **Application Server** | **Database** | **ML\_Headline PC** |
| user는 NewSum System Service를 받는 대상이다. | Application Server에서 운용되는 application을 사용할 수 있는 역할을 담당한다. | application server로 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. |

|  |  |  |
| --- | --- | --- |
| **NewSum server** | | |
| News producing system | News providing System | News recommending System |
| >News produce system은 실  시간으로 Naver 뉴스 crawling (사회, 정치, IT/과학, 경제 카테고리)한 후, 본문 요약하여 DB에 저장하는 시스템이 다. | > News clustering으로 뉴스 본문을 가지고 뉴스들을 군집 화한다. 군집 화된 뉴스들은 DB에 저장된다.  > 각 군집 화된 뉴스모음에서 다중문서요약이 실행되고, 다중문서 요약된 본문을 이용해 새로운 head line title을 추출한다. 여기서 생성된 다중 문서 요약본과 title은 DB에 저장된다. | > News recommending Syste  m은 사용자에게 개인 맞춤으  로 뉴스를 추천해주는 서비스  시스템이다. 각 사용자가 본 뉴스 data양에 따라 contents-based rec ommendation과 collaborative reco mmendation 둘 중 하나의 알고리 즘을 채택하여 사용된다. |

[Figure 2] NewSum 시스템 구성도

## **3.2 Application Server**

**3.2.1 Django REST Framework &　REST API**

Django는 python을 활용해 쉽게 웹 서비스를 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.

## **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** |
| * + 1. **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. |
| * + 1. **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. |
| * + 1. **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. |
| * + 1. **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. |

## **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.

## **3.5 Database 구성**

**3.3.1 Database 설계도(MySQL)**

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

**3.3.2 Database 구성 표**

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

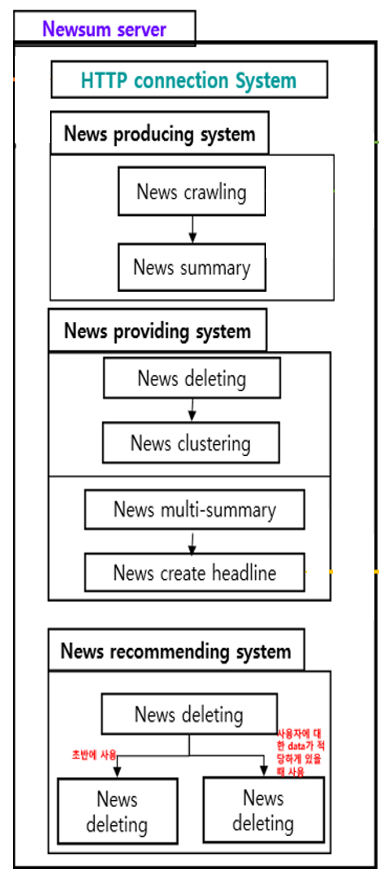
|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute** | |
| **articles** | News\_id(PK) | Crawling한 뉴스 고유의 uuid4부여 |
| Headline | Crawling한 뉴스의 제목 |
| Summary | Crawling한 뉴스의 본문을 lexrankr 알고리즘을 이용하여 본문 요약 content |
| url | Crawling한 뉴스의 URL주소 |
| Pub\_date | Crawling한 뉴스의 생성 날짜[뉴스기사에 포함 되어있는 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 input값으로 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 |
| 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 |

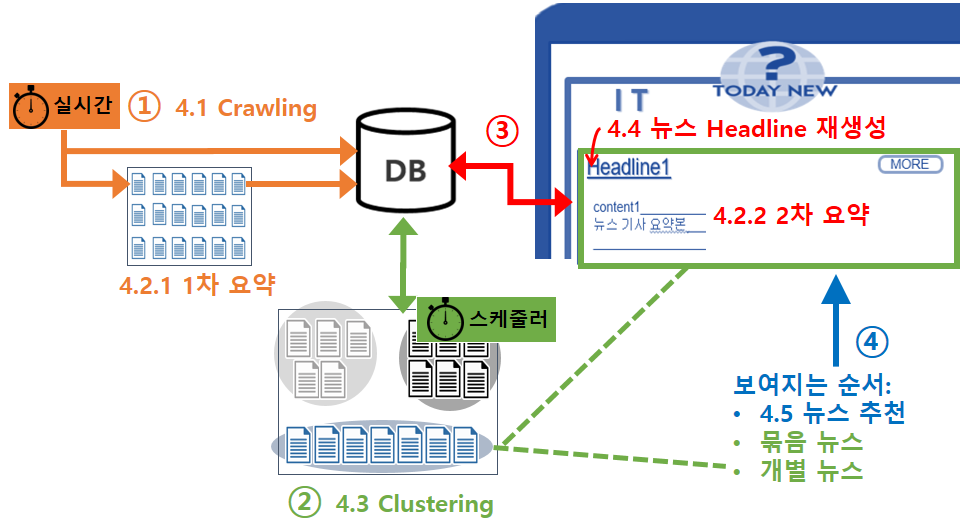
models.py에서 table과 field들의 attribute를 정의하고 python manage.py migrate를 통해 Database에 table을 생성한다.

**3.3.3 Database 저장 형태**

|  |  |
| --- | --- |
| **Table Name** | **저장 형태** |
| newssummary |  |
| Users |  |
| Cluster |  |
| UserRating |  |
| Recommend |  |

# **4. 주요 기술 설명 및 적용 방법**

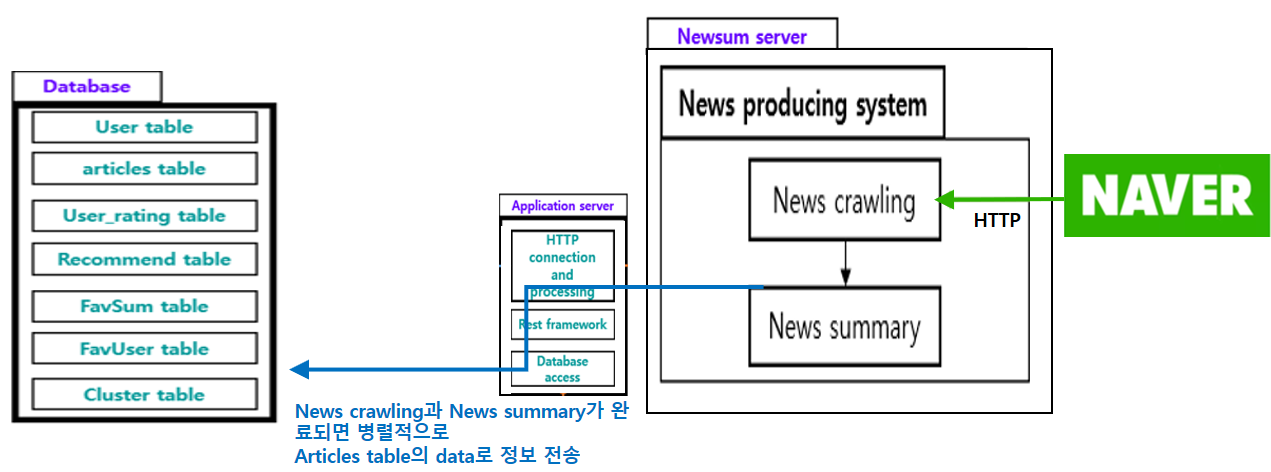
본 장에서는 2.2장 소개했던 Newsum APP의 서비스 기능을 구현하는데 사용한 기술과 적용 방법에 대한 구체적으로 설명한다. 3.3장에서 설명한 Newsum Server의 System들에 어떤 방식으로 기술을 적용하였으며 어떤 순서로 작동하는지에 대해 기술하였다.



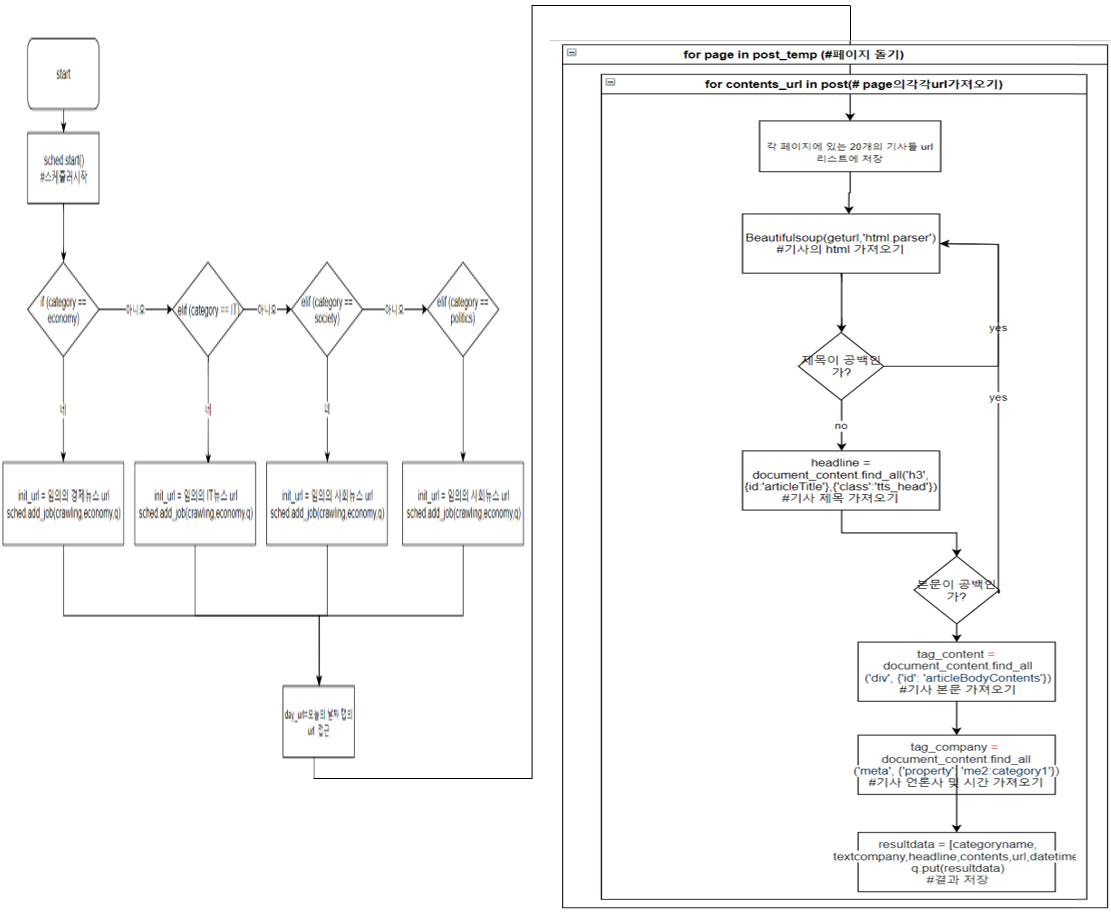
위의 좌측 구성도의 Newsum server 내부 System들이 우측 구성도의 번호순으로 작동하게 되며 최종적으로 사용자에게 오른쪽 APP그림의 일부와 같이, 혹은 2.2 서비스 기능의 사진과 같은 형태로 제공된다. 그림 구성도의 번호 속 명시된 기능들은 앞으로 이 장에서 기술하는 기능들이며 각 기능 번호는 이 장의 목차 순서와 일치한다. 또한 4.6에 대한 내용은 그림의 타이머로 대체했다.

## **4.1 Crawling**

크롤링이란 컴퓨터 소프트웨어 기술로 웹 사이트에서 원하는 정보를 추출하는 것을 의미하며, 크롤러는 인터넷의 웹 페이지를 방문해서 수많은 자료들을 수집하는 일을 하는 프로그램을 말한다. 사람이 모든 웹사이트를 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 클래스, News crawling 기능으로 정의된다. News crawling의 구성도는 위 그림과 같으며 이 기능을 통해 최종적으로 Database에 Naver 뉴스 속보가 실시간으로 저장된다. 아래의 4.1.1장에서 Newsum Crawler 구현 방법에 대해 자세히 기술한다.

**4.1.1 Newsum Crawler 구현**

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.



1. 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
2. 이때 크롤링된 내용은 resultdata = [news title, news contents, category, date, URL, publication]로 리스트 형식으로 저장되며 이 값은 q.put(resultdata)로 큐에 저장된다.

## **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 4.3에서 기술하는 그래프 기반 클러스터링 알고리즘을 적용한다.

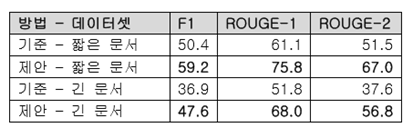
최종적으로 NewSum 요약 기능에서는 추출적 요약 기법인 Lexrank 알고리즘을 이용하여 기사 원문에서 3줄 요약문을 만들어 Database에 저장한다. (설문조사에 의하면 모바일에서 이용자가 좋은 뉴스라고 생각하는 뉴스의 길이는 3줄-5줄이다.) Newsum은 Lexrankr을 이용하여 2번의 요약을 한다. 각각의 요약을 News Summary(1차 요약), News multi-Summary(2차 요약)이라 부른다.

▶ 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.



* + 1. 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 문서 요약도 진행하였다.
    2. **4.2.1 News Summary 구현 (1차 요약)**



Newsum server에서 가장 첫번째로 작동되는 작업으로 News producing System이 있다. 이 시스템은 앞 절에서 말한 News crawling 기능과 News summary 기능이 병렬적으로 실시간으로 계속 구동된다.

이 때 News summary 기능에서는 4.1의 News crawling이 되면 기사 각각의 내용을 Lexrank를 이용하여 요약해주게 된다. 병렬 처리에 관한 내용은 4.6 Multiprocessing에서 다룬다.

News summary가 완료되면 바로 하나의 기사 data를 하나씩 Application server를 거쳐 Database에 저장한다. 아래 표는 이에 해당되는 구현 과정을 기술한다.

|  |  |
| --- | --- |
| **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. |

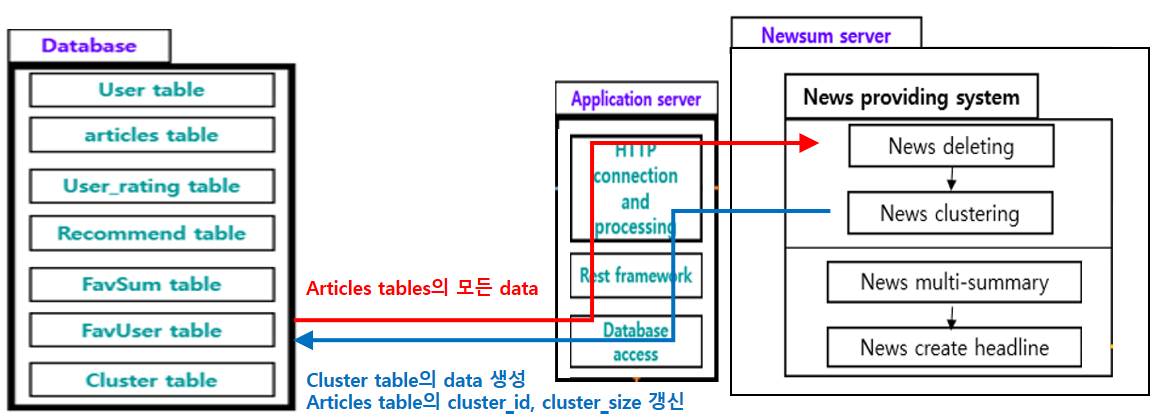
* + 1. **4.2.2 News Multi-Summary 구현 (2차 요약)**

|  |
| --- |
| **다중 뉴스 요약 구현 과정** |
| 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    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.    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)    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) |

## **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 label이 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 구하는 것이 많은 시간이 걸린다는 단점이 있다. NewSum에서는 최적의 k를 구하는 시간을 줄여 성능을 높이는 법을 실험하고 적용해보았다.

Newsum은 24시간의 실시간 뉴스를 군집 뉴스로 브리핑 해주는 서비스로 4.1의 News crawling 기능과 4.2의 1차 요약, News summary 기능으로 실시간으로 계속 Database에 뉴스기사 data가 저장된다. 이때 News providing System에서 자체적으로 지정한 update 시간이 되면 News deleting 기능과 News Clustering 기능이 이루어진다. (update 시간에 대한 병렬 처리는 3.6 스케줄링에서 다룬다) Clustering이 완료되면 이후 4.4와 4.2의 2차 요약, News multi-summary 기능이 시작된다.



Update 시간이 되면 News providing System은 Database에서 Dataset을 불러와 News deleting 기능에서 24시간 분량의 Data만 남기고 모두 삭제한다.

삭제가 완료된 즉시 News clustering 기능이 작동하여 24시간 분량의 Dataset을 전처리, 최적의 변수를 통한 K means module의 적용시킨다. 이때 자체적으로 구현한 클래스 K\_class.py를 사용하여 최적의 시간으로 적절한 성능을 가지는 Clustering을 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.

**4.3.2 전처리 과정**

News deleting이 모두 완료되면 순차적으로 News clustering이 시작된다. News clustering은 크게 3.4.2 전처리 과정, 3.4.3최적의 K값 찾기, 3.4.4 근접 기사 찾기, 3.4.5 DB저장으로 설계했다.

요구사항에서 명시한 것과 같이 군집화(Clustering)는 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

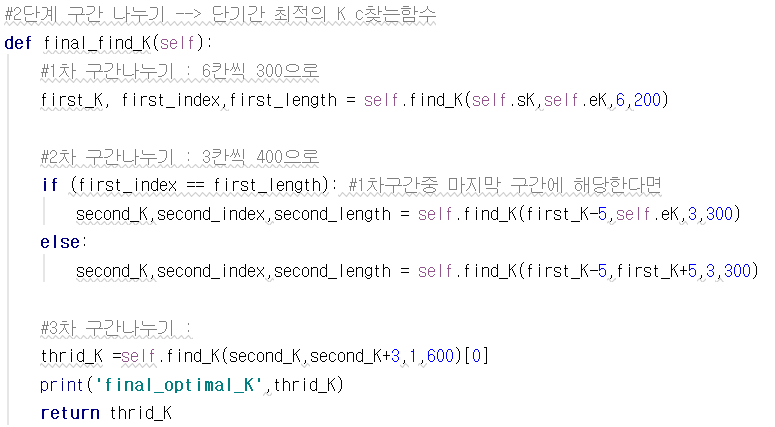
prorecessing class에 있다

|  |
| --- |
| ( Equation 1 [News clustering\_전처리 과정 1]) |
| Equation 1 tokenizer함수에서 ①형태소 분석 & 토큰화 처리 ②처리 불용어 제거 과정을 다룬다  Konlpy module의 twitter(현재 Okt() 이름이 바뀜)로 한국어 형태소 분석을 하고 pose =[“Noun”]으로 토큰화 처리를 한다. 또한 stopword로 특수문자 및 영어 숫자를 제거하여 불용어 제거를 한다.   |  | | --- | | (Equation 2 [News clustering\_전처리 과정 2]) | | Equation 2 result 함수에서 앞서 말한 tokenizer()함수를 불러 데이터를 한번 가공하고 이후 CounterVectorizer라는 Tfidfvectorizer 모듈로 벡터화를 한 후 정규화 처리를 해준다. 이 결과로 X라는 전처리 과정을 마친 Dataset으로 바꿔진다. | |

**4.3.3 최적의 K값찾기**

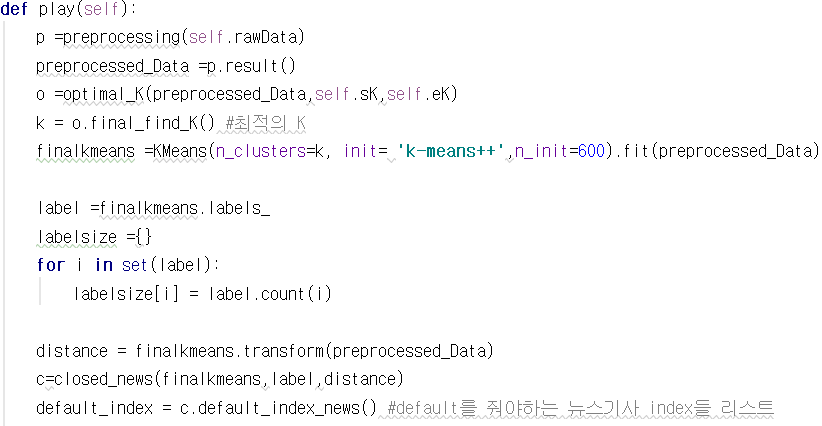
clustering에서 활용하는 Kmeans module의 최대 단점은 K값 즉, cluster의 개수를 정해줘야만 하는 것이다. 이때 이 K값에 따라 정확성이 달라지므로 최적의 K를 빠른 시간 내에 찾는 알고리즘을 설계하였다.

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.

(적절할 k 찾기 - 2단계 구간 나누기)

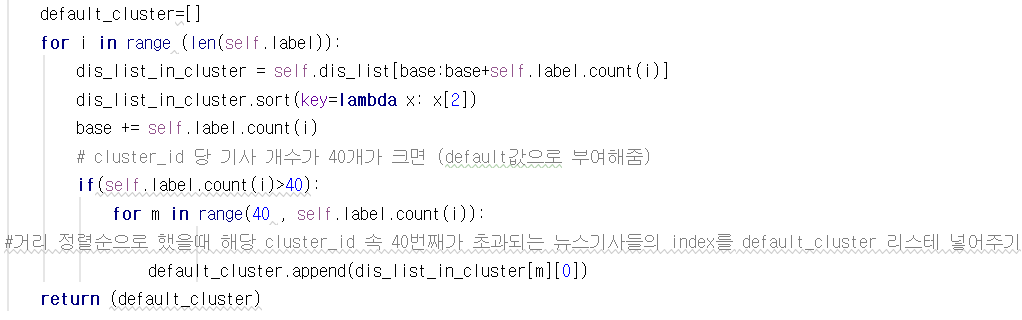
여기서 문제가 되었던 정확도란 Kmeans module의 자체 인자 값인 n\_init을 통해 변화를 줄 수 있다. n\_init이란 Kmeans module이 자체적으로 dataset 내의 임의의 점을 시작으로 지정해준 K만큼 군집을 만들기 시작하는데 이대 임의의 점들의 정해주는 횟수를 말한다. 즉 n\_init의 값이 높을수록 랜덤으로 주는 시작점이 많아지고 즉 반복학습이 많아지므로 더 적절한 군집들로 수렴한다. 하지만 반복학습이 많아짐에 따라 또한 속도 역시 느려지는 단점이 있다. 따라서 이를 해결하기 위해 Newsum의 News clustering에서는 ‘구간 별 정확도 나누기’ 방법으로 시간은 단축시키되 성능은 좋게 유지할 수 있는 elow 기법 최적의 K를 찾는 optimal\_K class를 구현하였다.

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.

**4.3.4 근접 기사 찾기**

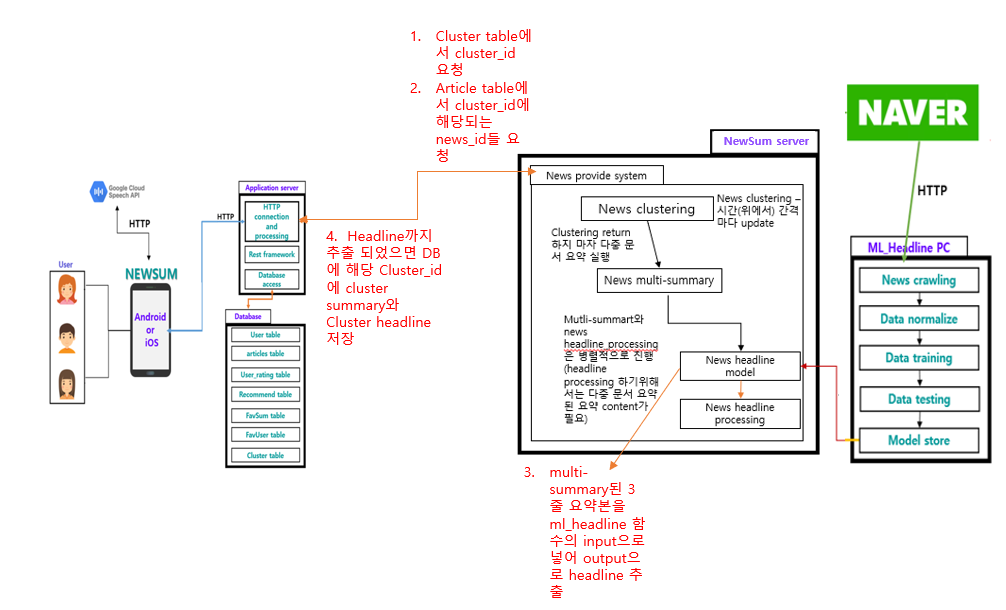
3.4.3까지의 과정을 통해 Database에서 가져온 24시간 Dataset을 preprocessing class를 통해 전처리 과정 , optimal\_K class를 통해 최적의 K값까지 찾았다. 이를 main class 인 run\_Kmeans class의 play함수에서 실행 시킨 후 가공처리 된 Dataset인 X와 최적의 K를 kmeans module을 통해 만든 finalkmeans인 kmeans model의 인자로 넣는다. 이의 결과로 finalkmeans은 적절한 K값으로 전처리 한 X를 clustering 한 모델을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

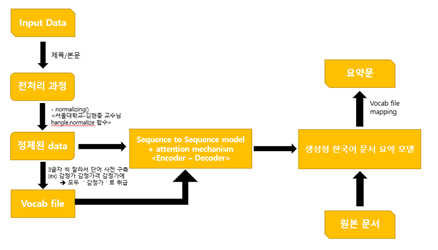


이때 근접 기사란 요구사항에서 명시한 것과 같이 클러스터 내부 화면에서 보여지는 개별 뉴스를 의미 한다. 따라서 한 개의 클러스터에서 존재하는 개별 기사는 최대 40개로, centroid와 가까운 순으로 순서를 매겨 40번째 까지의 뉴스data만 근접 기사로 생성된 cluster id를 부여하고, 나머지 값들은 모두 default cluster id를 주어 카테고리 화면에서 보여지는 개별 뉴스로 보여진다.

## **4.4 Attention Mechanism을 이용한 뉴스 헤드라인 재생성**

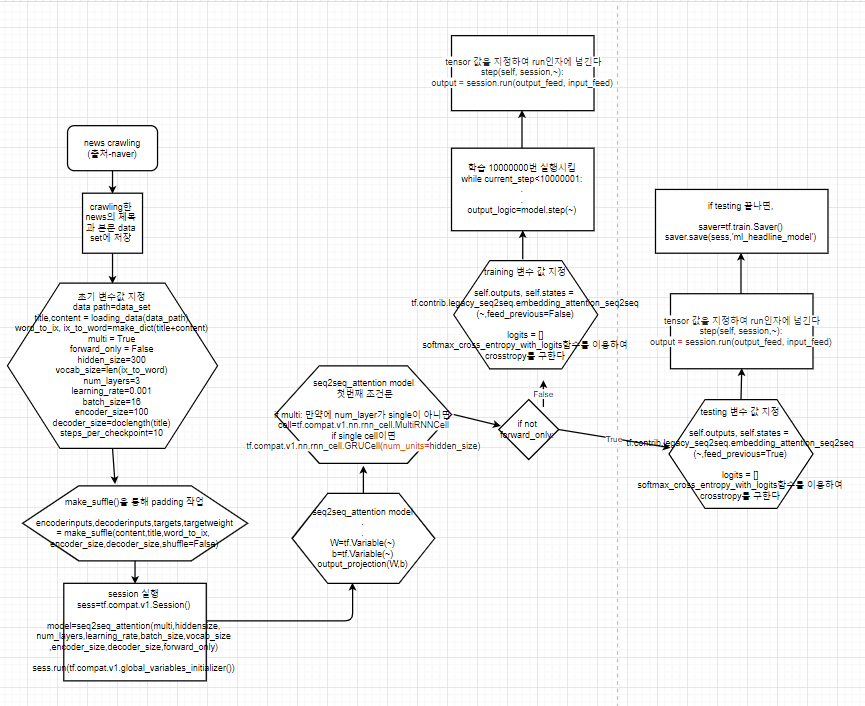


Headline과 summary를 설명하기 앞서, 앞 3.4와 3.5에서 news clustering(군집화)가 완료되면 군집화된 뉴스들을 다중 요약(multi-summary)한다. 다중 요약과 ml\_headline\_processing을 병렬적으로 진행하기 위해서 둘 사이에 queue를 이용한다. 다중 요약된 cotents들은 queue에 put하고 ml\_headline\_processing처리할 때 queue 에서 다중 요약된 contents들을 get하여 처리한다.



[간략한 ML\_headline model 생성과정]

**4.4.1 Headline Summary Model 생성 구현**



**▶ 위 알고리즘의 진행 순서**

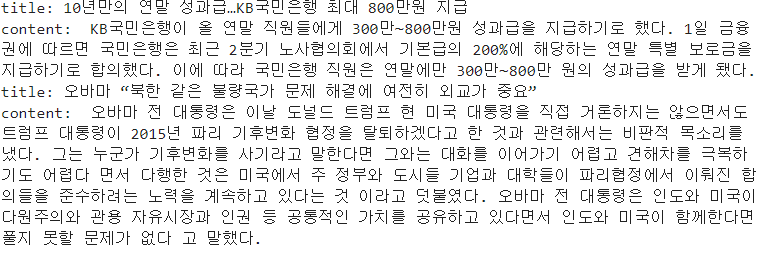
① ml\_headline server에서 news crawling을 통하여 dataset을 얻는다

- news crawling의 기간은 한달로 늘려 dataset을 넉넉히 얻는다[얻는 정보는 title과 content이다]

- 이때, 사용하는 news 기사 출처는 naver 뉴스이다[카테고리-정치,사회.IT/과학,경제]

- 다중 문서 요약의 content는 3문장이기 때문에, 맞춰서 dataset을 구축한다(즉, crawling한 content의

모든 뉴스는 lexrank 알고리즘으로 3문장으로 요약되어 있다)



[문서 요약되어 있는 content 예시]

② 충분한 Dataset을 얻은 후, main.py에서 주어진 data\_path에서 title과 content를 뽑아 normalize한다.

- 이때 사용되는 normalize함수는 서울대 김현중 박사님의 hangle.normalize 함수를 사용하였다.



- normalize함수에서 영어와 숫자를 무시하면 뉴스에서는 손해가 크기 때문에 영어와 숫자까지 정규화

한다.

③ normalize된 title과 content를 make\_dict()함수를 통해 정수 인코딩한다[return 값으로 word->index, index->word 딕셔너리 얻음]



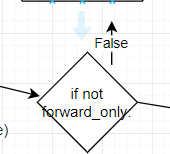
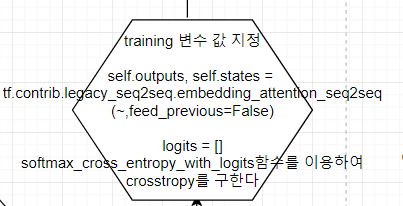
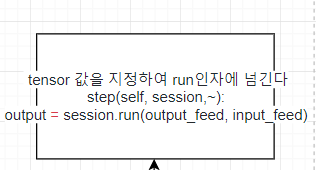
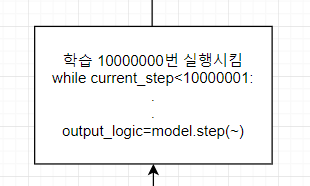
- 이때 단어들은 maxlength(3글자)로 끊어서 사전을 구축한다

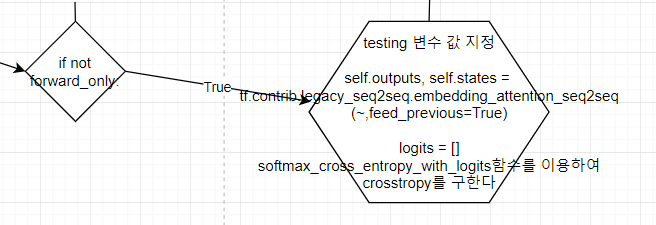
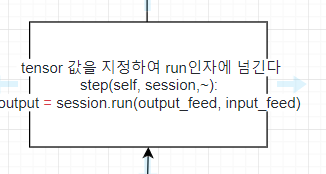
(ex) 감정가, 감정가격에, 감정가에 🡪 모두 ‘감정가’로 취급한다.

④ seq2seq model을 train과 test하기위한 초기값 변수 설정 후 padding작업까지 완료한다.

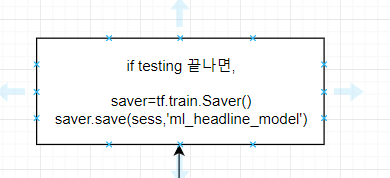
- content와 title의 길이보다 data sequence 길이가 짧으면 <PAD>를 넣어 길이를 맞춘다.

⑤ session 시작 후, step<100000될 때까지 training시킨다.

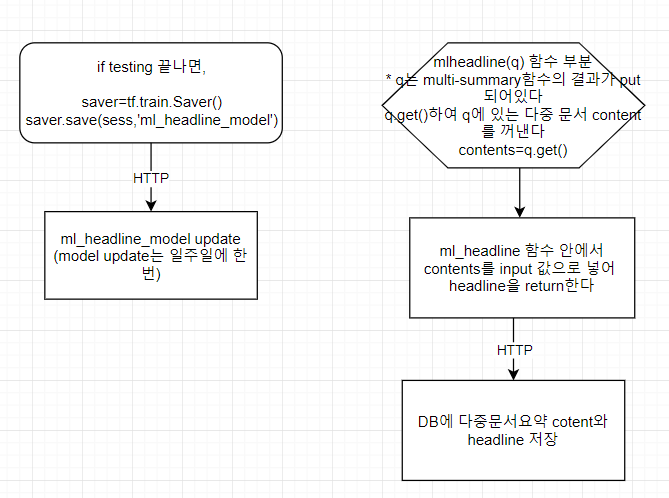


⑥ training 끝나면 testing 시작한다.

⑦ training까지 마치면 model 저장한다.

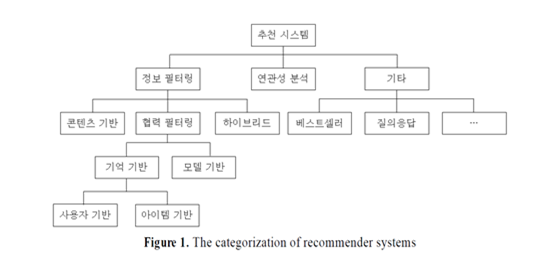


**4.4.2 ML\_Headline 동작 관련에 대한 설계**



|  |  |
| --- | --- |
| **진행 순서** | |
| **Model manual update** | ml\_headline server에서 model 생성되면 Today server와 HTTP통신 하여 Today server의 ml\_headline 모델을 manual update한다. |
| **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. |

## **4.5 Hybrid Filtering을 이용한 뉴스 추천**

 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.

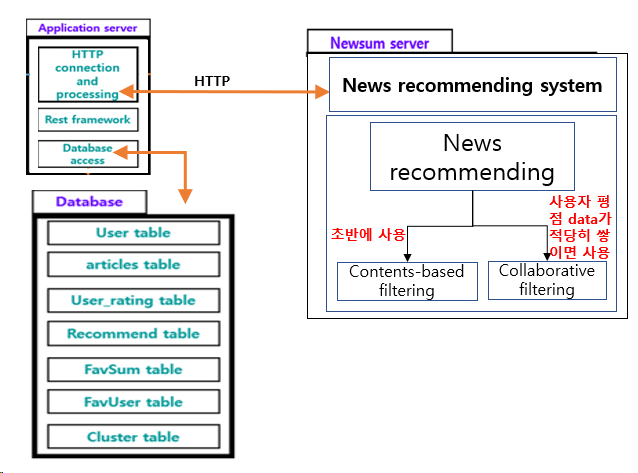
* + 1. **4.5.1 Collaborative Flitering – User based**

|  |  |
| --- | --- |
|  | * + 1. 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. |

**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.

|  |  |
| --- | --- |
|  | * + 1. 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.   * + 1. ④result\_list에 있는 뉴스들의 cluster\_id가 DB Recommend Table에 있고 그에 해당하는 user\_id가 추천 받을 사용자이면 result\_list에서 삭제한다.     2. 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.     3. ⑤ result\_list에 남아있는 뉴스의 cluster\_id를 DB Recommend Table에 user\_id와 함께 저장한다. |

**4.5.4 추천 시스템 구현 시 고려 사항**

**1.** **TF-IDF와 코사인 유사도**

|  |
| --- |
|  |

[tf-idf]

|  |
| --- |
|  |

[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.

**2.** **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.

**3.** **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.

**4.** **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 1시간 뒤 실행하며 스케쥴러 모듈을 이용하여 1시, 7시, 13시, 16시, 19시, 22시에 추천 시스템을 사용한다.

## **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.

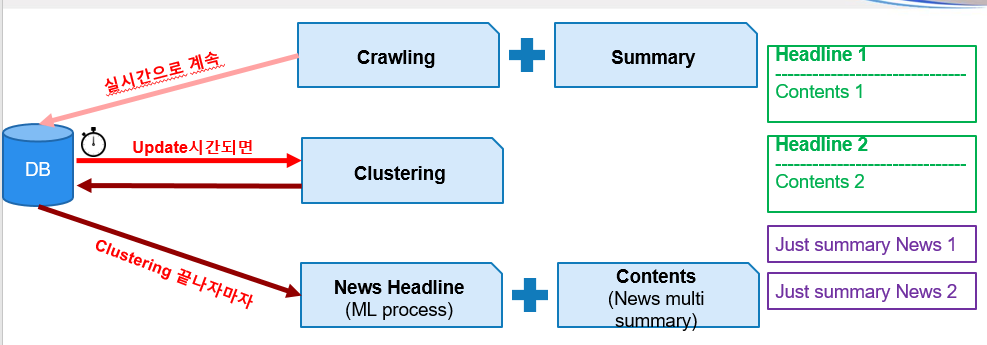
**4.6.1 News Producing System에 적용**

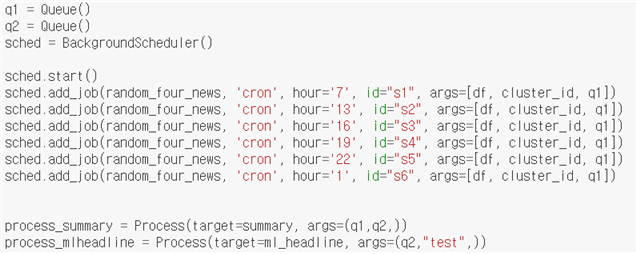
* + 1. ① News Crawler와 News Summary 간의 객체 교환이 있기 때문에 멀티 프로세싱의 Queue를 생성한다.

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.

**4.6.2 News Providing System에 적용**





* + 1. 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**

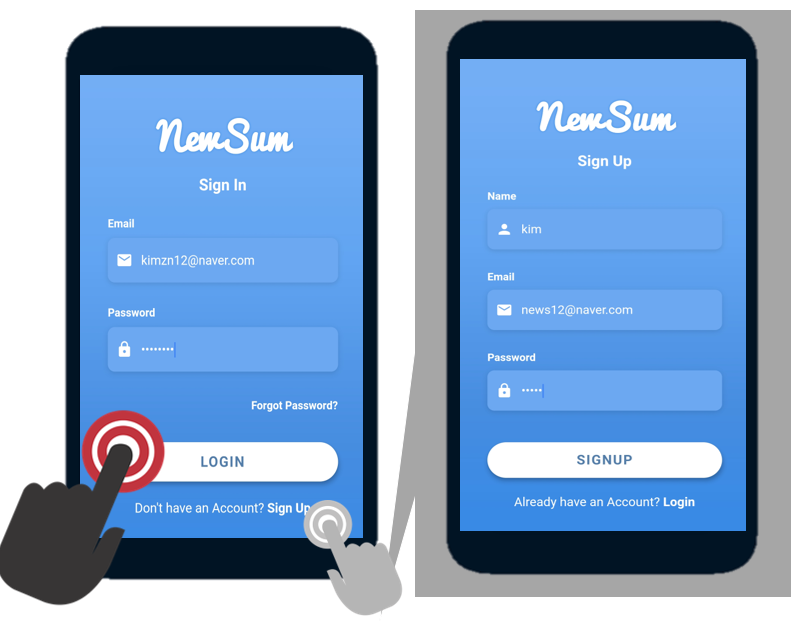


1. 클러스터링된 이후 일정 시간이 지난 후 즉 1시간 반 이후로 정해 각 사용자 마다 추천이 될 수 있도록 구현 했다.
2. getRecommendation 함수를 호출 하면 먼저 collaborative filtering을 통해 추천할 뉴스를 가져오고 없다면 contents-based filtering으로 넘어가도록 구현 했다.

# **5. NewSum 기능 설명**

## **5.1 인터페이스 및 기능 설명**

* + 1. **5.1.1 로그인/회원가입 기능**



**1. 회원가입 기능**

* + 1. ① NewSum은 로그인을 한 사용자에 한하여 서비스를 제공하기 때문에 사용자는 서비스를 이용하기 위해 먼저 회원가입을 해야한다. 첫 화면(로그인 창)의 Sign up버튼을 누르면 우측 그림과 같이 회원가입 창으로 넘어간다.
    2. ② 사용자가 Newsum APP에서 Email과 Password와 Name을 입력하고 Sign up 버튼을 누르면 data를 Firebase로 보낸다.
    3. ③ Firebase에서 유효ID를 Database ORM을 전송하여 DB를 만들고 데이터를 저장한다. 회원가입을 성공하면 다시 로그인 창을 보여준다.
    4. **(2) 로그인 기능(자동)**
    5. ① 사용자가 Newsum APP에서 Email과 Password를 입력하고 로그인 누르면 사용자의 data는 Firebase로 보내진다. Firebase를 통해 유효한 Email과 password인지 검사한다. 유효한 정보라면 로그인 성공된다.
    6. ② 로그인에 성공하면 회원의 정보를 가져와 홈 메인화면을 보여준다.
    7. ③ 한 번 로그인에 성공 하면 정보는 local DB에 저장되므로 다음에 앱을 실행 시킬 시 자동 로그인 된다.

**5.1.2 북마크 기능**

|  |  |
| --- | --- |
|  | ▶ 사용자는 개별 뉴스의 Readmore 화면에서 Bookmark 표시를 하여 뉴스 스크랩을 할 수 있다.  ▶ 사용자는 과거 스크랩한 뉴스들을 Bookmark 화면에 가서 볼 수 있다. 이때 사용자는 스크랩한 뉴스들의 제목과 내용을 제공 받을 수 있다. |

① 사용자가 개별 뉴스의 Readmore 화면에서 뉴스 북마크 표시를 하게 되면, 사용자 ID와 스크랩한 뉴스에 대한 ID값을 보내면 해당 뉴스 ID에 해당되는 제목과 요약 내용을 전달한다. 전달 받은 총 3가지의 정보를 FavSum table에 저장하고, User\_Id와 뉴스 ID를 FavUser에 저장한다.

* + 1. ② 사용자가 북마크 화면을 클릭하면, 사용자 ID를 보내어 해당 사용자의 과거 스크랩 뉴스정보들을 FavUser table과 FavSum table에서 전달받아 화면으로 제공받는다.

**5.1.3 홈 메인 화면**



① 홈 화면에서는 카테고리에 상관없이 종합적으로 뉴스가 보여진다. 뉴스들은 주제별로 묶여있으며, 선택 시 3줄로 요약된 뉴스들을 볼 수 있다. 상단의 스피커 버튼을 누르면 뉴스 브리핑을 들을 수 있다.

* + 1. ② 홈화면을 클릭하면 사용자는 먼저 사용자 개별 추천순, 군집 뉴스(묶음 뉴스)의 크기순으로 뉴스를 제공받을 수 있다. 이때 개별 추천순은 사용자 별 추천 뉴스를 의미하고, 군집 뉴스의 크기는 현재 이슈되고 있는 정도를 측정하여 이슈의 랭킹을 의미한다.
    2. ③ 사용자가 APP에서 홈화면을 클릭하면 Application server로 User\_id와 홈화면 상태를 전송하고 Application server가 이를 Database로 전달한다.

④ Database는 정보를 전달받은 후, Recommend table과 Cluster table의 cluster\_size을 통해 뉴스의 순서와 해당 뉴스들의 news\_id를 통해 articles table에서 정보를 가져와 Application server에 전송하여 APP 화면에 사용자의 추천 뉴스와 군집 뉴스가 보여진다.

**5.1.4 카테고리 화면**

① 총 4가지 카테고리(Society, Economy, IT, Politics)중 선택 가능하며 각 카테고리 별로 뉴스를 보여준다. 뉴스는 일괄적인 것이 아닌 사용자마다 추천 뉴스를 보여준다.

② 카테고리 선택화면에서 특정 카테고리를 클릭하면 사용자는 해당 카테고리에 대한 군집 뉴스와 개별 뉴스를 제공받을 수 있다.

③ 사용자가 APP의 카테고리 선택 화면에서 Society, Economy, IT & Science, Politics 카테고리 중 원하는 카테고리를 선택하면 Application server로 해당 카테고리 상태가 전달된다.

④ Application server는 Database로 정보를 전달하고 Database는 Cluster table 내에서 해당 카테고리에 대한 군집뉴스 정보와 개별 뉴스 정보를 Application server에게 전송한다.

⑤ Application server는 받은 군집 뉴스 정보를 APP화면에 보여준다.

**5.1.5 군집 뉴스 보여주기**



① 군집 뉴스란 Newsum 시스템을 통해 새롭게 가공된 뉴스로, 유사한 내용을 가진 여러개의 개별 뉴스를 묶어 새로 생성한 뉴스이다. Database의 Cluster table의 cluster\_headline과 cluster\_summary가 군집 뉴스의 제목과 내용이 된다.

② 각 군집 뉴스 마다 ‘Read more’버튼을 가지며 Read more 버튼을 클릭하면 클러스터 내부 화면에서 해당 cluster의 개별 뉴스들을 볼 수 있다. 이때 이 개별 뉴스들이 보여지는 순서를 cluster\_size가 결정한다.

③ Application server가 사용자의 요청에 따라 해당되는 상태를 Database로 전달 해 주면 Database의 Cluster table에서 cluster size를 통해 군집 뉴스가 나열되는 순서에 대한 data를, Cluster table에서 FK인 news\_id를 통해 해당 cluster 내 개별 뉴스들의 뉴스 정보 data를 news\_id가 PK로 있는articles table에서 불러와 Application server에게 전송한다.

④Application server에서 전달 받은 data를 통해 군집 뉴스를 articles table의 뉴스 정보로, 특정 군집 뉴스의 ‘Read more’버튼을 클릭한 클러스터 내부화면에서 cluster\_size 순서로 개별 뉴스를 APP 화면에 보여준다.

**5.1.6 개별 뉴스 보여주기**



① 개별 뉴스를 보여주는 상태는 카테고리 화면에서 보여주는 개별 뉴스(첫번째 그림)와 군집 뉴스의 ‘Read more’ 클릭하여 클러스터 내부 화면(두번째 그림)에서 보여주는 개별 뉴스가 있다. 하지만 흔히 말하는 개별 뉴스는 카테고리 화면에서 보여주는 개별 뉴스를 의미한다.

② Application server가 사용자의 요청에 따라 해당되는 카테고리 상태 혹은 클러스터 내부 화면 상태를 Database로 전달 해 주면, Database의 articles table에서 FK인 cluster\_id가 default인 뉴스들의 headline, summary, category, pub\_date, sum\_date, url를 Application server에게 전송한다.

③ Application server에서 전달 받은 data를 통해 카테고리 화면 상태에서는 pub\_date, headline, summary의 일부를, 클러스터 내부 화면에서는 pub\_date, headline만 APP 화면에 보여준다.

④ 이때 개별 뉴스 보기에서도 ‘Read More’버튼이 있다.

버튼 클릭 시 ReadMore화면이 보여지고 여기서는 해당 개별 뉴스의 headline, summary, category, pub\_date, sum\_date를 보여주고 북마크 버튼, 평점 버튼, Article 버튼이 존재한다. 이때 Article 버튼을 클릭하면 해당 개별 뉴스의 url 페이지가 보여지게 된다.

**5.1.7 평점 기능**



① 사용자는 개별 뉴스마다 평점을 부여할 수 있다. 1-5점까지의 평점을 매길 수 있다.

② Application server가 사용자가 부여한 평점 데이터를 Database로 전달 해 주면, Database의 UserRating table의 score에 평점이 저장된다. 이 정보는 후에 추천 시스템에 사용된다.

**5.1.8 TTS 기능**

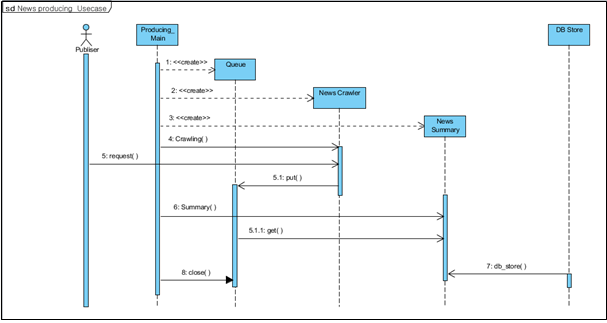


① 사용자가 APP의 홈화면에서 플레이 버튼을 클릭하면 Google TTS API로 Data[홈화면 내의 추천 뉴스]를 전송한다.

② Google TTS API는 Streaming 방식으로 APP으로 Data의 음성을 전달하여 사용자는 추천 뉴스를 라디오처럼 들을 수 있다.

**5.2 Sequence Diagram**

* + 1. **5.2.1 News Provide System Sequence Diagram**



* + 1. **5.2.2 News Cluster System Sequence Diagram**

|  |
| --- |
|  |

* + 1. **5.2.3 Headline & Multi-summary System Sequence Diagram**

|  |
| --- |
|  |

* + 2. **5.2.4 News Recommend System Sequence Diagram**

|  |
| --- |
|  |

## **5.3 APP flow chart**



# **6. 적용방안 및 기대효과**

**(1) 빠른 시간 내 뉴스 편집 및 제공 가능**

오늘날 생성되는 정보량이 많아지면서 다양한 정보의 뉴스가 범람하게 되었다. 이로 인해 방대한 양의 무분별한 정보가 난무하여 사람들은 정보를 수용하는데 혼란을 겪고 있다. Today News는 인공지능을 기반으로 한 자동화 뉴스 제공 시스템으로 사람의 개입 없이 많은 양의 뉴스를 신속하고 정확하게 편집 (중복 제거 및 요약, 주제별 군집화)하여 사용자에게 제공한다.

**(2) 주제별로 다양한 뉴스 기사를 제공하여 사용자의 뉴스 수용 및 활용 능력의 향상을 도움**

뉴스 전체 내용이 아닌 요약된 뉴스들을 주제별로 보여주며 중복된 뉴스를 제거하고 비슷한 주제를 가진 뉴스들을 클러스터링 해주어 사용자가 같은 주제를 가진 다양한 뉴스를 한 번에 볼 수 있게 한다. 주제의 헤드라인을 추출하여 제공하기 때문에 현재 뉴스 정보의 흐름을 빠르게 파악하는 것을 도우며 사용자가 한 주제에 하나의 뉴스만 접하는 것이 아닌 여러 뉴스를 접하는 것을 도와 주기 때문에 뉴스 수용 및 활용 능력을 높여줄 것으로 기대된다.

**(3) 개인화 맞춤 추천 서비스로 사용자 니즈 충족 및 콘텐츠 소비량의 증가**

Today News는 사용자에게 맞춤형 경험을 제공하고자 사용자의 관심사를 고려한 뉴스 추천 시스템을 구현하였다. 사용자의 뉴스 선호도를 분석하여 추천 뉴스가 일괄적인 것이 아니라 로그인 한 회원마다 달리 나타나 사용자에게 맞춤형 경험을 제공할 수 있을 것으로 보인다. 사용자는 충족시킬 수 있어 콘텐츠 소비량이 증가할 것이다. 개인화 경험을 제공함으로써 사용자의 만족도를 높여 콘텐츠 소비량이 증가할 것으로 보인다.

# **7. 프로젝트 세부 추친 일정**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **프로젝트 기간** | **2019.9.9~2019.12.9** | | | | | | | | | | | | |
| **개발 내용** | **프로젝트 기간** | | | | | | | | | | | | |
| **1주** | **2주** | **3주** | **4주** | **5주** | **6주** | **7주** | **8주** | **9주** | **10주** | **11주** | **12주** | **13주** |
| **아이디어 회의 및 계획** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **프로젝트 관련 자료 조사** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **역할 분담 및 핵심 기술 조사** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **사업 제안서 작성** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **요구 사항 정의서 작성** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **서버 및 데이터 베이스 구현** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **앱 구현** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **뉴스 크롤링** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **클러스터링 시스템구현** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **요약 시스템 구성(추상적)** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **요약 시스템 구성(추출적)** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **추천 시스템 구현** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **개별 테스트 및 보안** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **통합 테스트 및 보안** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **최종 발표 및 시연** |  |  |  |  |  |  |  |  |  |  |  |  |  |

# **8. 팀원 담당 업무**

