

프로젝트명 : 키워드 기반 경제 콘텐츠 큐레이션
캡스톤 디자인Ⅱ, 중간보고서

Version 1.0

개발 팀원 명(팀리더) : 송재민

박재필

이용경

대표 연락처 : 010-5118-4832

e-mail : 20181622@edu.hanbat.ac.kr

캡스톤 디자인 II 중간보고서 내용

1. 요구사항 정의서에 명시된 기능에 대하여 현재까지 진척된 결과 및 그 내용을 기술하시오.

1- 1. 키워드 분석

- 분석

고정 키워드는 보다 나은 신뢰성을 위해 뉴스 기반 통계 검색 이용
(<http://data.kostat.go.kr/social/keyword/index.do>)

일일 키워드는 직접 구해내는 방식

네이버 뉴스 경제 카테고리를 기반으로, 기사를 분석.

- 설계

기사별로 제목과 기사를 합친 것을 문장단위로 자르고,

Komoran 형태소 분석기로 그 문장을 명사단위로 만들어 준다.

(Komoran 형태소 분석기로 추출되지 않는 명사는 사용자 사전 사용)

```
# Komoran 형태소 분석기 사용자 사전 추가
komoran = Komoran(userdic='user_dictionary.txt')

nouns = [] # 추출할 명사 저장할 리스트
# 각 문장들에 대하여 명사를 추출. 허나, 불용어 없고 단어의 길이는 1 초과여야합니다. (한 단어가 키워드일리는 없기때문에)
for sentence in sentences:
    nouns.append(' '.join([noun for noun in komoran.nouns(str(sentence))
                           if noun not in stopwords and len(noun) > 1]))

return nouns
```

1	카카오뱅크	NNP
2	국토교통부	NNP
3	LK-99	NNP
4	메리츠회재	NNP
5	신호희망타운	NNP
6	오일쇼크	NNP
7	모건스탠리	NNP
8	롯데글로벌로지스	NNP
9	케이뱅크	NNP
10	코로나19	NNP
11	L6디스플레이	NNP
12	LNG	NNP
13	OTT	NNP
14	전기차	NNP
15	비구이위안	NNP
16	특례보급자리를	NNP
17	DSR	NNP
18	준법감시위원회	NNP
19	영광	NNP
20	KT	NNP
21	L6에너지솔루션	NNP
22	산업기술보호법	NNP
23	CSM	NNP
24	클리아렐	NNP
25	캔서문샷	NNP
26	SUV	NNP

그 후, TextRank 알고리즘을 통해 명사들의 중요도를 계산한 후,
기사별로 상위 중요도 높은 10개의 키워드를 추출한다.

$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

어제의 전체 경제 기사를 대상으로, 전체 기사 중,
가장 중요도 높게 많이 나온 10개를 일일키워드로 선정하게 된다.
(고정 키워드와 겹치게 될 시, 그 다음 순위의 키워드를 일일키워드로 가져오게 된다.)
상위 30개의 데이터는 워드클라우드 데이터로 사용한다.

```

===== 고정 키워드 크롤링을 시작합니다 =====
- 고정 키워드
['국유재산', '수익증권', '재판매', 'IOC', '뉴미디어', '세액공제', 'e스포츠', '로그인', '프린트', '다큐멘터리']
===== 고정 키워드 크롤링을 마칩니다. =====

- 어제의 일일 키워드
['대출', '서울', '투자', '아파트', '금리', '미국', '중국', 'gs건설', '처분', '영업정지']

- 워드 클라우드 사용 데이터
[['대출', 39], ['서울', 31], ['투자', 26], ['아파트', 26], ['금리', 25], ['미국', 23], ['중국', 20], ['gs건설', 19], ['처분', 19], ['영업정지', 18],

```

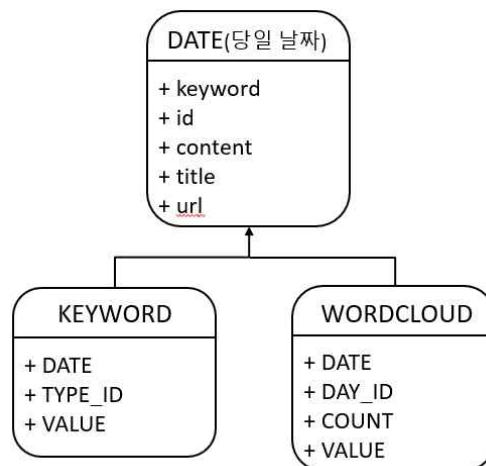
1- 2. 크롤링 및 전처리

- 분석

아래 목차에서 시스템 개요는 키워드 기반 콘텐츠 큐레이션의 대략적인 순서도이다. 뉴스기사에서 추출할 수 있는 각각의 키워드에 대한 데이터를 정제되지 않은 많은 양의 데이터들을 추출한다. 추출된 데이터들에서 불필요한 문장, 단어, 기호들을 제거하는 전처리 과정을 거쳐 정형화된 데이터로 전처리하고 DB에 저장한다. 키워드들의 전처리된 데이터에서 교집합이 일어나는 즉, 서로 관련된 데이터들만 선별하고, 데이터들을 군집화를 통해 주제별로 분류하고 분류된 글들 중 키워드와 관련된 내용을 NLP모델을 통해 콘텐츠 큐레이션을 수행해 다양한 형태의 콘텐츠를 생성하는 것을 목표로 하고 있다.

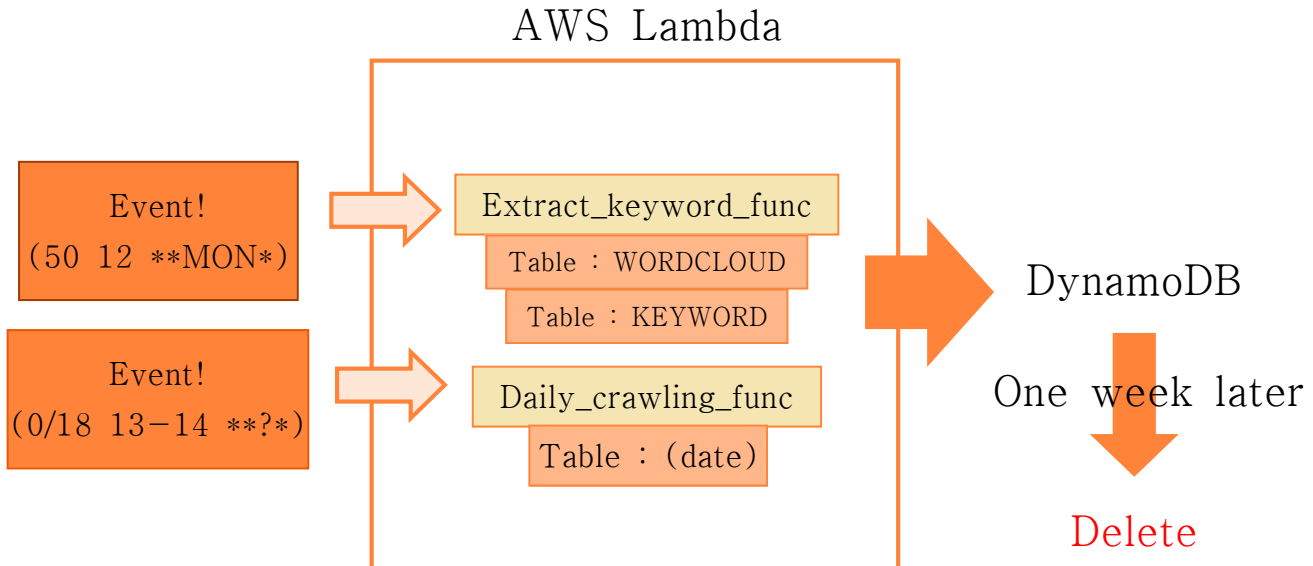
- 설계

DB 설계 모델



No-SQL DB를 사용하기 때문에 스키마 등의 제약을 받지 않아 자유롭게 데이터 저장이 가능하며 KEYWORD 테이블에는 일주일마다 변경되는 고정키워드, 그리고 어제의 일일 키워드들을 저장해두고, WORDCLOUD 테이블에 날짜마다 키워드 분석 파트에서 가장 높게 나타난 상위 30개의 키워드를 갯수와 함께 저장해준다. 자동 크롤링 시 두 테이블에서 키워드를 읽어들이어 크롤링을 실시하여 DATE 테이블에 뉴스 기사를 저장한다.

자동 크롤링 파이프 라인



실시간으로 크롤링을 할 경우 런타임이 굉장히 길어지기 때문에 AWS Lambda를 통해 위의 사진처럼 파이프라인을 작성해 트리거를 설정해 특정 시간이 되면 크롤링 함수를 작동시켜 자동으로 크롤링을 하여 뉴스기사들을 미리 DynamoDB에 저장시켜 바로바로 읽어들이어 사용할 수 있게 만들어 실행 런타임을 감소시키고 뉴스기사 최신화에도 용이한 효과를 낸다.

소스코드

뉴스기사 크롤링 함수

```

while idx < newsCount:
    # NewsList가 존재할때 try, 존재하지 않는다면 Except
    try:
        table = driver.find_element(By.XPATH, '//ul[@class="list_news"]')
    except:
        print("검색 결과가 존재하지 않습니다. 다음 언론사에서 검색합니다.")
        no_exist = 1
        break

    # 네이버 뉴스의 태그를 찾는 과정
    li_list = table.find_elements(By.XPATH, './li[contains(@id, "sp_nws")]')
    area_list = [li.find_element(By.XPATH, './div[@class="news_area"]') for li in li_list]
    info_list = [info.find_element(By.XPATH, './div[@class="news_info"]') for info in area_list]
    group_list = [group.find_element(By.XPATH, './div[@class="info_group"]') for group in info_list]
    a_list = [naver.find_element(By.XPATH, './a[@class="info"][1]') for naver in group_list]
    time.sleep(sleep_sec)

    for n in a_list[:len(a_list)]:
        if idx == newsCount:
            break

        n_url = n.get_attribute('href')

        isValid, soup = ivaf.is_valid_article(n_url)
        if isValid == True:
            title, content = cmtf.crawling_main_text(soup, press)

            # 문장이 비어있거나(이럴경우 'a' 출력) 같은 뉴스가 두번 저장될 경우
            if content == 'a' or dict_idx - 1 >= 0 and content == news_dict[dict_idx - 1]['content']:
                idx += 1
                newsCount += 1
                continue
            news_dict[dict_idx] = {'title': title,
                                   'keyword': keyword,
                                   'agency': press,
                                   'url': n_url,
                                   'content': content}

            idx += 1
            dict_idx += 1
        elif isValid == False:
            idx += 1
            newsCount += 1
            continue
  
```

뉴스기사 불필요문장 제거

```
##### 전처리 #####
def trimming_text(press, text):
    # 종종 위의 text가 섞여 있을 수도 있다고 함.
    rare_pattern = ""[\\n\\n\\n\\n// flash 오류를 우회하기 위한 함수 추가\\nfunction _flash_removeCallback() {}""
    text = text.replace(rare_pattern, '')

    text = text.replace('\\n', '').replace('\\r', '').replace('<br>', '').replace('\\t', '').strip()

    # 첫 문장 시작이 [로 시작해서 ]로 끝나는 문장 지우기
    pub_pattern1 = re.compile("^\\[[^\\]]+(?=\\)]\\]")
    text = pub_pattern1.sub("", text)

    # [[] 이렇게 되있는 문장이 있어서 한번더 점검
    pub_pattern1_2 = re.compile("^\\[\\]")
    text = pub_pattern1_2.sub("", text)

    # 첫 문장 시작이 >로 시작해서 html로 끝나는 문장 지우기
    pub_pattern2 = re.compile(">.+(?=html)html")
    text = pub_pattern2.sub("", text)

    # ~ 기자로 끝남.
    pub_pattern3 = re.compile("[.]{3,6}기자")
    text = pub_pattern3.sub(".", text)

    ##### 불필요한 기호 제거 #####
    needless_sign1 = re.compile("—")
    text = needless_sign1.sub("", text)

    needless_sign2 = re.compile("△")
    text = needless_sign2.sub("", text)
```

이벤트 트리거

이벤트 일정 정보

Cron 식

0 13 ? * MON *

다음 10개 트리거 날짜

Mon, 25 Sep 2023 13:00:00 UTC
Mon, 02 Oct 2023 13:00:00 UTC
Mon, 09 Oct 2023 13:00:00 UTC
Mon, 16 Oct 2023 13:00:00 UTC
Mon, 23 Oct 2023 13:00:00 UTC
Mon, 30 Oct 2023 13:00:00 UTC
Mon, 06 Nov 2023 13:00:00 UTC
Mon, 13 Nov 2023 13:00:00 UTC
Mon, 20 Nov 2023 13:00:00 UTC
Mon, 27 Nov 2023 13:00:00 UTC

특정 유사도 수치 이상의 뉴스기사 군집 추출

```
# ko-sentence-transper을 이용해 키워드와 관련도를 계산하여 특정 점수 이상의 군집만 가져옴
use_cluster_labels = [] # 키워드들에 교집합된 cluster_label_dbscan 저장(라벨링 번호들 저장)
for i in range(1, len(set(cluster_label_dbscan)) - 1):
    # 값이 없을 경우 있으면 넘어가기
    if len(data.loc[data['cluster_label_dbscan'] == i]['nouns']) < min_sample: continue

    # ko-sentence 사용, 현재 i(형성된 군집 중 하나)의 유사도 수치구하기
    print("현재 군집 :", i)
    mean = gsf.ko_sentence_func(data.loc[data['cluster_label_dbscan'] == i]['nouns_join'], queries)

    # 특정 수치의 유사도 수치가 나왔을 때 현재 군집 라벨링번호 저장(n에다가 저장)
    if mean > 0.4:
        print("<<<해당 키워드 포함 확인>>>")
        print("군집 번호 :", i)
        print("평균 : %.4f" % (mean))
        use_cluster_labels.append(i)
        print()
```

```
현재 키워드 : 금리 물가 경기침체
(Score: 0.4592)
(Score: 0.4428)
(Score: 0.4299)
(Score: 0.4169)
(Score: 0.4021)
<<<해당 키워드 포함 확인>>>
군집 번호 : 28
평균 : 0.4302
```

DB에 저장된 뉴스기사들의 TF-IDF 수치를 기반으로 DBSCAN 알고리즘을 이용해 뉴스기사들을 군집화 시키고 군집화된 뉴스기사들의 유사도를 Ko_sentence-transper 라이브러리를 통해 구하여 특정 수치의 유사도가 넘어가면 이를 키워드들의 내용이 전부 포함된 내용으로 간주하고 이 군집 데이터를 따로 읽어들이어 Post-processing 단계에서 요약에 사용된다.

1- 3. 웹 페이지.

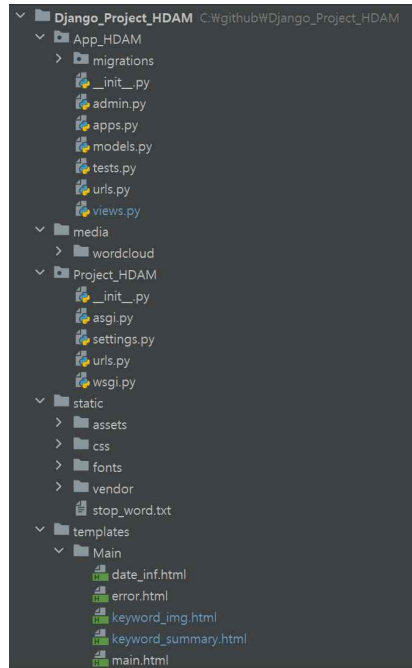
- 분석

키워드 분석에 대한 결과(키워드, 워드 클라우드)

그리고, 후처리된 내용, 후처리된 내용과 관련된 기사를 보여줄 수 있어야 한다.

- 설계

Django 프로젝트 구성



웹 페이지는 MVT 패턴의 Django를 사용하여 제작하였다.

boto3를 이용하여, AWS DynamoDB로부터, 키워드에 대한 정보를 가져오고, 워드 클라우드를 생성하기 위한 정보를 받아와, 워드 클라우드를 생성하고 Django의 템플릿 태그를 이용해 웹 화면으로 보여 준다.

```
# 선택한 일자가 담겨있는 변수 date를 이용해, 그에 맞는 키워드들을 가져와야 한다. (KEYWORD 테이블에서)
# DynamoDB에서 KEYWORD 테이블의 키워드들을 가져오기
table_keyword = dynamodb.Table('KEYWORD')

# 키워드 가져오기, (Query 사용시, 파티션 키 정렬 키 사용)
# 고정 키워드 가져오기
response_fix = table_keyword.query(
    KeyConditionExpression=Key('DATE').eq(date) &
    Key('TYPE_ID').begins_with('FIX') # 고정 키워드 인 것
)

# 어제 일일 키워드 가져오기
response_day = table_keyword.query(
    KeyConditionExpression=Key('DATE').eq(date) &
    Key('TYPE_ID').begins_with('DAY') # 일일 키워드 인 것
)
```

```
# DynamoDB에서 WORDCLOUD 테이블에서 정보 가져오기
table_wordcloud = dynamodb.Table('WORDCLOUD')

# 워드 클라우드 데이터 가져오기
response_wordcloud = table_wordcloud.query(
    KeyConditionExpression=Key('DATE').eq(date) &
    Key('DAY_ID').begins_with('DAY') # 워드 클라우드 데이터를 가져오기.
)

wordcloud_item = response_wordcloud['Items']
```



```
def generate_wordcloud(data, date):
    wordcloud_name = str(date) + ".png"
    wordcloud_save_path = 'media/wordcloud/' + str(wordcloud_name)

    # 이미지 파일이 이미 존재하는지 확인. 존재할 시 워드 클라우드를 생성하지 않음
    if os.path.exists(wordcloud_save_path):
        return wordcloud_save_path

    figure = fig.Figure(figsize=(8, 4)) # Figure를 직접 생성
    canvas = FigureCanvas(figure) # FigureCanvas 객체 생성
    ax = figure.add_subplot(111) # Axes 객체 추가

    # 워드클라우드 생성 및 그림 그리기
    font_path = 'static/fonts/NanumSquareNeo-cBd.ttf'
    wordcloud = WordCloud(width=1000,
                           height=1000,
                           background_color='white',
                           font_path=font_path,
                           max_words=20,
                           max_font_size=300).generate_from_frequencies(data)

    ax.imshow(wordcloud, interpolation='bilinear')
    ax.axis('off')

    # 이미지 파일로 저장
    canvas.print_png(wordcloud_save_path)

    return wordcloud_save_path
```

```
<div class="btn-group btn-group-toggle mb-4" data-toggle="buttons">
  <!-- 체크박스예 name 속성과 value 속성 추가 -->
  {% for keyword in fix_keyword %}
    <label class="btn btn-info rounded-pill" style="margin-right: 10px;">
      <input type="checkbox" name="keyword" value="{{keyword.VALUE}}" autocomplete="off">{{keyword.VALUE}}
    </label>
  {% endfor %}
</div>

<!-- 일일 키워드 부분-->
<div class="d-sm-flex align-items-center mb-4">
  <h1 class="h3 mb-0 text-gray-800">어제의 일일 키워드</h1>
</div>

<div class="btn-group btn-group-toggle mb-4" data-toggle="buttons">
  <!-- 체크박스예 name 속성과 value 속성 추가 -->
  {% for keyword in day_keyword %}
    <label class="btn btn-info rounded-pill mb-4" style="margin-right: 10px;">
      <input type="checkbox" name="keyword" value="{{keyword.VALUE}}" autocomplete="off">{{keyword.VALUE}}
    </label>
  {% endfor %}
</div>
```

```
<div class="card-body text-center">
  
</div>
```

1- 4. 후처리 및 요약

- 분석

DB의 저장된 데이터들을 군집화를 통해 주제별로 분류하고 분류된 글 중 키워드와 관련된 내용을 반복 사용이 가능한 그래프 기반의 데이터 형태로 변환 후 NLP모델을 통해 요약을 반복 진행하여 콘텐츠 큐레이션을 진행하도록 한다.

- 설계

위의 그림과 같이 그래프와 노드의 객체를 선언하여 내부 메서드를 사용하여 텍스트 중요도 계산, 유사도 측정 등 다양한 연산을 사용해 다중 문서를 하나의 문서로 출력하도록 그래프를 생성하는 구조를 하고 있다.

- 현재까지의 구현 및 테스트

그래프의 전체적인 뼈대를 구성하였으며 앞으로 통계적 기반의 그래프로 교체하여 중요도가 높아진 문장에 대한 타당성을 제공할 것이다.

Graph

- nodes
- edges
- link_node
- Top_rate_node
- Return_edge_idx
- Linking_node
- News_reform(요약문 생성)
- Extract_cad_node
- Display

Node

- idx
- Sentence_array
- Key_word
- Check_sim
- Extract_sim

Node



Node



Node



그래프 구현

```

class Node:
    def __init__(self, idx):
        self.idx = idx
        self.sentence_array = np.array([])
        self.key_word = np.array([])
        self.before_node_idx = 0

    def check_sim(self, word):
        #작은것의 길이를 기준으로 타 문장과 비교함
        stand_len = len(set(self.key_word) | set(word))
        inter_len = len(set(self.key_word) & set(word))
        try:
            score = inter_len / stand_len
        except:
            return 0
        return score

    def extract_sim(self, word):
        stand_len = len(set(self.key_word))
        inter_len = len(set(self.key_word) & set(word))
        try:
            score = inter_len / stand_len
        except:
            return 0
        return score

    def add_keyword(self, word):
        self.key_word = np.array(list(set(np.append(self.key_word, word))))

    def add_sentence(self, sentence):
        self.sentence_array = np.append(self.sentence_array, sentence)

    def return_idx(self):
        return self.idx

    def return_key_word(self):
        return self.key_word

    def return_sentence(self):
        return self.sentence_array

    def return_rep_sentence(self):
        return self.sentence_array[0]

```

```

class Graph:
    def __init__(self):
        self.nodes = []
        self.edges = {}
        self.link_node = []

    def add_node(self, word_vec, sentence):
        idx = len(self.nodes)
        node = Node(idx)
        self.edges[node] = []
        node.add_keyword(word_vec)
        node.add_sentence(sentence)
        self.nodes.append(node)

    def add_to_node(self, idx, word_vec, sentence):
        self.nodes[idx].add_keyword(word_vec)
        self.nodes[idx].add_sentence(sentence)

    def add_edge(self, node1, node2):
        if node1 in self.nodes and node2 in self.nodes:
            self.edges[node1].append(node2)

    def top_rate_nodes(self, sentence_vec):
        score_array = np.array([])
        for node in self.nodes:
            score = node.check_sim(sentence_vec)
            score_array = np.append(score_array, score)
        if score_array[score_array.argmax()] > 0.6:
            return score_array.argmax()
        else:
            return -1

    def return_edge_idx(self, idx):
        temp_edge = self.edges[self.nodes[idx]]
        temp_array = []
        for i in temp_edge:
            temp_array.append(i.idx)
        return temp_array

    def linking_node(self):
        link = []
        i = 0
        while self.return_edge_idx(i)[0] not in link:
            next_idx = self.return_edge_idx(i)
            i = next_idx[0]
            link.append(i)
        self.link_node = link
        return link

    def news_reform(self):
        reformed_news = ""
        for N in self.link_node:
            reformed_news = reformed_news + self.nodes[N].return_rep_sentence()
        return reformed_news

    def extract_cand_node(self, sentence):
        sentence_vec = title2idx(sentence, vocab_dict)
        save_idx = 0
        for idx in self.link_node:
            score = self.nodes[idx].extract_sim(sentence_vec)
            if score > 0.7:
                self.link_node.remove(idx)

    def display(self):
        for node in self.nodes:
            neighbors = self.edges[node]
            neighbors_idx = [neighbor.idx for neighbor in neighbors]
            print(f"{node.idx} -> {neighbors_idx}")

```

2. 프로젝트 수행을 위해 적용된 추진전략, 수행 방법의 결과를 작성하고, 만일 적용과정에서 문제점이 도출되었다면 그 문제를 분석하고 해결방안을 기술하시오.

- 크롤링 파트

기존의 방식은 사용자에게 키워드를 입력받는 방식이었지만, 입력을 받는 동시에 크롤링을 시행한다면, 상당히 큰 시간이 소요되는 문제점이 존재하였다.

이러한 시간적 제약으로 인해 고정 키워드, 일일 키워드를 미리 추출하여 AWS Lambda의 eventbridge를 사용해 키워드들을 특정 시간대에 미리 크롤링하는 방식으로 수정하였다.

- 후처리 및 요약파트

postprocessing 과정을 수행하며 H/W적 자원 부족 및 데이터 부족으로 인하여 인공지능 모델을 설계 하는 것에 대하여 많은 문제점이 발생하였고, 다른 기존 모델을 활용하여 요약을 진행할 시 내용의 축약이 많이 되며 다중 문서를 요약함에 있어서 모델의 입력의 크기가 기하급수적으로 커져 전체적인 내용을 담는 하나의 글을 작성하기로 하여 그래프라는 데이터 구조를 생각하게 되었다.

프로젝트명 : 키워드 기반 경제 콘텐츠 큐레이션

소프트웨어 요구사항 정의서

Version 1.0

개발 팀원 명 (팀리더) : 송재민
박재필
이용경

대표 연락처 : 010-5118-4832
e-mail : 20181622@edu.hanbat.ac.kr

목차

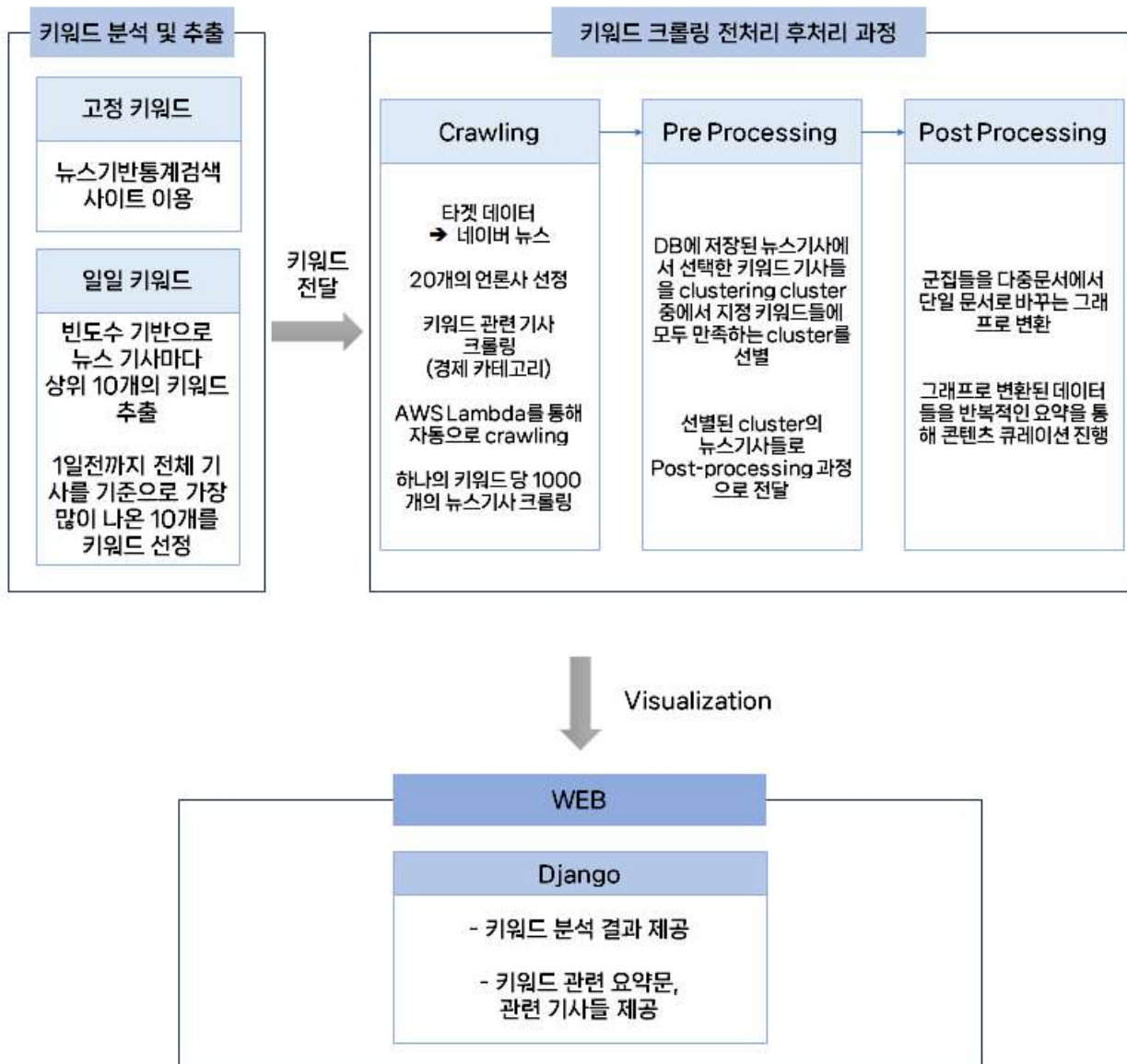
1. 개요
2. 시스템 장비 구성요구사항
3. 기능 요구사항
4. 성능 요구사항
5. 인터페이스 요구사항
6. 데이터 요구사항
7. 테스트 요구사항
8. 보안 요구사항
9. 품질 요구사항
10. 제약 사항
11. 프로젝트 관리 요구사항

1. 시스템 개요

1) 목표

현재 재테크에 대한 관심이 뜨거워지는 가운데, 이에 대한 여러 정보들을 얻기 위해 커뮤니티 게시글이나, 지인들로부터 정보를 얻던 과거와 달리 현재는 뉴스, 신문 등 관련 기사를 통해 보다 더 전문적인 재테크 정보를 얻으려는 모습이 나타나고 있다. 이에 대하여, 바쁜 현대사회에서 모든 경제 뉴스를 전부 보는 것은 어려운 일이기
에, 용이하게 재테크 관련 정보를 얻기 위해서 인기 키워드로 연결되어있는, 재테크
에 관한 콘텐츠들을 요약하여 보여주는 것을 목표로 한다.

2) 시스템 구상도



2. 시스템 장비 구성 요구사항

요구사항 고유번호		ECR-001		
요구사항 명칭		프로젝트 시각화 프레임워크		
요구사항 분류		시스템 장비 구성 요구사항	응락수준	필수
요구사항 상세설명	정의	웹 프레임워크		
	세부내용	<ul style="list-style-type: none"> - 장비 품목 : Django - 장비 수량 : 1식 - 장비 기능 : 풀스택 웹 프레임워크 - 장비 성능 및 특징 : 파이썬 웹 프레임워크, MVT 기반, URL 직관적이게 쉽게 표현 가능 등 		

요구사항 고유번호		ECR-002		
요구사항 명칭		프론트 엔드 프레임워크		
요구사항 분류		시스템 장비 구성 요구사항	응락수준	필수
요구사항 상세설명	정의	프론트엔드 프레임워크		
	세부내용	<ul style="list-style-type: none"> - 장비 품목 : bootstrap (start Bootstrap SB-Admin2) - 장비 수량 : 1식 - 장비 기능 : 오픈 소스 프론트엔드 프레임워크 - 장비 성능 및 특징 : 각종 레이아웃, 버튼 입력창 등 디자인과 기능 제공 		

요구사항 고유번호		ECR-003		
요구사항 명칭		DB 관리 소프트웨어		
요구사항 분류		시스템 장비 구성 요구사항	응락수준	필수
요구사항 상세설명	정의	- DB 저장 소프트웨어		
	세부내용	<ul style="list-style-type: none"> - 장비 품목 : AWS DynamoDB - 장비 수량 : 1식 - 장비 기능 : DB - 장비 성능 및 특징 : 키 값과 문서 데이터 모델을 지원하는 NoSQL 데이터베이스의 일종으로 읽고 쓰기 편하며 저장형식이 유연하고 AWS 기능들과의 연동성이 뛰어나. 		

요구사항 고유번호		ECR-004		
요구사항 명칭		자동 크롤링 동작 도구		
요구사항 분류		시스템 장비 구성 요구사항	응락수준	필수
요구사항 상세설명	정의	- 자동 크롤링 실행		
	세부내용	<ul style="list-style-type: none"> - 장비 품목 : AWS Lambda - 장비 수량 : 1식 - 장비 기능 : Lambda Function - 장비 성능 및 특징 : eventbridge를 통해 트리거를 설정하여 특정 시간대 매일 한 번씩 크롤링을 동작하는 함수를 실행하여, 자동으로 크롤링을 동작시켜, 데이터를 수집한다. 		

3. 기능 요구사항

요구사항 고유번호		SFR-001		
요구사항 명칭		Auto Data Crawling		
요구사항 분류		기능 요구사항	응답수준	필수
요구사항 상세설명	정의	- 자동 크롤링을 통한 데이터 수집		
	세부내용	- 뉴스 포털 사이트 등 여러 사이트에서 정보를 수집한다. - Selenium, BeautifulSoup, Requests 등의 라이브러리 사용 - AWS Lambda를 통해 매일 자동으로 크롤링을 실행해 실시간 뉴스 기사를 수집한다.		

요구사항 고유번호		SFR-002		
요구사항 명칭		Keyword Analysis		
요구사항 분류		기능 요구사항	응답수준	필수
요구사항 상세설명	정의	키워드 분석을 통한, 키워드 추출		
	세부내용	- 고정키워드, 일일키워드로 분류 - 고정키워드 : 뉴스 기반 통계 검색의 경제 키워드 추출 사용 (일주일 단위) - 어제의 일일 키워드 : 어제 하루 경제 뉴스들을 분석하여, 어제의 일일 경제 키워드를 추출 (하루 단위) - 공통 : Selenium 사용 - 일일 키워드 파트 : TextRank 알고리즘, Komoran 형태소 분석기 등 사용		

요구사항 고유번호		SFR-003		
요구사항 명칭		NLP clustering		
요구사항 분류		기능 요구사항	응답수준	필수
요구사항 상세설명	정의	- 자연어 처리 - 유사한 주제끼리 분류		
	세부내용	- 벡터화 시켜 distance를 계산하여 군집화시킨다. - 크롤링한 뉴스 기사를 유사한 주제끼리 clustering을 실시한다.		

요구사항 고유번호		SFR-004		
요구사항 명칭		NLP Similarity measurement		
요구사항 분류		기능 요구사항	응답수준	필수
요구사항 상세설명	정의	- clustering된 데이터들의 유사도 측정		
	세부내용	- Ko-sentence-transper를 통해 clustering된 뉴스기사들 중에서 키워드들과 유사도를 측정하여 일정 수치의 이상의 cluster들을 선별한다.		

요구사항 고유번호		SFR-005		
요구사항 명칭		NLP Generation		
요구사항 분류		기능 요구사항	응답수준	필수
요구사항 상세설명	정의	- 자연어 처리 - 전체 내용을 요약하여 정해진 형식으로 출력		
	세부내용	- Generation 모델을 사용하여 여러개로 나누어진 내용을 하나의 글로 작성한다.		

4. 성능 요구사항

요구사항 고유번호		PER-001		
요구사항 명칭		크롤링 시간에 따른 딜레이		
요구사항 분류		성능 요구사항	응락수준	필수
요구사항 상세설명	정의	- 정보 크롤링 시간		
	세부내용	- 서버의 접근 금지를 막기위해, 그리고 동적크롤링을 위해 필수적으로 <code>time.sleep()</code> 사용		


요구사항 고유번호		PER-002		
요구사항 명칭		빠른 속도를 위한 멀티쓰레드 사용		
요구사항 분류		성능 요구사항	응락수준	필수
요구사항 상세설명	정의	- 크롤링 시간, 키워드 분석 시간 축소		
	세부내용	- Threading 모듈로 멀티 쓰레드 사용. - 키워드 분석, 크롤링시, 약 3배의 시간을 축소		

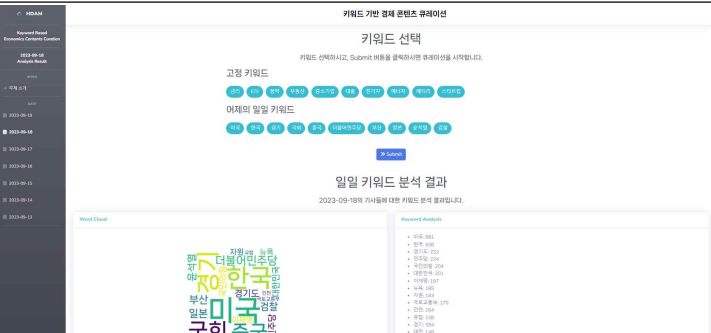
요구사항 고유번호		PER-003		
요구사항 명칭		AWS Lambda 런타임 제한		
요구사항 분류		성능 요구사항	응락수준	필수
요구사항 상세설명	정의	- Lambda function 실행 시 런타임 제한 고려		
	세부내용	- Lambda function 최대 런타임이 15분으로 제한되었음을 고려하여 코드를 작성한다. - 크롤링 실행 시 런타임을 고려하여 크롤링을 세분화 시켜 실행한다.		

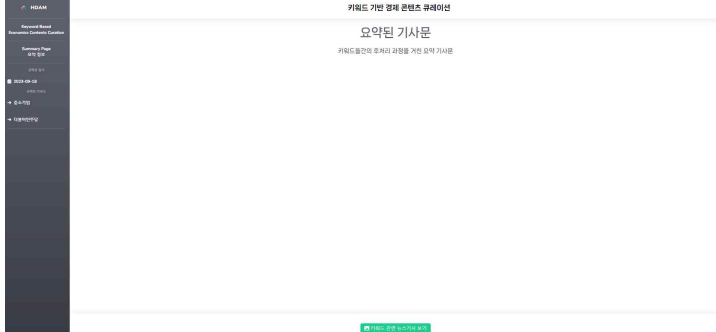
요구사항 고유번호		PER-004		
요구사항 명칭		clustering에 발생하는 성능		
요구사항 분류		성능 요구사항	응락수준	필수
요구사항 상세설명	정의	- 벡터화 중 발생하는 지연시간 평가 - 벡터의 정확도 평가		
	세부내용	- 학습된 KO-BERT 모델을 사용함에 encoding화 되며 필요한 시간을 측정한다. - 주제에 따라 벡터화된 distance를 측정하여 벡터의 정확도를 측정하여야 한다.		

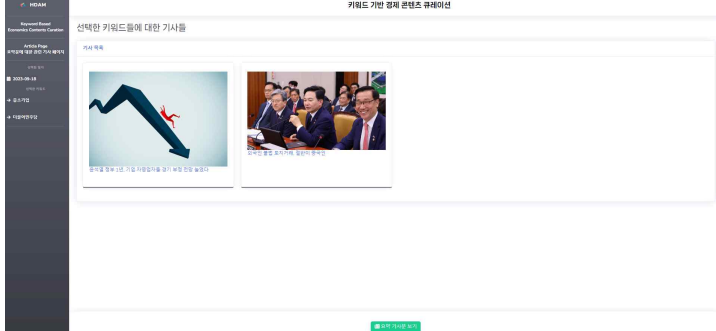
요구사항 고유번호		PER-005		
요구사항 명칭		자연어 생성에 발생하는 성능		
요구사항 분류		성능 요구사항	응락수준	필수
요구사항 상세설명	정의	- 자연어 생성에 필요한 지연시간 평가 - 자연어 생성의 정확도 평가		
	세부내용	- 자연어 생성에 글의 개수에 따른 지연시간을 측정한다. - 자연어 생성에 필요한 정보가 나열되어 있는지에 대해 정확도 평가한다.		

5. 인터페이스 요구사항

요구사항 고유번호		SIR-001		
요구사항 명칭		메인 페이지 인터페이스		
요구사항 분류		인터페이스 요구사항	응답수준	필수
요구사항 상세설명	정의	- 메인 페이지 인터페이스 요구사항		
	세부내용	- 메인 페이지에 주제에 대한 소개의 내용을 담는다.		
	화면예시			

요구사항 고유번호		SIR-002		
요구사항 명칭		키워드 분석 페이지 인터페이스		
요구사항 분류		인터페이스 요구사항	응답수준	필수
요구사항 상세설명	정의	- 키워드 분석 페이지 인터페이스		
	세부내용	- 7일 전부터 어제까지의 날짜에서, 원하는 날짜를 선택하면 이동하게 되는 페이지 - 워드클라우드로 해당 날짜에 대한 키워드 분석 결과, 키워드를 제공하고 요약된 정보를 보고싶은 키워드를 선택한 후, 제출 버튼을 클릭하면 요약문 확인 가능		
	화면예시			

요구사항 고유번호		SIR-003		
요구사항 명칭		요약문 페이지 인터페이스		
요구사항 분류		인터페이스 요구사항	용량수준	필수
요구사항 상세설명	정의	- 키워드 분석 페이지 인터페이스		
	세부내용	- 내가 선택한 날짜, 키워드들에 대하여 교집합으로 연관되어 있는 content를 추려 보여준다. - 최종적으로 작업을 거쳐 완성된 글을 출력한다.		
	화면예시			

요구사항 고유번호		SIR-004		
요구사항 명칭		요약문 관련 기사 제공 페이지 인터페이스		
요구사항 분류		인터페이스 요구사항	용량수준	필수
요구사항 상세설명	정의	- 요약문 관련 기사 제공 페이지		
	세부내용	- 글을 만드는데 밑바탕이 된 기사들의 제목, url 등을 제공하여, 사용자가 해당 요약문에 대해, 부가적으로 궁금한 부분에 대해 확인할 수 있게끔 한다.		
	화면예시			

6. 데이터 요구사항

요구사항 고유번호		DAR-001		
요구사항 명칭		다양한 데이터 포맷		
요구사항 분류		데이터 요구사항	응락수준	필수
요구사항 상세설명	세부내용	<ul style="list-style-type: none"> - 핀테크 관련 텍스트, 이미지 등 키워드와 관련된 다양한 포맷의 데이터를 추출한다. - 추출한 데이터를 불필요한 요소들을 제거하여 사용가능한 문장으로 변환한다. 		

요구사항 고유번호		DAR-002		
요구사항 명칭		학습 데이터		
요구사항 분류		데이터 요구사항	응락수준	필수
요구사항 상세설명	세부내용	<ul style="list-style-type: none"> - 기존의 모델을 사용하되 우리가 사용하는 글의 형식에 맞추어 추가 학습하기 위한 글 데이터셋을 구축한다. 		

요구사항 고유번호		DAR-004		
요구사항 명칭		데이터 형식		
요구사항 분류		데이터 요구사항	응락수준	필수
요구사항 상세설명	세부내용	<ul style="list-style-type: none"> - 여러 가지 글을 사용하되 일정한 형식에 맞춰 날짜, 출처 등 다양한 글에 대한 정보가 필요하다. 		

요구사항 고유번호		DAR-003		
요구사항 명칭		데이터 형식 변경		
요구사항 분류		데이터 요구사항	응락수준	필수
요구사항 상세설명	세부내용	<ul style="list-style-type: none"> - 뉴스를 그래프 형식의 데이터로 변경하여 반복할 수 있으며 고유한 내용을 유지할 수 있도록 변형시킨다. 		

7. 테스트 요구 사항

요구사항 고유번호		TER-001		
요구사항 명칭		테스트 방안		
요구사항 분류		테스트 요구사항	응용수준	필수
요구사항 상세설명	세부내용	<ul style="list-style-type: none"> - 전체적인 프로세스는 Extraction, Load, Transform에서 일어나므로 각각의 task에서 test 진행한다. - Extraction 단계에서 크롤링을 진행하여 키워드에 맞는 결과를 도출하는지 평가한다. - Load 단계에서 크롤링된 데이터들을 전처리 후 이를 키워드의 교집합된 데이터를 채택하는지 평가를 진행한다. - Transform 단계에서 글들을 clustering, generation 된 결과를 확인한다. - 위 단계를 각각 확인 후 이를 통합하여 전체적인 process를 평가 		

요구사항 고유번호		TER-002		
요구사항 명칭		단위 테스트		
요구사항 분류		테스트 요구사항	응용수준	필수
요구사항 상세설명	세부내용	<ul style="list-style-type: none"> - 기능 요구사항에 기술된 요구사항에 대한 각각의 테스트를 수행한다. - 데이터 추출, 저장, 출력이 정상적으로 작동하는지 테스트한다. 		

요구사항 고유번호		TER-003		
요구사항 명칭		통합 테스트		
요구사항 분류		테스트 요구사항	응용수준	필수
요구사항 상세설명	세부내용	<ul style="list-style-type: none"> - 테스트가 완료된 최소 단위의 기능들을 통합하여 기능들이 정상적으로 연결되어 오류없이 수행되는지 테스트를 진행 - 최초 입력부터 최종 입력까지의 수행 결과가 정상적으로 출력되는지 테스트를 진행 		

8. 보안 요구사항

요구사항 고유번호		SER-001		
요구사항 명칭		출처에 대한 보호		
요구사항 분류		보안 요구사항	응답수준	필수
요구사항 상세설명	세부내용	- 인터넷에 올라와 있는 글들을 crawling하는 과정이므로 robots.txt를 확인하여 crawling이 가능/불가능한지 판단하는 과정이 필요함		

요구사항 고유번호		SER-002		
요구사항 명칭		AWS 인증키 숨김		
요구사항 분류		보안 요구사항	응답수준	필수
요구사항 상세설명	세부내용	- AWS의 DynamoDB를 사용하므로, Django 프로젝트 내부에 인증키 필요. - .env 파일 내부에, AWS 인증키(엑세스 키, 시크릿 엑세스 키, 리전)를 숨겨서 보관		

9. 품질 요구사항

요구사항 고유번호		QUR-001		
요구사항 명칭		크롤링 과정에서 변경사항에 대한 업데이트		
요구사항 분류		품질 요구사항	응답수준	필수
요구사항 상세설명	정의	크롤링 대상 웹사이트 변경사항 관리		
	세부내용	- 크롤링 대상 사이트의 페이지 코드, 태그들의 변경시, 크롤러 소스코드 업데이트를 하지 않을 시, 전체적인 로직이 정상적으로 동작하지 못함. - 지속적인 모니터링을 통한 변경사항 업데이트가 필요		

요구사항 고유번호		QUR-002		
요구사항 명칭		크롤링 범위		
요구사항 분류		품질 요구사항	응답수준	필수
요구사항 상세설명	정의	- 크롤링할 사이트 지정		
	세부내용	- 크롤링 데이터 풀을 뉴스 기사로 제약		

요구사항 고유번호		QUR-003		
요구사항 명칭		오류 발생률		
요구사항 분류		품질 요구사항	응답수준	필수
요구사항 상세설명	정의	- 오류 관리		
	세부내용	- 자연어 처리 특성상 데이터의 길이가 매우 가변적이며 이에 따라 입력된 글의 길이에 따라 오류를 발생할 수 있으므로 load 과정에서 일부 데이터는 길이에 대한 관리가 필요하다.		

10. 제약 사항

요구사항 고유번호		COR-001		
요구사항 명칭		크롤링 범위		
요구사항 분류		제약 사항	응락수준	필수
요구사항 상세설명	정의	- 크롤링 사이트 지정		
	세부내용	- 크롤링 데이터 풀을 뉴스 기사로 제약		

요구사항 고유번호		COR-002		
요구사항 명칭		키워드 선택 제약		
요구사항 분류		제약 사항	응락수준	필수
요구사항 상세설명	정의	- 키워드 선택 제약		
	세부내용	- 다중 키워드 선택이 가능한데, 키워드를 선택할 수 있는 개수를 제한시켜, 요약문의 퀄리티라든지, 전처리 시간 면에서, 높은 성능을 낼 수 있도록 한다.		

요구사항 고유번호		COR-003		
요구사항 명칭		전처리 및 필터링		
요구사항 분류		제약 사항	응락수준	필수
요구사항 상세설명	정의	- 한글 데이터 전처리		
	세부내용	- 사용하지 않는 문장, 단어, 기호 등 불필요한 요소들은 처리 필요 - 관련도 측정작업에 불필요한 것들의 처리가 필요하다.		

11. 프로젝트 관리 요구사항

요구사항 고유번호		PMR-001		
요구사항 명칭		프로젝트 수행 조직		
요구사항 분류		프로젝트 관리 요구사항	응락수준	필수
요구사항 상세설명	세부내용	- 이용경 : Django를 이용한 웹 페이지 구축 및 프론트엔드의 전반적인 부분을 담당. 크롤링, 키워드분석 파트 담당. - 송재민 : AWS를 이용한 자동 크롤링 파이프라인 구축 및 DynamoDB를 통한 데이터 저장 및 갱신하는 등 백엔드의 전반적인 부분을 담당 - 박재필 : 데이터를 그래프 형식의 데이터 구조로 바꿔 반복 사용이 가능한 데이터를 생성한 후, 그 데이터를 인공지능 요약 모델을 반복 사용해 전체적인 요약을 실행함		

요구사항 고유번호		PMR-002		
요구사항 명칭		프로젝트 일정 계획		
요구사항 분류		프로젝트 관리 요구사항	응락수준	필수
요구사항 상세설명	세부내용	- ~ 9월 : 프로젝트 기능 마무리 및 유지보수 - ~ 10월 : 프로젝트 배포 및 예외처리 실시		