종합설계 프로젝트 최종수행보고서

프로젝트명	문서토픽추출시스템
팀번호	S3-12
문서제목	수행계획서( ) 2차발표 중간보고서( ) 3차발표 중간보고서( ) 최종결과보고서(O)

2020.11.24

팀원 : 김학영 (팀장)
문용현 팀원
안윤빈 팀원

지도교수 : 박정민 교수 (인)
지도교수 : 교수 (인)

문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	
2019.12.18	김학영 (팀장)	1.0	수행계획서	최초작성
2020.02.11	안윤빈	2.0	2차발표자료	설계서추가
2020.04.28	문용현	3.0	3차발표자료	시험결과추가
2020.11.24	안윤빈	4.0	최종결과보고서	시험결과 수정

문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~3) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~4) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~5) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~7) 참고자료	Ⅰ Ⅱ Ⅲ

이 문서는 한국산업기술대학교 컴퓨터공학부의
“종합설계” 교과목에서 프로젝트 “문서토픽추출시스템” 을
수행하는
(S3-12, 김학영,문용현,안윤빈) 들이 작성한 것으로 사용하기
위해서는 팀원들의 허락이 필요합니다.

목 차

Ⅰ. 서론

1. 작품선정 배경 및 필요성

필요성	내 용
수요배경	• 정보의 양이 기하급수적으로 증가하여 많은 정보를 다 하나하나 살펴보는 것이 어려워졌고, 문서의 핵심 키워드 위주로 정보를 찾으려고 하는 사람의 수가 늘어남 • 책 구매 시 웹 사이트 및 SNS를 이용하여 책 리뷰를 참고하는 소비자들이 증가 • 문서의 토픽을 추출하여 추출된 토픽을 쉽게 볼 수 있게 시각화 해주는 시스템의 필요성 증가
필요성 1	• 기존의 리뷰 분석은 양적 분석과 주제별 내용 분석이 주를 이룸 • 분석 도구를 사용하는 경우는 그 방법이 다양하고 일관되지 않아 문서마다 도출된 연구결과를 분석하기가 어려운 실정 • 연구자의 주관적 가치와 개인적인 의견이 반영될 위험성이 있으며 많은 분량의 데이터를 소화하기 어려움
필요성 2	• 토픽 분석을 통해 유망기술을 파악하고 예측하여 불확실성에 대해 미리 준비하는 것이 중요 • 양적 분석과 기존의 내용 분석이 갖는 단점을 보완 가능
필요성 3	• 웹을 통해 결과를 출력하여 사용자의 편의 증대 • 웹 크롤링을 통해 데이터 수집이 용이해짐

2. 기존 연구/기술동향 분석

- **[1] 동적토픽모델 기법을 적용한 특정 뉴스 토픽의 변화과정 분석(배진희,2019)** :
  잠재 디리클레 할당 모형의 다양한 변용 모형 중 문서의 **시간성을 고려하지 않는다**는 문제를 개선한 **동적 토픽 모형**에 대해 소개
  장점: **최적의 토픽개수**를 찾기 위한 원리와 토픽개수에 따른 복잡도의 변동 추이 확인 가능
  단점: 토픽을 분석하는 것 에만 초점이 맞춰져 있어 **데이터 시각화**를 고려하지 않음

- **[2] 일기자료 연구에서 토픽모델링 기법의 활용가능성 검토(남춘호,2016)** :
  해당 분야에 대한 별다른 사전적 전문지식 없이도 방대한 디지털 텍스트 자료로부터 소수의 의미 있는 토픽을 추출해 주는 알고리즘으로 알려진 토픽모델링 기법의 특징과 이론적 전제를 살펴보며, 농민일기 분석에 예시적으로 적용
  장점 : 토픽모델링 기법을 사용하여 예시로 설정한 데이터를 처리하고 나온 결과를 **시각화** 하여 이해를 도와주며 관련 연구의 방향성을 제시
  단점 : 분석하고 처리한 데이터를 그래프를 이용하여 **시각화** 하였지만 그 결과를 **웹으로 보여주는 편의성**은 고려하지 않음

- **[3] 단어 유사도를 이용한 뉴스 토픽 추출(김동욱,이수원, 2017)** :
  토픽 중복문제에 대해 강건한 잠재 디리클레 할당으로 토픽을 추출하고 **단어 간 유사도**를 이용하여 토픽 분리 및 토픽 병합의 단계를 거쳐 최종적으로 토픽을 보정하는 방법 제안
  장점 : **토픽 중복 문제**, **토픽 혼재 문제**를 해결할 수 있는 방법 제안
  단점 : 많은 데이터를 분석하지 못함.

- **[4] Study on the Topic Mining and Dynamic Visualization in View of LDA Model(Ting Xie, Ping Qin, Libo Zhu, 2018)** :
  LDA 토픽 모델을 만들고 토픽 마이닝, 클러스터링, 동적 시각화에 대한 내용을 소개
  장점: LDA 모델링을 실시하고, 추상적인 수준에서 토픽 마이닝에 대해 심층적으로 설명한다.
  단점: 사용자가 직접 파일을 다운로드하여 문서를 로드 해야함

[5] 한글 키워드 추출 시스템 (http://nlp.kookmin.ac.kr/cgi-bin/indexT.cgi, 강승식)
한국어 형태소 분석기를 이용하여 키워드를 추출하는 시스템
장점: 웹을 통하여 키워드 분석을 하여 키워드의 빈도와 점수를 매기고 키워드의 순위를 매김
단점: 사용자가 직접 문서의 단위를 입력을 하여 분석을 실행해야함

공통적인 문제점 :

1. **데이터 수집의 불편함** :
토픽 추출에 관한 연구이다 보니 사용자가 직접 문서를 다운로드하여 불러오고 해야 하는 불편함이 있음

2. **분석 결과 시각화 :**
현재 나와 있는 연구의 대부분은 분석에 대한 내용이고, 사용자가 입력한 분석 결과를 웹으로 시각화하여 보여주는 시스템은 없음

3. **문서마다 일관되지 않은 분석 :**
분석 도구를 사용하는 경우는 그 방법이 다양하고 일관되지 않아 문서마다 도출된 연구결과를 분석하기가 어려운 실정

3. 개발목표

최종목표

■ **문서 토픽 추출 시스템**을 이용한 책 리뷰 핵심 키워드 추출을 통해 책의 내용 유추 및 사용자들의 리뷰 분석

단계	내 용
1단계	■ **사용자 편의 문서 수집 시스템** 구축 • 웹 크롤링을 이용한 **수집 시스템** 개발 • 사용자가 분석하고자 하는 **데이터 수집 시스템** 구축
2단계	■ **데이터 처리 및 분석 시스템** 구축 • 문서에서 **불필요한 부분을 필터링**하는 **데이터 정제 시스템** 개발 • **데이터 분석**을 위한 **통계 계산 시스템** 개발 • **토픽 추출**을 위한 **분석 시스템** 개발 • **추출된 토픽**에 대한 **감정 분석 시스템** 개발
3단계	■ 분석 시스템과 사용자간의 **인터페이스(application)** 개발 • 시각화된 분석 결과를 보여주는 **웹 시각화 시스템** 개발 • 분석 결과를 **웹으로 전송하는 전송 시스템** 구축

4. 팀 역할 분담

성명	학번	주요 역할
김학영 (팀장)	2016150012	웹 크롤링 프로세스 설계 사용가능 언어: C, python
문용현	2017152012	데이터 처리 프로세스 설계 사용가능 언어: python, JAVA
안윤빈	2017152022	데이터 분석 프로세스 설계 사용가능 언어: JAVA, C

5. 개발 일정

추진사항	12월	1월	2월	3월	4월	5월	6월	7-9월
자료수집								
요구사항 정의 및 분석								
시스템 설계								
구현								
통합 및 테스트								
유지보수								
최종 검토 및 발표								

6. 개발 환경

- 개발언어 : R 3.52, Python 3.7

- 주요 라이브러리 :

BeautifulSoup(웹 크롤링)

: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Selenium(웹 자동화)

: https://selenium-python.readthedocs.io/

pyqt5(GUI)

: https://www.riverbankcomputing.com/software/pyqt/intro

KoNLP(한글 자연어 분석)

: https://www.rdocumentation.org/packages/KoNLP/versions/0.80.1

GGPlot2(시각화 기능)

: https://ggplot2.tidyverse.org/

- 개발 툴 : R Studio, PyCharm

- O/S : Window 10 64bit, macOS

Ⅱ. 본론

1. 개발내용

순번	개 발 내 용
1	■ 웹 크롤링 시스템 개발 • python을 이용하여 시스템을 구축함. • **교보문고 홈페이지에서 사용자 리뷰 데이터 수집**하고 , **텍스트 파일로 저장**
2	■ 토픽 추출 시스템 개발 • R을 이용하여 **문서를 분석에 활용할 수 있도록 전처리**함. • 문서를 분석하기 위한 토픽 모델링 기법인 **LDA 알고리즘을 구현**함.
3	■ 감정 분석 시스템 개발 • LDA 알고리즘에서 추출된 토픽으로 **감정 분석**을 실시함. • 분석 결과를 그래프로 **시각화**함.
4	■ 웹 시각화 시스템 개발 • R을 이용하여 수신된 **시각화 데이터를 웹 애플리케이션에 출력**함. • **시각화된 데이터**를 웹 서버를 통해 **웹으로 전송**함.

1-1) LDA 알고리즘

가. LDA의 개념

LDA(Latent Dirichlet Allocation)는 대표적인 토픽모델링 기법으로 문서들이 하나 이상의 토픽을 포함하고 있다고 가정한다. 단어의 순서는 중요하지 않고 출현 빈도만으로 토픽을 추출하며 미리 알고 있는 토픽별 단어 수 분포를 바탕으로, 주어진 문서에서 발견된 단어 수 분포를 분석하여 해당 문서가 어떤 토픽들을 가지고 있을지를 예측하는 기법이다. 베이즈 추론을 이용한 깁스 샘플링 방법을 이용하여 토픽을 할당한다[10].

나. LDA의 핵심 가정

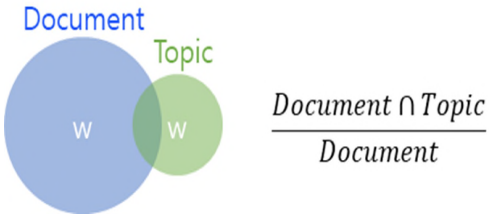
LDA에서 토픽은 고정된 단어의 분포이며 토픽의 수는 보통 연구자가 결정한다. 토픽의 단어 분포와 문서의 토픽 분포의 결합으로 문서 내 단어들이 생성된다고 가정한다. 실제 관찰 가능한 문서 내의 단어로 토픽의 단어 분포와 문서의 토픽분포를 추정한다.

다. 깁스 샘플링

LDA 알고리즘에서는 깁스 샘플링을 이용하여 토픽을 할당한다. 깁스 샘플링에서는 각 단어에 임의의 토픽이 할당된 후 문서별로 토픽과 단어(Word)의 분포가 결정된다. 모든 문서들은 토픽을 갖

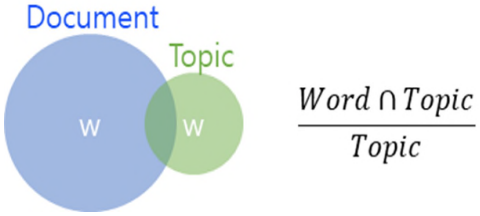
고, 모든 토픽은 단어의 분포를 가진다. 이러한 문서들에 K개의 토픽들 중 하나를 랜덤하게 할당한 뒤, 각 토픽에 대해 다음 두 가지를 계산한다[9].

① P(topic t | document d)



위 그림은 문서에 있는 단어들 중 토픽에 해당하는 단어들의 비율을 계산하는 방법을 나타낸다. 문서에 들어있는 단어들이 할당된 토픽에 얼마나 포함되는지 계산하여 문서의 단어와 토픽의 단어가 일치하는 비율이 높은 토픽을, 그 문서의 토픽으로 지정하기 위한 과정이다.

② P(word w | topic t)



위 그림은 한 단어가 속한 토픽 안에 있는 각 단어 분포를 계산하는 과정을 나타낸다. 이는 토픽이 문서들의 단어를 대표할 만큼 올바른 토픽인지 확인하기 위한 과정이다. 이후 P(topic t | document d) * P(word w | topic t)에 따라 토픽을 새로 선정한다. (토픽과 문서의 타당성) * (단어와 토픽의 타당성)이 낮은 비율을 보이면 적절하지 않은 토픽으로 간주하고 새로운 토픽을 선정한다. 위의 과정을 반복해 토픽을 선정한다.

라. LDA의 과정

우선, 사용자는 알고리즘에게 토픽의 개수 k를 알려준다. LDA는 토픽의 개수 k를 입력 받으면, k개의 토픽이 M개의 전체 문서에 걸쳐 분포되어 있다고 가정한다.

두 번째로 문서에서 추출한 모든 단어를 k개 중 하나의 토픽에 할당한다. 모든 문서의 모든 단어에 대해서 k개 중 하나의 토픽을 랜덤으로 할당하며 이 작업이 끝난 뒤, 각 문서는 토픽을 가지며 토픽은 단어 분포를 가지는 상태가 된다. 토픽을 랜덤으로 할당했기 때문에 이 결과는 전부 틀린 상태이다. 만약 한 단어가 한 문서에서 2회 이상 등장하였다면, 각 단어는 서로 다른 토픽에 할당되었을 수 있다.

세 번째로 모든 문서의 모든 단어에 대해 아래의 사항을 반복하여 진행한다. 어떤 문서의 각 단어 w는 자신은 잘못된 토픽에 할당되어져 있지만, 다른 단어들은 전부 올바른 토픽에

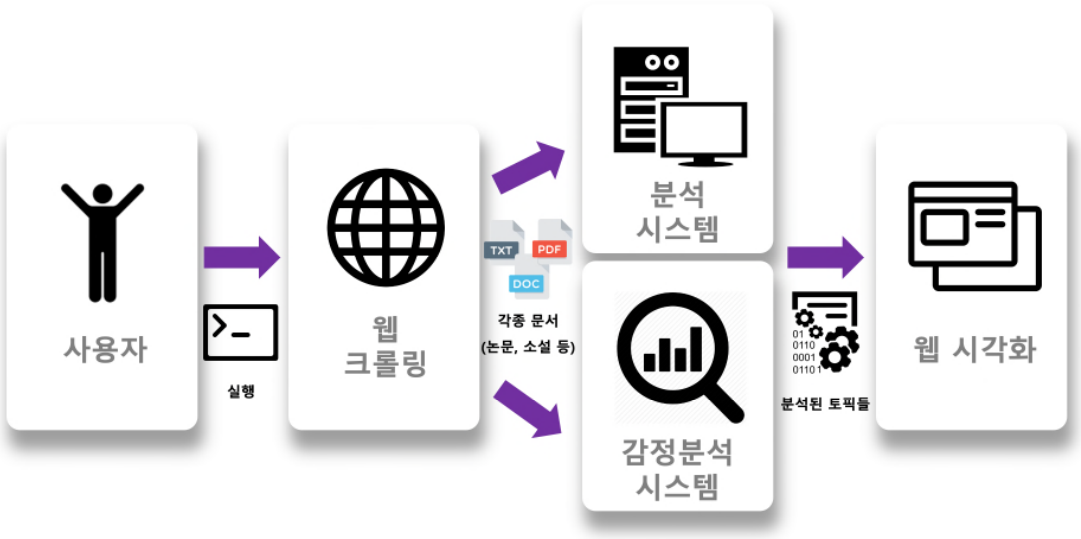
할당되어져 있는 상태라고 가정한다. 이에 따라 단어 w는 문서 d의 단어들 중 토픽 t에 해당하는 단어들의 비율인 p(topic t | document d)과 단어 w를 갖고 있는 모든 문서들 중 토픽 t가 할당된 비율인 p(word w | topic t) 두 가지 기준에 따라서 토픽이 재 할당된다. 이 과정을 반복하면, 모든 할당이 완료된 수렴 상태가 된다.

2. 문제 및 해결방안

문제	해결 방안	구체적 방법
1) 데이터 수집의 불편함	사용자의 **편의성 증대를 위한** 웹 크롤링 시스템과 GUI를 제안	• 사용자가 수집할 데이터를 선택하면 **해당 데이터를 텍스트 파일로 저장**함
2) 분석 결과 시각화를 고려하지 않아 사용자의 편의 부족	사용자가 분석된 데이터를 쉽게 확인 가능한 **토픽 분석 시스템**을 제안	• LDA 알고리즘을 활용하여 **객관적인 토픽 분석 결과를 제공**함. • 토픽이 추출된 분석 데이터의 결과를 **웹 시각화**를 이용하여 사용자가 **시각화된 데이터를 통하여 쉽게 확인이 가능**하도록 개선함.

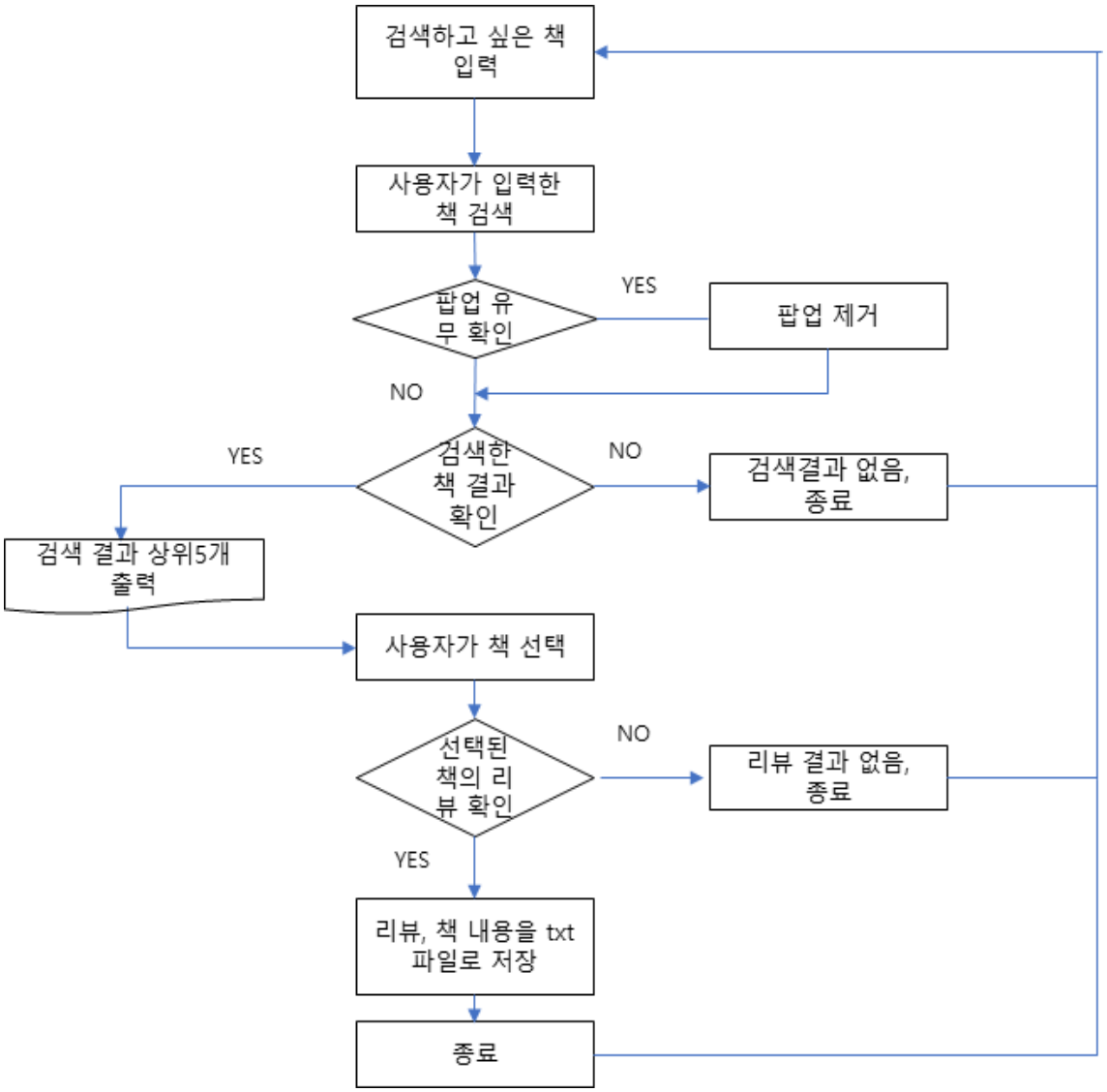
3. 시험시나리오

시나리오명	내 용
사용자	• 사용자는 크롤링할 문서를 검색 • 사용자는 다양한 책의 리뷰를 수집 가능
웹 크롤링	• 사용자가 입력한 책 이름을 자동으로 교보문고 홈페이지에서 문서들을 추출 • 추출한 문서들을 추출 시스템이 읽을 수 있는 일정한 파일 형식으로 변환 • 사용자가 사용하기 쉽게 GUI를 이용함 • 사용자는 한 번에 한 책을 크롤링 할 수 있음
문서 토픽 추출	• 웹 크롤링의 결과로 수집된 문서들을 로드 • 데이터셋과 관련된 통계를 계산 • 핵심 키워드인 토픽 추출을 수행
감정 분석	• 크롤링된 리뷰들을 로드 • 필요없는 문장부호들을 삭제 • 감정사전(긍정어,부정어사전)과 대조하여 감정분석
웹 시각화	• 추출 시스템에서 분석된 결과를 웹 환경에서 시각화 • 시각화된 데이터를 웹서버를 통해 웹 애플리케이션으로 전송



4. 상세 설계

1) 웹 크롤링 모듈



1-1) 웹 크롤링 모듈 코드

*** 웹 크롤링을 위한 변수 선언**

```
13  search = None # 검색어
14  selectNum = None # 책 선택 인덱스
15  status = False # 웹 크롤링 시작 상태 True : 진행중 , False : 꺼짐
```

*** pyqt5를 이용하여 GUI 생성**

```
149     def __init__(self,parent=None):
150         super().__init__()
151         self.ui = uic.loadUi("gui.ui")
152         self.ui.search_btn.clicked.connect(self.searchBook)
153         self.ui.quit_btn.clicked.connect(self.quitGUI)
154         self.ui.bookList.itemDoubleClicked.connect(self.select_book)
155         self.ui.show()
```

- 151 : Qt designer를 이용하여 생성한 .ui 파일을 불러온다.
- 152 ~ 154 : Object에 지정한 이벤트(152: 클릭, 153: 클릭, 154: 더블클릭)를 적용한다.
- 155 : 불러온 UI를 출력

*** 검색 버튼 이벤트**

```
159     def searchBook(self):
160         global search
161         global status
162         if status is False:
163             status = True
164             self.worker = Worker()
165             search = self.ui.search_name.text()
166             self.worker.finished.connect(self.update_list)
167             self.worker.runCheck.connect(self.checkEnd)
168             self.worker.reviewCheck.connect(self.checkReview)
169             self.worker.start() # 웹 크롤링 시작
```

status(웹크롤링 시작상태 True : 진행중, False : 꺼짐)

- 164 : 크롤링 Thread 생성
- 165 : 사용자가 입력한 값 저장
- 166 ~ 168 : Thread에서 보낸 데이터를 ui와 통신 하기위해 pyqtSlot으로 연결
- 169 : 크롤링 Thread 실행

* pyqtSlot 이벤트 선언

①. 검색 된 상위 5개의 책 리스트 변수 데이터 처리 이벤트

```
177     @pyqtSlot(list)
178     def update_list(self, data):
179         # data에 books에 있는 책 리스트가 넘어옴
180         # 만약 책 리스트가 비어있을 시 팝업 출력
181         if len(data) == 0:
182             QtWidgets.QMessageBox.about(self,'Message','검색 결과 없음, 다시 검색하세요  ')
183         # 책 리스트가 있을 시 bookList에 값 추가
184         else:
185             # 책 이름으로 추가
186             for bookName in data:
187                 self.ui.bookList.addItem(str(bookName))
```

- 182 : QMessageBox를 이용하여 책 리스트가 비어 있을 시에 팝업 출력
- 187 : bookList Object에 bookName을 추가하여 검색된 상위 5개의 책 출력

②. 크롤링 끝났을 시 이벤트

```
191     def checkEnd(self,data):
192         QtWidgets.QMessageBox.about(self,'Message','리뷰 추출이 완료되었습니다.')
193         self.ui.bookList.clear()
```

- 193 : bookList Object를 비움

③. 검색된 리뷰가 없을 시 이벤트

```
195     @pyqtSlot(bool)
196     def checkReview(self,data):
197         if data == False:
198             QtWidgets.QMessageBox.about(self,'Message','리뷰가 없습니다. 종료하겠습니다.')
199             self.ui.bookList.clear()
```

* 종료 버튼 이벤트

```
156     def quitGUI(self):
157         QtWidgets.QMessageBox.about(self, 'Message', '프로그램을 종료합니다')
158         exit()
```

* 책 선택 이벤트

```
170     def select_book(self):
171         global selectNum
172         # QtWidgets.QMessageBox.about(self, 'Message', self.ui.bookList.currentItem().text())
173         selectNum = self.ui.bookList.currentRow()+1
174         # print(self.ui.bookList.currentRow())
175         print(self.ui.bookList.currentItem().text())
```

173 : bookList Object에서 사용자가 더블클릭하여 선택된 책 번호 저장

* 크롤링 Thread class

```
16  class Worker(QThread):
17      finished = pyqtSignal(list)
18      runCheck = pyqtSignal(bool) # 크롤링 후 팝업 띄우기용
19      reviewCheck = pyqtSignal(bool) # 리뷰 유무 확인 True : review 있음, False : 없음
20      def run(self):
21          global status
22          global search
```

- 17 ~ 19 : pyqtSlot과 연결하기위한 Signal 선언

```
20      def run(self):
21          global status
22          global search
23          global selectNum
24          checkend = False # False : 진행중 , True: 종료됨
25          reviewExist = None # True : review 있음, False : 없음
26          path = "C:/dev/chromedriver.exe"
27          print(path)
28          driver = webdriver.Chrome(path)
29          driver.implicitly_wait(3)
30          driver.get('http://www.kyobobook.co.kr/index.laf')
31          # time.sleep(2)
32          driver.implicitly_wait(3)
```

- 21 ~ 23 : 전역변수 불러오기
- 24 : 크롤링 진행 상태 변수
- 25 : 리뷰 유무 변수
- 26 : 웹 자동화를 위한 selenium 사용에 쓰일 webdriver 경로
- 28 : Chrome을 이용한 Webdriver 객체
- 29 : 웹 브라우저를 불러오기까지 기다림
- 30 : 크롤링에 쓰일 교보문고 홈페이지 연결

```
33          if driver.window_handles[1]:
34              print("팝업 제거.")
35              driver.switch_to.window(driver.window_handles[1])
36              driver.close()
37              driver.switch_to.window(driver.window_handles[0])
38          driver.find_element_by_xpath("//*[@id='searchKeyword']").send_keys(search)
39          driver.find_element_by_xpath("/html/body/div[4]/div[1]/div[1]/div[1]/form[2]/div/input").click()
40          books = [] # 검색한 책 리스트
41          driver.implicitly_wait(3)
```

- 33 : 팝업이 떠있는지 확인
- 35 ~ 36 : 현재 window를 팝업으로 변경 후 팝업 종료
- 37 : 기본 window로 변경
- 38 ~ 39 : 사용자가 입력한 search 변수 값을 검색창에 입력 후 검색

```
42          try:
43              bookExist = driver.find_element_by_xpath("//*[@id='search_list']/tr[1]/td[2]/div[2]/a")
44              # 검색한 책 찾았을 경우
45              for i in range(1, 6):
46                  try:
47                      bookName = driver.find_element_by_xpath(
48                          "//*[@id='search_list']/tr[" + str(i) + "]/td[2]/div[2]/a/strong").text
49                      print(str(i) + ".", bookName)
50                      books.append(bookName)
51                  # 책 이름이 없을 시 반복문 종료
52                  except NoSuchElementException:
53                      break
54              self.finished.emit(books) # 검색 완료 후 결과 전송(검색 결과 있음)
55
56              # 사용자가 책을 선택할 때 까지 대기
57              while True :
58                  if selectNum is None:
59                      pass
60                  else : break
61          except NoSuchElementException:
62              # 검색한 책 찾지 못했을 경우
63              self.finished.emit(books) # 검색 완료 후 결과 전송(검색 결과 없음)
```

- 43 : try, except를 이용하여 bookExist 변수에 검색된 책 존재 확인
- 45 ~ 50 : bookName에 책 리스트의 이름이 존재하면 저장
- 52 : 검색된 책이 5개 미만이면 종료
- 54 : finished pyqtSignal 이벤트로 Ui에 books 리스트 전송
- 57 ~ 60 : 사용자가 UI에서 책을 선택할 때 까지 대기

```
61              except NoSuchElementException:
62                  # 검색한 책 찾지 못했을 경우
63                  self.finished.emit(books) # 검색 완료 후 결과 전송(검색 결과 없음)
64                  search = None
65                  selectNum = None
66                  status = False
67                  checkend = True
68                  driver.quit()
69                  # self.quit()
70                  return None
```

- 61 ~ 63 : bookExist 변수에 검색된 책이 없을 경우, UI 에 books 리스트 전송
- 64 ~ 70 : 종료를 위한 변수 초기화 와 webdriver 종료

```
71          selectedBook = driver.find_element_by_xpath("//*[@id='search_list']/tr[" + str(selectNum) + "]/td[2]/div[2]/a")
72          selectedBookName = driver.find_element_by_xpath("//*[@id='search_list']/tr[" + str(selectNum) + "]/td[2]/div[2]/a/strong").text
73          selectedBook.click()
74          window_before = driver.window_handles[0]
75          try:
76              showReview = driver.find_element_by_link_text("전체보기")
```

- 71 ~ 73 : 사용자가 선택한 책 클릭
- 74 : 리뷰 창을 처리하기위한 현재 창 저장
- 76 ~ 78 : 리뷰를 보기위한 showReview 변수 확인
- 79 ~ 89 : showReview가 존재하지 않을 경우 종료
- 82 : reviewExist 데이터를 reviewCheck pyqtSignal 이벤트를 이용해 UI로 전달

```
90              showReview.click()
91              window_after = driver.window_handles[1]
92              driver.switch_to.window(window_after)
93
94              print("현재윈도우 URL : ", driver.current_url)
95              print("변경전윈도우: ", str(window_before))
96              print("변경후윈도우: ", str(window_after))
```

- 91 ~ 92 : showReview 클릭 시 나오는 리뷰 팝업 window를 저장 후 변경

```
98          html = driver.page_source
99          bsObject = BeautifulSoup(html, "html.parser")
100         reviews = bsObject.find("ul", {"class": "list_detail_booklog"})
101         pages = bsObject.find("div", {"class": "list_paging"}).find("ul").find_all('li')
102         reviewNum = 1
103         for i in range(1, len(pages) + 1):
104             html = driver.page_source
```

- 98 : 현재 window의 html 소스를 저장
- 99 : BeautifulSoup를 이용하여 html 파싱
- 100 : reviews에 리뷰를 담고있는 ul 태그 저장
- 101 : 페이징 처리를 하기위한 li 저장
- 102 : 리뷰 순서 선언

```
103        for i in range(1, len(pages) + 1):
104            html = driver.page_source
105            bsObject = BeautifulSoup(html, "html.parser")
106            reviews = bsObject.find("ul", {"class": "list_detail_booklog"})
107            reviewslis = reviews.find_all('li')
108            for review in reviewslis:
109                reviewC = review.find("div", {"class": "content"})
110                if reviewC is None:
111                    continue
```

- 103 : 페이지 이동을 위한 반복문
- 104 : 페이지 이동시 html소스 불러오기
- 107 : 현재 페이지의 리뷰들 저장
- 109 : 리뷰의 내용을 저장

```
113                Path("./" + selectedBookName).mkdir(parents=True, exist_ok=True)
114                savedDir = "./" + selectedBookName + "/"
115                savedName = str(reviewNum) + ".txt"
```

- 113 ~ 115 : 현재 책 이름으로 리뷰 저장 폴더 생성 후 파일 저장을 위한 경로 저장

```
117            reviewContent = reviewC.get_text()
118            # 데이터 띄어쓰기,줄바꿈 제거
119            reviewContent = " ".join(reviewContent.split())
120            reviewContent = reviewContent.replace('>', '').replace('<', '').replace('*', '').replace('-',
121                                                                                                      '').replace(
122                '#', '').replace('^', '').replace('~', '')
```

- 117 ~ 112 : 리뷰 내용에 포함된 개행 문자, 특수 문자 제거

```
124                path = savedDir + savedName
125                with open(path, "w", encoding="cp949", errors='ignore') as f:
126                    f.write(reviewContent)
127                    print(reviewNum, "번째 리뷰가 저장되었습니다.")
128                f.close()
129                reviewNum = reviewNum + 1
```

- 124 : 경로 저장
- 125 ~ 128 : path 변수에 저장된 경로로 텍스트 파일 생성, 토픽 추출을 위해 인코딩 설정

```
131            if i < len(pages):
132                driver.find_element_by_link_text(str(i + 1)).click()
133            # 마지막 페이지 크롤링 후 종료
134            else:
135                break
```

- 131 ~ 135 : 다음 페이지로 이동

```
136             print("리뷰 추출이 완료되었습니다.")
137             openPath = "./"+selectedBookName
138             openPath = os.path.realpath(openPath)
139             os.startfile(openPath)
140             search = None
141             selectNum = None
142             status = False
143             checkend = True
144             savedDir = None
145             reviewExist = None
146             self.runCheck.emit(checkend) # True 보내면서 종료
147             driver.quit()
```

- 137 ~ 139 : 저장된 리뷰 폴더를 윈도우에서 열기
- 140 ~ 147 : 변수 초기화후 runCheck pyqtSignal 이벤트로 pyqtSlot에 checkend 전송

*** 메인**

```
200 if __name__ == '__main__':
201     app = QtWidgets.QApplication(sys.argv)
202     w = Form()
203     sys.exit(app.exec())
```

- 202 : Form class 객체 생성

2) 데이터 로드 모듈

- 웹 크롤링으로 수집한 리뷰 텍스트 파일을 R로 불러와 리스트로 저장

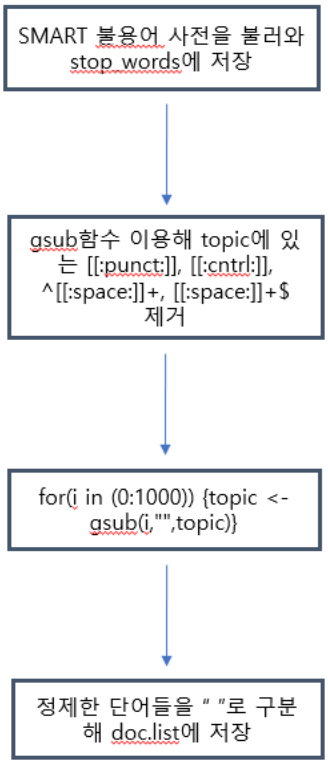
2-1) 데이터 로드 모듈 코드

```
path <- file.path("C:/Users/안윤빈/Desktop/4-1/종설1/gamsung")
kor <- list.files(file.path(path, "아몬드"))  #폴더 경로 중 en
kor.files <- file.path(path, "아몬드", kor)   #아까 list-up한
#all.files <- c(kor.files, eng.files) 다른 폴더에 있는 파일과
txt <- lapply(kor.files, readLines)    #txt에 이름을 붙여서 새
topic <- setNames(txt, kor.files)      # 텍스트 파일의 목록만들
topic <- sapply(topic, function(x) paste(x, collapse = " "))
```

- 코드 설명은 모듈 내용과 동일

3) 데이터 정제 모듈

- 불용어 처리 목록을 가져온 뒤, 리뷰를 저장한 리스트에서 단어를 추출하기 위해 데이터를 정제



3-1) 데이터 정제 모듈 코드

```
library(tm) # 텍스트 마이닝 패키지
stop_words <- stopwords("SMART") # 패키지에서 지원하는 불용어 목록
stop_words <- c(stopwords("SMART"), "")
# stopwords("SMART")

# pre-processing:
topic <- gsub("'", "", topic)  # remove apostrophes(')
topic <- gsub("[[:punct:]]", " ", topic)  # replace punctuation(구두점(!@#,.)) with space
topic <- gsub("[[:cntrl:]]", " ", topic)  # replace control characters(제어문자(\n, \t)) with space
topic <- gsub("^[[:space:]]+", "", topic) # remove whitespace at beginning of documents
topic <- gsub("[[:space:]]+$", "", topic) # remove whitespace at end of documents
for(i in (0:100)) {topic <- gsub(i,"",topic)}
for(i in (2000:2020)) {topic <- gsub(i,"",topic)}
topic <- gsub("lineheigh", "", topic)
topic <- gsub("setextparagraphalignlef", "", topic)
topic <- gsub("clas", "", topic)
topic <- gsub("styl", "", topic)
topic <- gsub("setextparagrap", "", topic)
topic <- gsub("있습니", "", topic)
topic <- gsub("malgu", "", topic)
topic <- gsub("gothi", "", topic)
topic <- gsub("합니", "", topic)
topic <- gsub("때문", "", topic)
doc.list <- strsplit(topic, "[[:space:]]+")
```

- 코드 설명은 모듈 내용과 동일

4) 자료구조 변환 모듈

- 리뷰에서 추출한 단어들을 출현 빈도수와 함께 테이블로 변환하여 저장

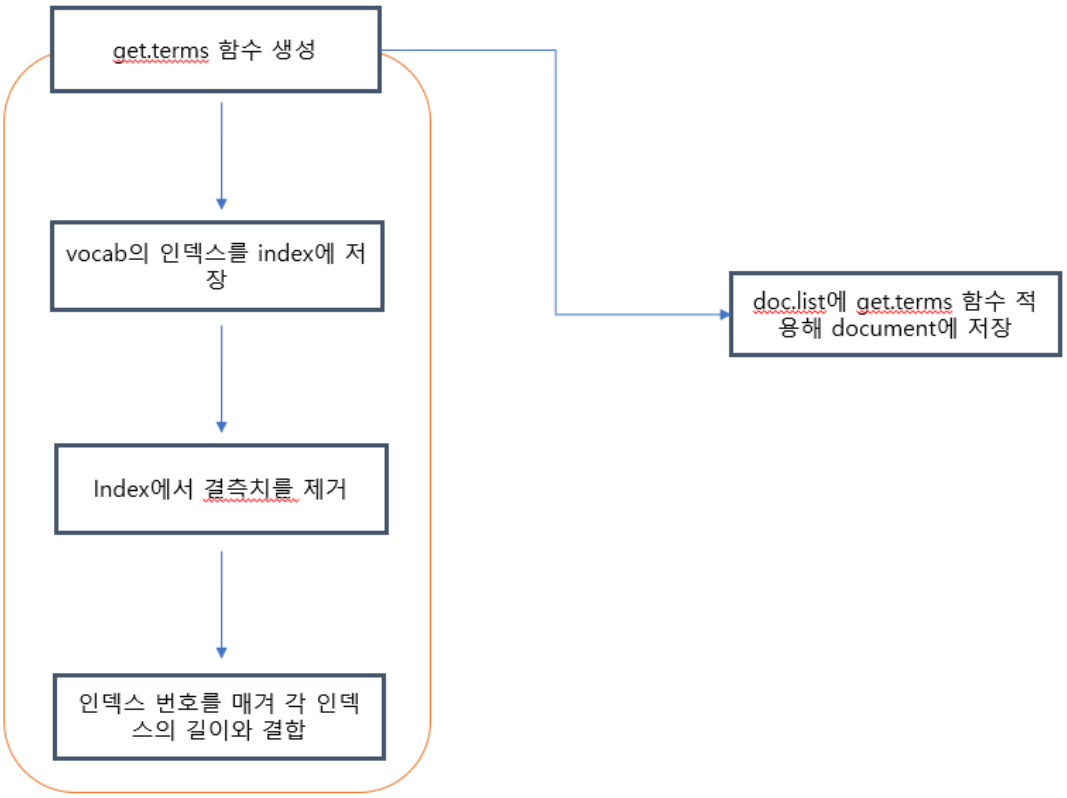
4-1) 자료구조 변환 모듈 코드

```
# 명사 추출
useSejongDic()  # 세종 사전 사용
doc.list <- sapply(doc.list, extractNoun, USE.NAMES=F)
doc.list <- unlist(doc.list)
doc.list <- Filter(function(x){nchar(x)>1}, doc.list)

# compute the table of terms:
# 저장한 용어를 테이블 형식으로 저장
term.table <- table(doc.list)
term.table <- sort(term.table, decreasing = TRUE)

# remove terms that are stop words or occur fewer than 5 times:
# 불용어 또는 5회 미만으로 언급된 단어들을 제거
del <- names(term.table) %in% stop_words | term.table < 5
term.table <- term.table[!del]
vocab <- names(term.table)
```

- 코드 설명은 모듈 내용과 동일

5) 모델 생성 모듈

- 데이터 분석을 위한 LDA 모델을 생성



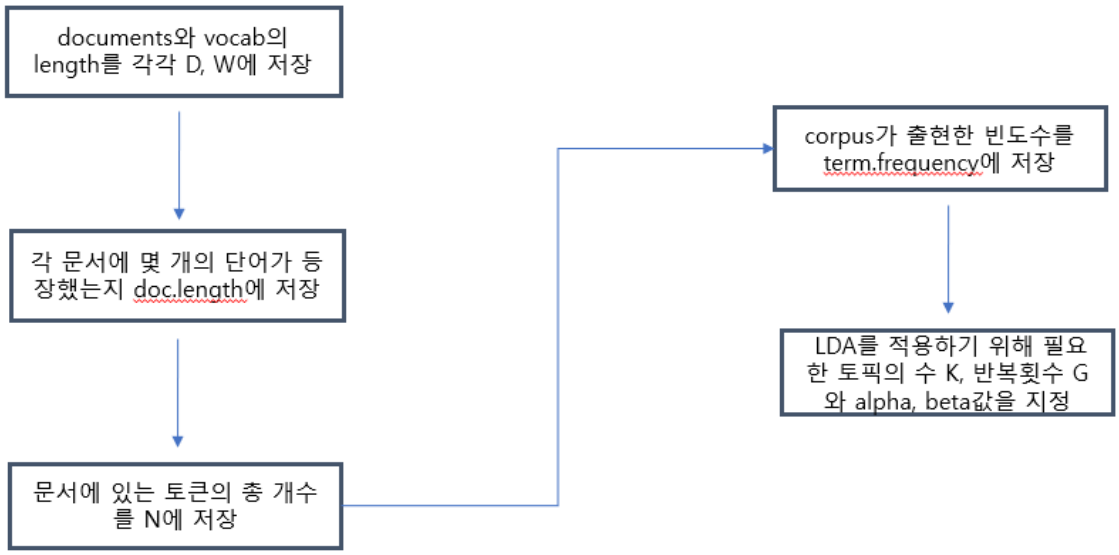
5-1) 모델 생성 모듈 코드

```
get.terms <- function(x) {
  index <- match(x, vocab)
  index <- index[!is.na(index)]
  rbind(as.integer(index-1), as.integer(rep(1, length(index))))
}
documents <- lapply(doc.list, get.terms) # 각 문서에 나오는 단어와 빈도수를 리스트 형식으로 저장
```

- 코드 설명은 모듈 내용과 동일

6) 통계 계산 모듈

- LDA 분석에 활용되는 통계를 계산하고, 변수를 지정



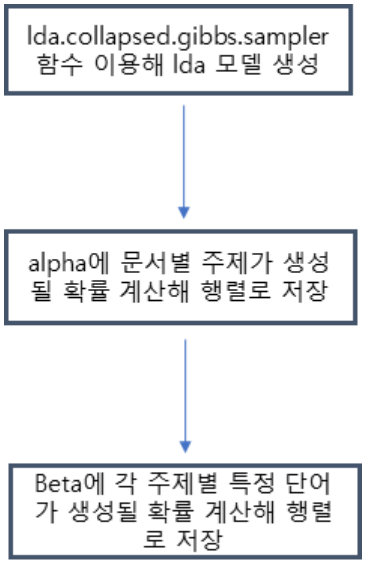
6-1) 통계 계산 모듈 코드

```
# 데이터셋과 관련된 일부 통계를 계산
D <- length(documents)  # number of documents (2,000)
W <- length(vocab)  # number of terms in the vocab (14,568)
doc.length <- sapply(documents, function(x) sum(x[2, ]))  # number of tokens(단어) per document [312, 288, 170
N <- sum(doc.length)  # total number of tokens in the data (546,827)

term.frequency <- as.integer(term.table)  # frequencies of terms in the corpus [8939, 5544, 2411, 2410, 2143,

# MCMC and model tuning parameters:
# MCMC 및 모델 튜닝 매개 변수
K <- 3  # 토픽의 개수 설정
G <- 5000 # 반복 횟수
alpha <- 0.02
eta <- 0.02
```

- 코드 설명은 모듈 내용과 동일

7) 분석 모듈

- R에 있는 LDA 라이브러리를 활용해 LDA 분석



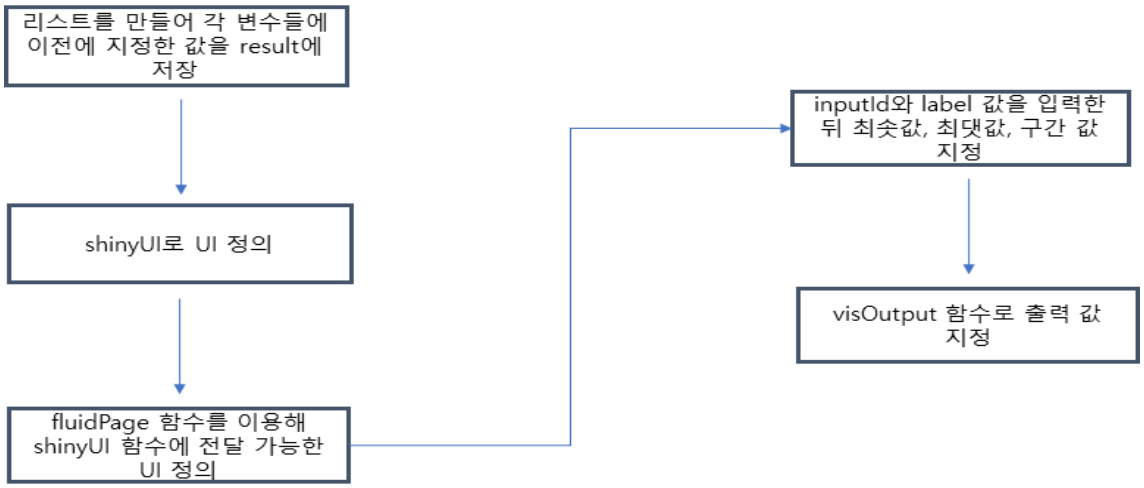
7-1) 분석 모듈 코드

```
library(lda)
library(topicmodels)
set.seed(357) # sampling
fit <- lda.collapsed.gibbs.sampler(documents = documents, K = K, vocab = vocab,
                                   num.iterations = G, alpha = alpha,
                                   eta = eta, initial = NULL, burnin = 0,
                                   compute.log.likelihood = TRUE)

theta <- t(apply(fit$document_sums + alpha, 2, function(x) x/sum(x)))
phi <- t(apply(t(fit$topics) + eta, 2, function(x) x/sum(x)))
```

- 코드 설명은 모듈 내용과 동일

8) 결과 전송 모듈

- LDA 결과를 웹에서 시각화하기 위한 유저 인터페이스를 구성



8-1) 결과 전송 모듈 코드

```
result <- list(phi = phi,
               theta = theta,
               doc.length = doc.length,
               vocab = vocab,
               term.frequency = term.frequency,encoding='UTF-8')

library(LDAvis)
options(encoding = 'UTF-8')
# create the JSON object to feed the visualization:
json <- createJSON(phi = result$phi,
                   theta = result$theta,
                   doc.length = result$doc.length,
                   vocab = result$vocab,
                   term.frequency = result$term.frequency,encoding='UTF-8')

serVis(json, out.dir = 'vis', open.browser = TRUE)

library(shiny)
library(LDAvis)

data(TwentyNewsgroups, package = "LDAvis")
ui <- shinyUI(
  fluidPage(tabsetPanel( #r샤이니를 이용하여 웹으로 결과 나타내는 부분
    tabPanel("topic", sliderInput("nTerms", "Number of terms to display", min = 20, max = 40, value = 30), visOutput('myChart') ),
    tabPanel("sentimental", "content")
  ))
)
```

- 코드 설명은 모듈 내용과 동일

9) 시각화 모듈

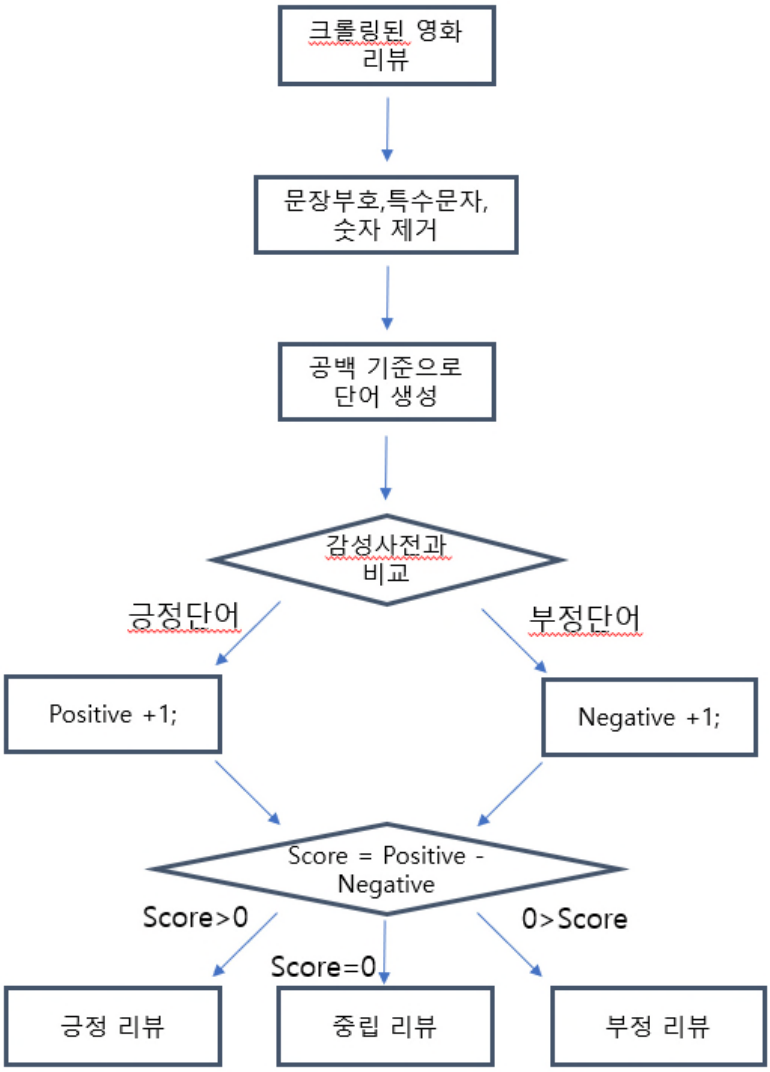
- 웹에서 시각화할 정보를 서버로 전송

9-1) 시각화 모듈 코드

```
server <- shinyServer(function(input, output, session) {
  output$myChart <- renderVis({
    if(!is.null(input$nTerms)){
      with(result,
           createJSON(phi = result$phi,
                      theta = result$theta,
                      doc.length = result$doc.length,
                      vocab = result$vocab,
                      term.frequency = result$term.frequency,encoding='UTF-8'))


    }
  })
})

shinyApp(ui = ui, server = server)
```

- 코드 설명은 모듈 내용과 동일

10) 감정분석 모듈



*** 감정분석**

감정분석이란 정형화되어있지 않은 텍스트에서 사람들의 태도, 의견 혹은 설향과 같은 정보를 얻어내는 것을 말한다. 감정분석을 하려면 우선 감정사전이 필요한데, 긍정어사전과 부정어사전 두가지 사전이 필요하다. 각각의 사전은 txt파일로 긍정어와 부정어가 종류별로 나열되어 있는 형태이다. 감정분석을 시행하면 분석한 문서에 있는 단어들과 감정사전을 대조하여 문서속 긍정어가 몇개인지, 부정어가 몇개인지를 찾아낸다. 그 후, 문서속 긍정어의 개수에서 부정어의 개수를 뺐을 때, 양수가 나온다면 긍정적인 문서, 음수가 나온다면 부정적인 문서, 0이 나온다면 중립적인 의견을 가진 문서로 판별하게 된다.

* 감정분석에 필요한 패키지

```
#install.packages('plyr')
#install.packages('stringr')
#install.packages('shiny')
#install.packages('ggplot2')
#install.packages('showtext')
rm(list=ls())
```

* 감정분석에 필요한 라이브러리 호출

```
library(plyr)
library(stringr)
library(shiny)
library(ggplot2)
```

* 인코딩 설정(사용중인 OS마다 다를 수 있음)

```
Sys.setlocale(category = "LC_CTYPE", locale = "ko_KR.UTF-8")
```

* 웹 크롤링의 결과로 크롤링된 리뷰 불러오기

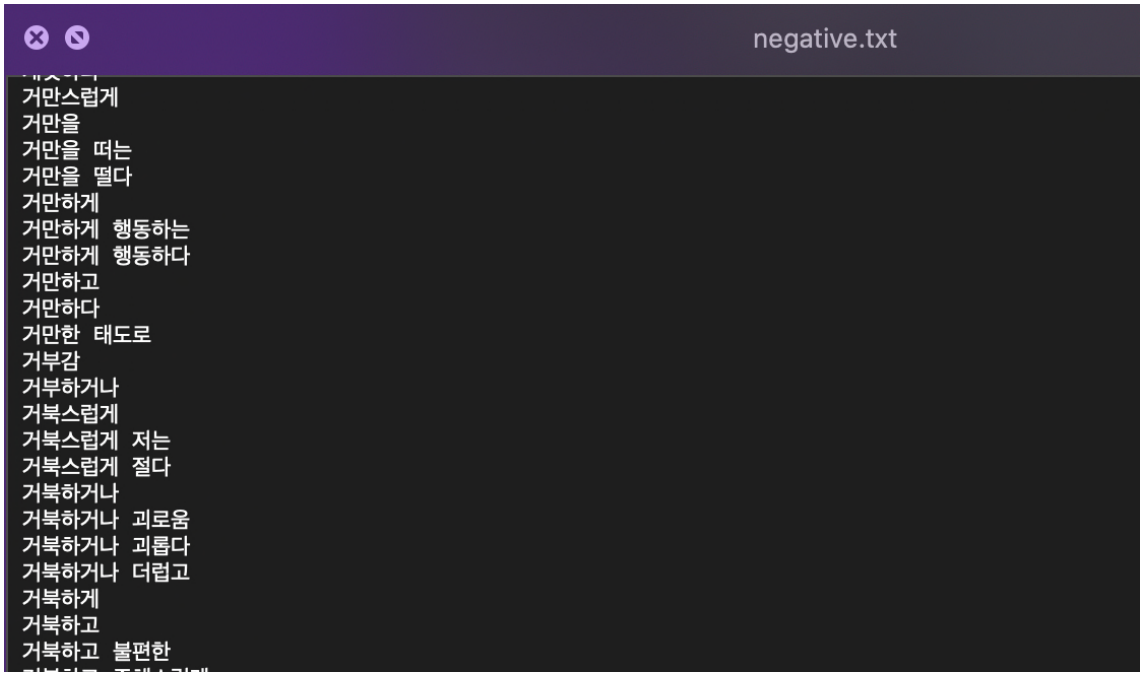
```
setwd("/Users/welcomeonboardboy/Documents/gamsung") #불러들일경로 (negative,positive.txt여기에있어야함)
#txt<-readLines("15.txt",warn=FALSE) #감정분석할 텍스트파일 불러오기

path <- file.path("/Users/welcomeonboardboy/Documents/gamsung")  # 폴더 경로를 객체로 만든다
kor <- list.files(file.path(path, "스스로 행복하라"))  #폴더 경로 중 eng라는 폴더에 있는 파일들 이름을 list-up해서 객체로 만든다.
kor.files <- file.path(path, "스스로 행복하라", kor)   #아까 list-up한 파일의 이름들로 폴더의 경로를 다시 객체로 만든다.
#all.files <- c(kor.files, eng.files) 다른 폴더에 있는 파일과 같이 돌릴때 사용
txt <- lapply(kor.files, readLines)    #txt에 이름을 붙여서 새 객체 생성됨
topic <- setNames(txt, kor.files)      # 텍스트 파일의 목록만큼 데이터를 읽어오고, 그 결과를 list 형태로 저장
topic <- sapply(topic, function(x) paste(x, collapse = " "))  #topic에 있는 리스트들을 공백을 두고 이어붙임
```

* 긍정어, 부정어 사전 불러오기

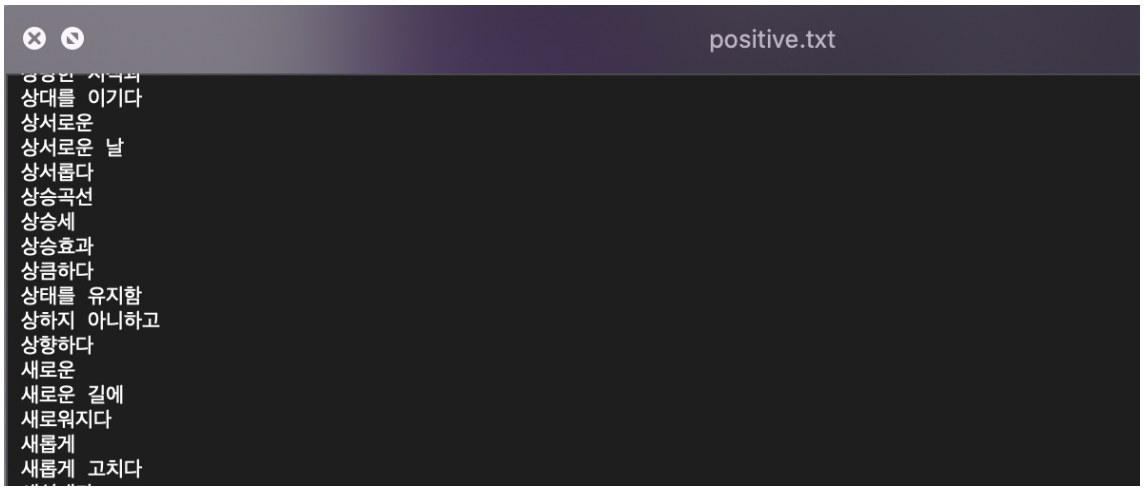
```
positive <- readLines("positive.txt")
positive=positive[-1]
negative <- readLines("negative.txt")
negative=negative[-1]
```

- 군산대 부정어 사전을 이용했다.

- 부정어 사전 일부 캡쳐 (가나다 순서로 부정적인 의미를 지닌 표현들이 나열되어있다.)



- 부정어 사전 일부 캡쳐 (가나다 순서로 부정적인 의미를 지닌 표현들이 나열되어있다.)



* 감정분석

```
sentimental = function(sentences, positive, negative){

  scores = laply(sentences, function(sentence, positive, negative) {

    sentence = gsub('[[:punct:]]', '', sentence) # 문장부호 제거
    sentence = gsub('[[:cntrl:]]', '', sentence) # 특수문자 제거
    sentence = gsub('\\d+', '', sentence)        # 숫자 제거

    word.list = str_split(sentence, '\\s+')      # 공백 기준으로 단어 생성 -> \\s+ : 공백 정규식,
+(1개 이상)
    words = unlist(word.list)                    # unlist() : list를 vector 객체로 구조변경

    pos.matches = match(words, positive)           # words의 단어를 positive에서 matching
    neg.matches = match(words, negative)

    pos.matches = !is.na(pos.matches)            # NA 제거, 위치(숫자)만 추출
    neg.matches = !is.na(neg.matches)

    score = sum(pos.matches) - sum(neg.matches)  # 긍정 - 부정
    return(score)
  }, positive, negative)

  scores.df = data.frame(score=scores, text=sentences)
  return(scores.df)
}
```

- 감정분석의 첫번째 스텝은 크롤링된 리뷰 속 문장부호, 특수문자 그리고 숫자를 제거하는 것이다. 감정사전과의 단어를 비교할 때 최대한 정확한 결과를 얻기 위해서이다. 또, 인터넷 문화의 특성상 맞춤법이 잘 지켜지지 않기 때문이기도 하다.

- 문장부호, 특수문자, 숫자를 제거한 후에는 공백을 기준으로 단어를 나눈다.

- 나뉘어진 단어 하나하나를 긍정어사전과 부정어사전과 대조하여 리뷰속 긍정어와 부정어의 갯수를 알아낸다.

- 한 리뷰 속 (긍정어의 수 - 부정어의 수) 의 값이 양수이면 긍정적인 리뷰, 음수이면 부정적인 리뷰, 0이면 중립적인 리뷰로 판단하게 된다.

* 결과를 파이차트로 표현

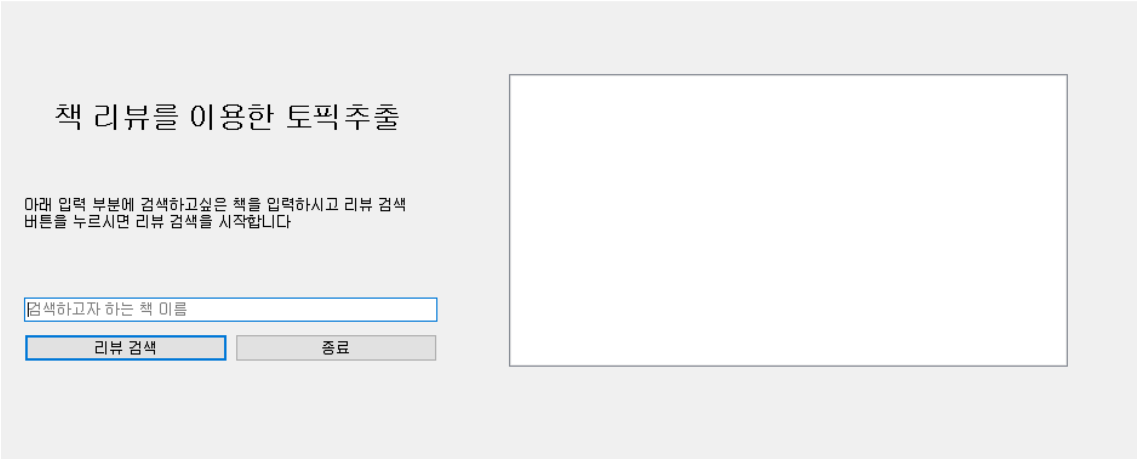
```
result=sentimental(topic, positive, negative)

result$color[result$score >=1] = "Olive Drab 2"
result$color[result$score ==0] = "Gray 60"
result$color[result$score < 0] = "Orange Red 2"

result$remark[result$score >=1] = "긍정"
result$remark[result$score ==0] = "중립"
result$remark[result$score < 0] = "부정"

sentiment_result= table(result$remark)
pie(sentiment_result, main="감성분석 결과",col=c("Olive Drab 2","Orange Red 2","Gray 60"), radius
=0.8) #결과 파이그래프로 나타냄
sentiment_result #결과 숫자로 확인
```

* 결과를 rShiny를 이용하여 웹 시각화

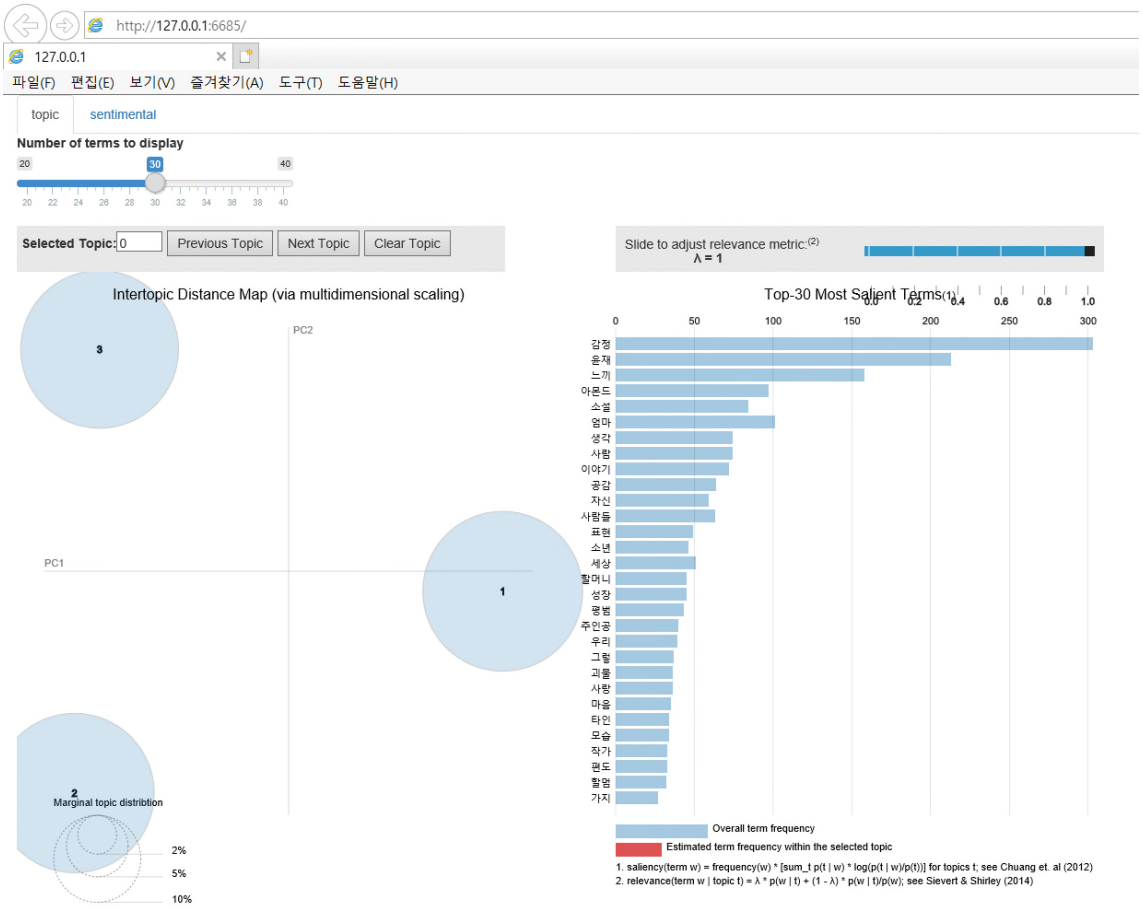
```
ui <- fluidPage( tabsetPanel( #r샤이니를 이용하여 웹으로 결과 나타내는 부분
  tabPanel("topic","content"),
  tabPanel("sentimental", mainPanel(plotOutput(outputId = "sentiment_result")))
))

server <- function(input,output){
  output$sentiment_result <- renderPlot({pie(sentiment_result, main="감정분석 결과",
                                             col=c("Olive Drab 2","Orange Red 2","Gray 60"), radius=0.8)})
}
shinyApp(ui=ui,server=server)
```

5. Prototype 구현

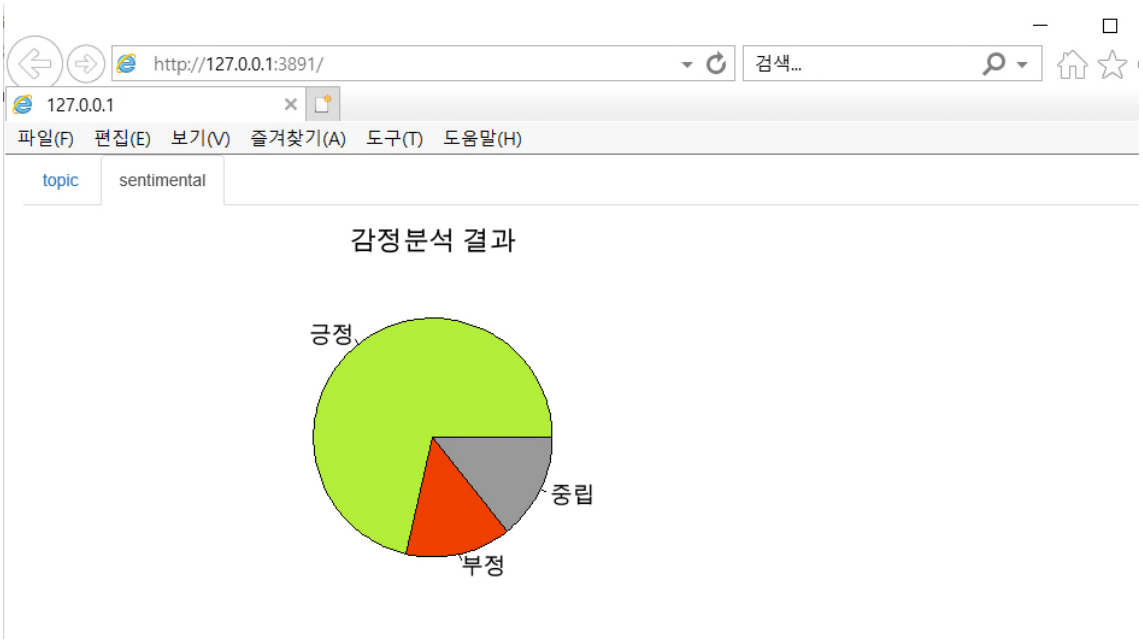
* 웹 크롤링 GUI 화면



* 토픽 추출 결과 화면



* 감정분석 결과 웹 시각화


http://127.0.0.1:3891/
검색...
127.0.0.1
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
topic
sentimental
감정분석 결과
긍정
중립
부정

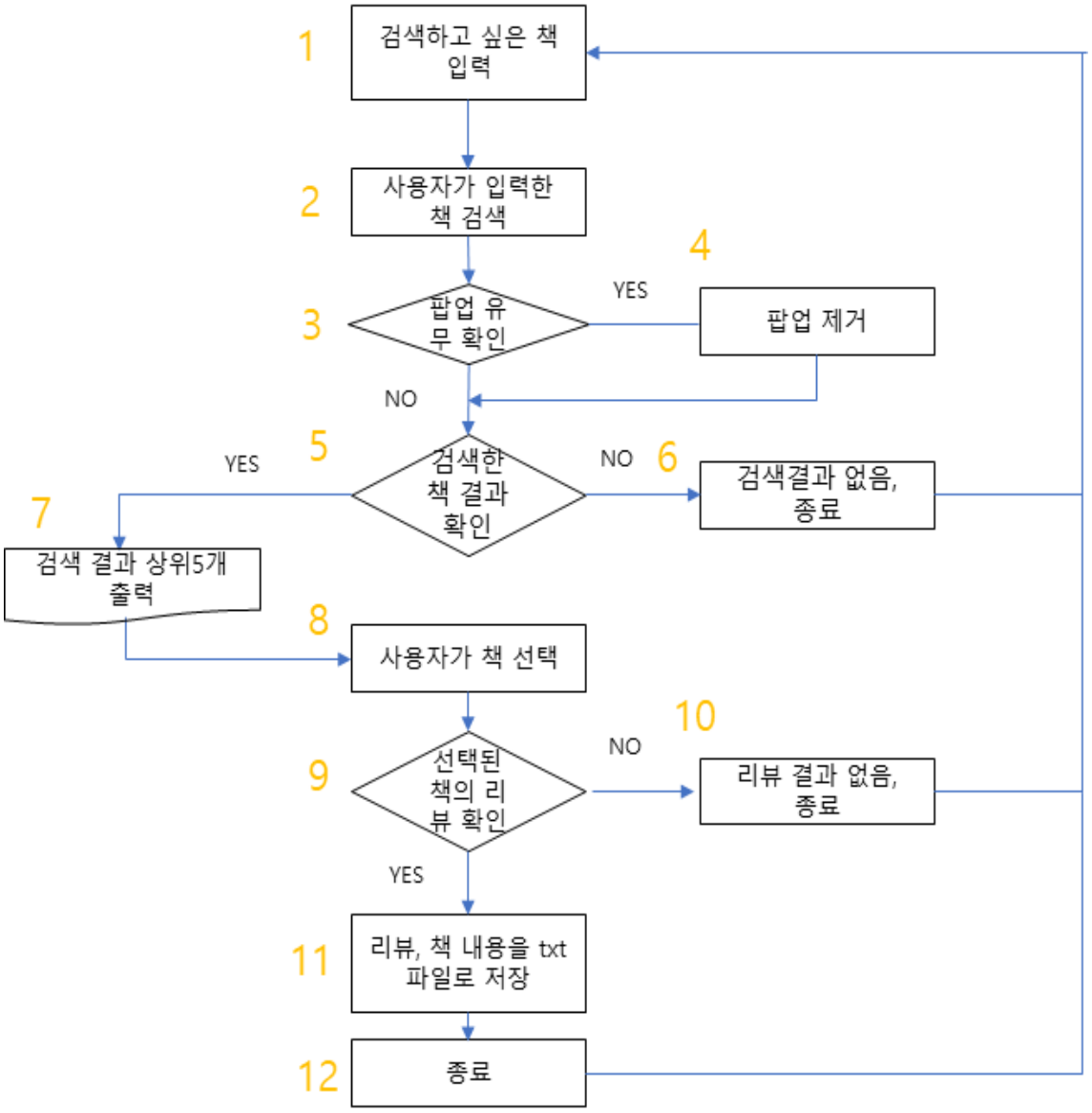
- 기능 구현 사항

	완성	진행중	미완성
웹 크롤링	- UI, 이벤트 처리(2020/02/02) - 크롤링 자동화(2020/02/28) - 쓰레드(2020/03/22)		
토픽 추출	- LDA 알고리즘(2019/12/05) - 웹 시각화(2020/01/20) - 문서 전처리 보완 (2020/06/09) - 웹 시각화 UI 개선 (2020/06/12)		
감정 분석	- 크롤링된 결과 감정분석 (2020/03/30) - 감정분석 결과 파이차트로 표현(2020/04/05) - 감정분석 결과 웹 시각화 (2020/04/21) - 감정분석 그래프 수정 (2020/06/11) - 트리맵 그래프 추가 (2020/06/12)		

6. 시험/ 테스트 결과

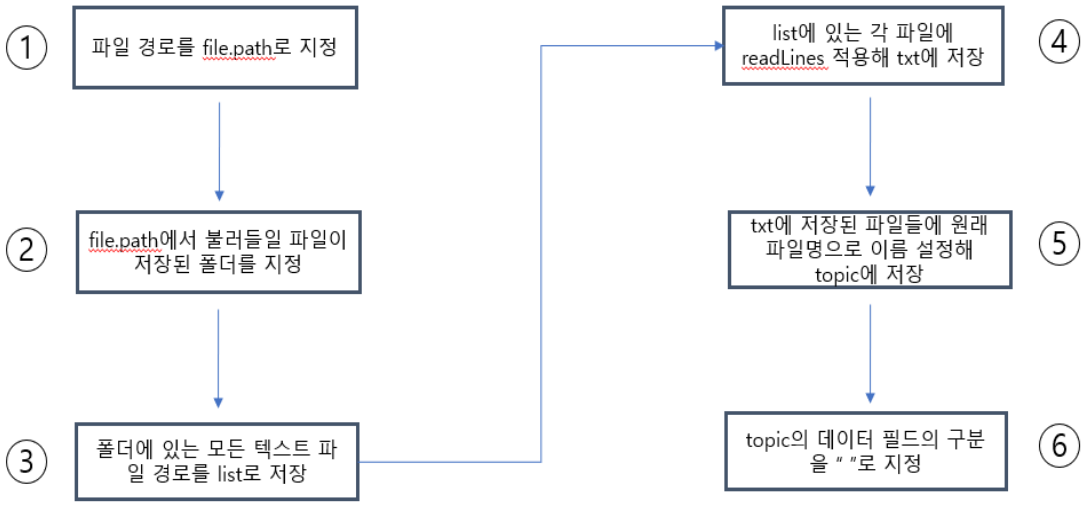
- 화이트 박스 테스트 기법을 사용

1) 웹 크롤링 모듈



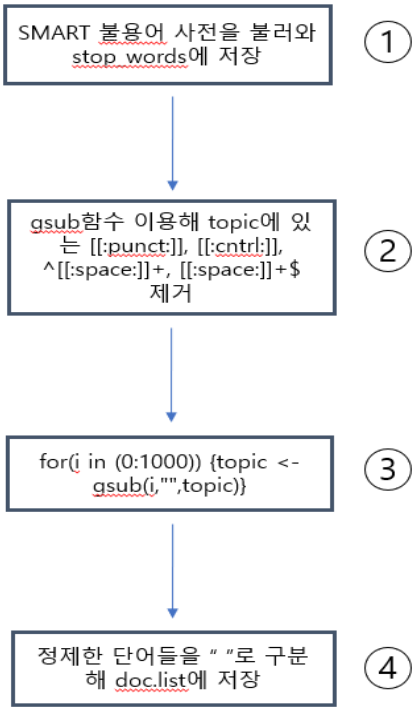
경로	순서	동작 여부
경로 1	1-2-3-4-5-7-8-9-11-12	O
경로 2	1-2-3-5-7-8-9-11-12	O
경로 3	1-2-3-4-5-6	O
경로 4	1-2-3-4-5-7-8-9-10	O

2) 데이터 로드 모듈



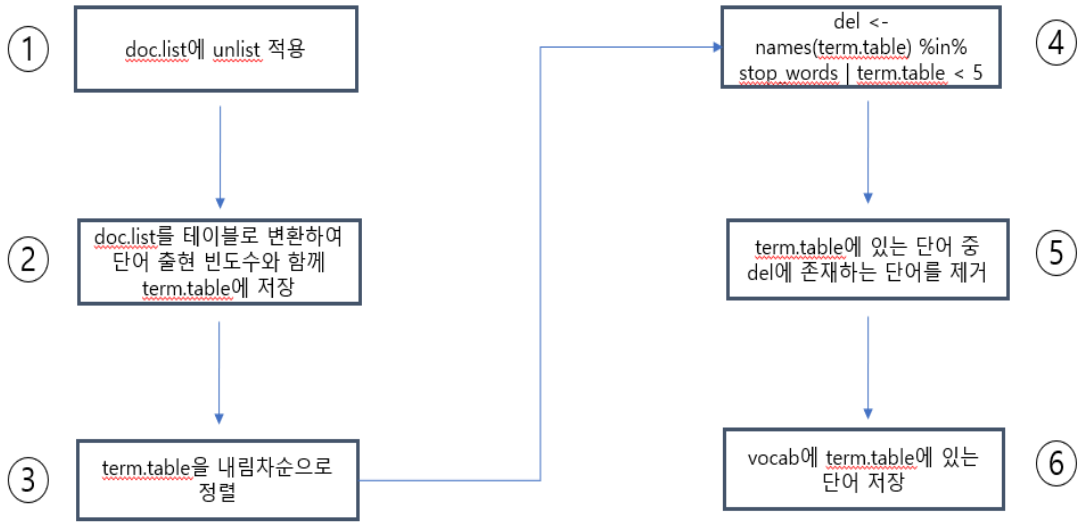
순서	동작 여부
1-2-3-4-5-6	O

3) 데이터 정제 모듈



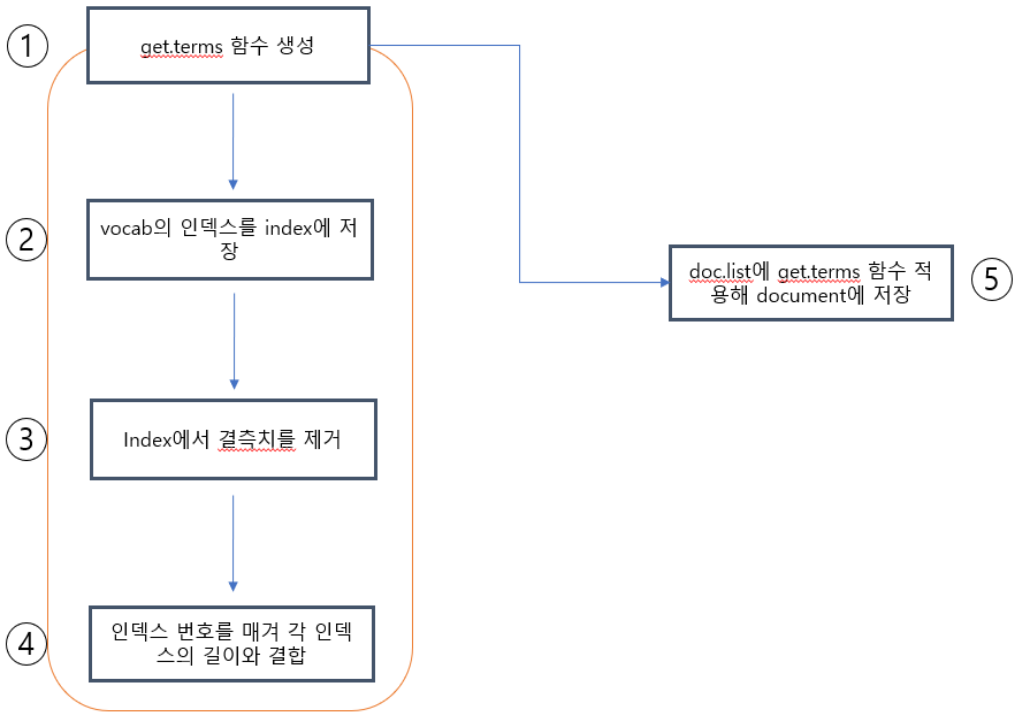
순서	동작 여부
1-2-3-4	O

4) 자료구조 변환 모듈



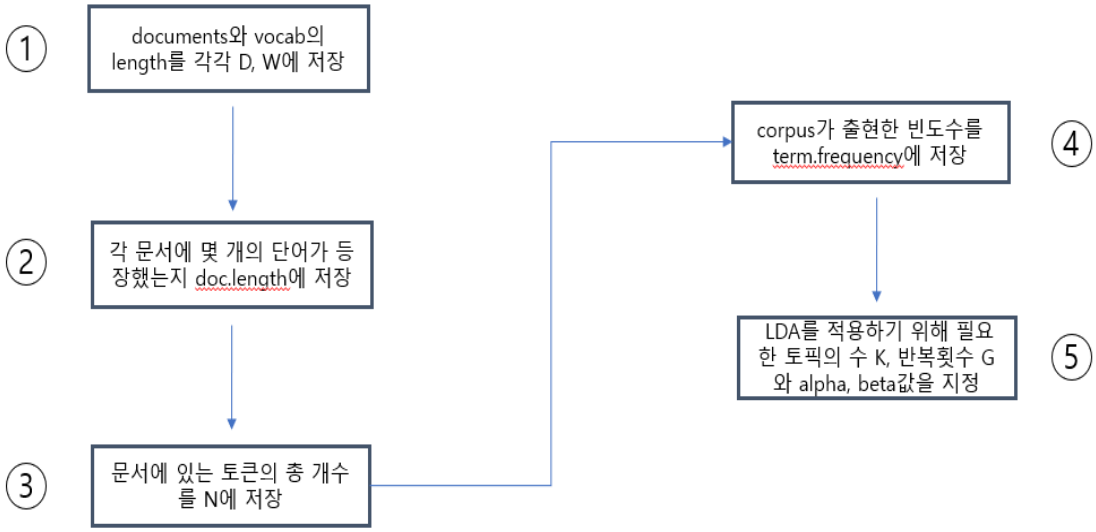
순서	동작 여부
1-2-3-4-5-6	O

5) 모델 생성 모듈



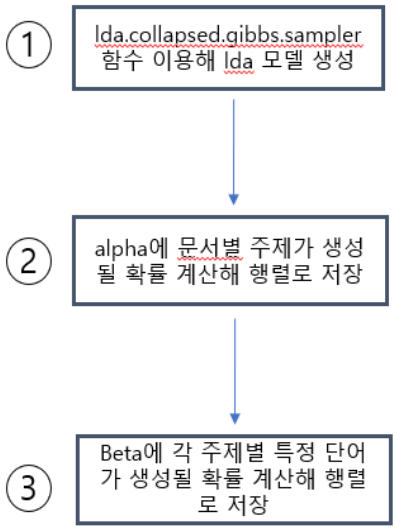
순서	동작 여부
1-2-3-4-5	O

6) 통계 계산 모듈



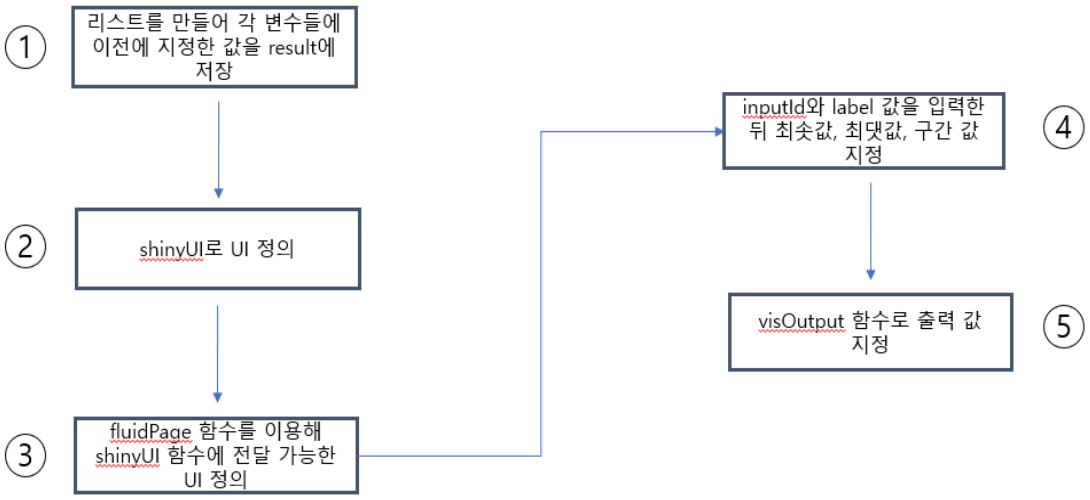
순서	동작 여부
1-2-3-4-5	O

7) 분석 모듈



순서	동작 여부
1-2-3	O

8) 결과 전송 모듈

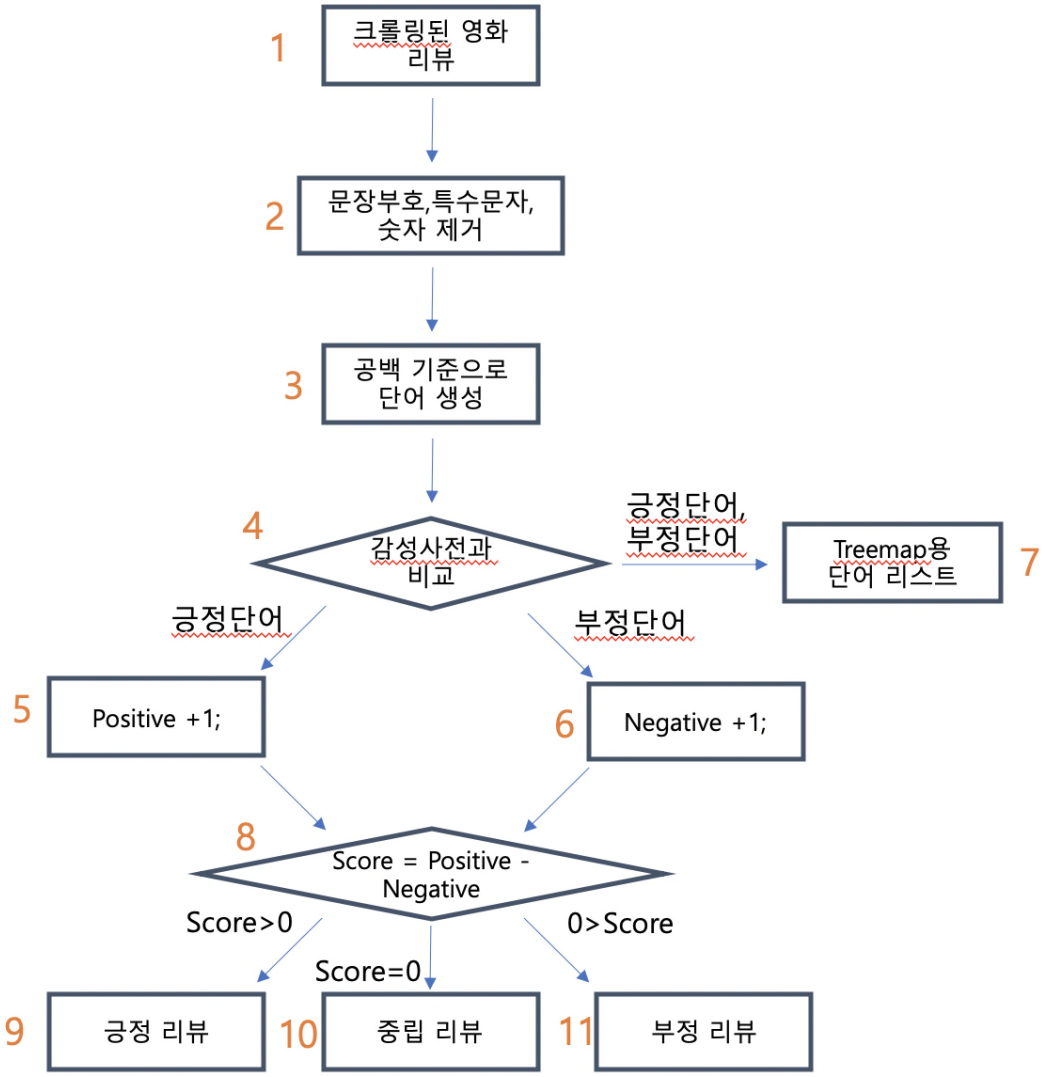


순서	동작 여부
1-2-3-4-5	O

9) 시각화 모듈

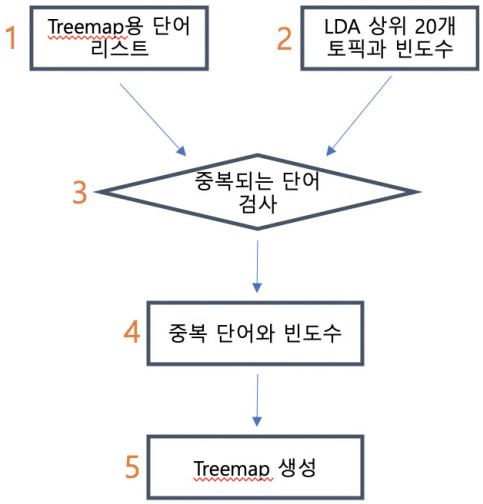
순서	동작 여부
1-2-3-4-5	O

10) 감정분석 모듈



경로	순서	동작 여부
경로 1	1-2-3-4-7	O
경로 2	1-2-3-4-5-8-9	O
경로 3	1-2-3-4-5-8-10	O
경로 4	1-2-3-4-5-8-11	O
경로 5	1-2-3-4-6-8-9	O
경로 6	1-2-3-4-6-8-10	O
경로 7	1-2-3-4-6-8-11	O

11.Treemap 모듈



경로	순서	동작여부
경로 1	1-3-4-5	O
경로 2	2-3-4-5	O

7. Coding & Demo

1) 웹크롤링

1)-1 코드

```
14  selectNum = None # 책 선택 인덱스
15  status = False # 웹 크롤링 시작 상태 True : 진행중 , False : 꺼짐
16  bookContent =None
17  class Worker(QThread):
18      finished = pyqtSignal(list)
19      runCheck = pyqtSignal(bool) # 크롤링 후 팝업 띄우기용
20      reviewCheck = pyqtSignal(bool) # 리뷰 유무 확인 True : review 있음, False : 없음
21      def run(self):
22          global status
23          global search
24          global selectNum
25          global bookContent
26          checkend = False # False : 진행중 , True: 종료됨
27          reviewExist = None # True : review 있음, False : 없음
28          path = "C:/dev/chromedriver.exe"
29          print(path)
30          driver = webdriver.Chrome(path)
31          driver.implicitly_wait(5)
32          driver.get('http://www.kyobobook.co.kr/index.laf')
33          time.sleep(3)
34          driver.implicitly_wait(3)
```

1~34 : 웹 크롤링위한 변수 설정과 초기 세팅

```
35          try:
36              # if driver.window_handles[1]:
37              driver.switch_to.window(driver.window_handles[1])
38              print("팝업 제거.")
39              driver.close()
40              driver.switch_to.window(driver.window_handles[0])
41          except Exception as error :
42              print(error)
43              print("현재 팝업이 없음")
44          driver.find_element_by_xpath("//*[@id='searchKeyword']").send_keys(search)
45          driver.find_element_by_xpath("/html/body/div[4]/div[1]/div[1]/div[1]/form[2]/div/input").click()
46          books = [] # 검색한 책 리스트
47          driver.implicitly_wait(3)
```

35~47 : 팝업제거 후 웹 크롤링 시작

```
48          try:
49              bookExist = driver.find_element_by_xpath("//*[@id='search_list']/tr[1]/td[2]/div[2]/a")
50              print(bookExist.text)
51              # 검색한 책 찾았을 경우
52              for i in range(1, 6):
53                  try:
54                      bookName = driver.find_element_by_xpath(
55                          "//*[@id='search_list']/tr[" + str(i) + "]/td[2]/div[2]/a/strong").text
56                      print(str(i) + ".", bookName)
57                      books.append(bookName)
58                  # 책 이름이 없을 시 반복문 종료
59                  except NoSuchElementException:
60                      break
61              self.finished.emit(books) # 검색 완료 후 결과 전송(검색 결과 있음)
62
63              # 사용자가 책을 선택할 때 까지 대기
64              while True :
65                  if selectNum is None:
66                      pass
67                  else : break
68          except NoSuchElementException:
69              # 검색한 책 찾지 못했을 경우
70              self.finished.emit(books) # 검색 완료 후 결과 전송(검색 결과 없음)
71              search = None
72              selectNum = None
73              status = False
74              checkend = True
75              driver.quit()
76              # self.quit()
77              return None
```

48~77 : 책 검색 기능 . 결과 있을 시 결과 ui에 출력 후 사용자 선택 대기. 결과 없을 시 종료

```
78          selectedBook = driver.find_element_by_xpath("//*[@id='search_list']/tr[" + str(selectNum) + "]/td[2]/div[2]/a")
79          selectedBookName = driver.find_element_by_xpath("//*[@id='search_list']/tr[" + str(selectNum) + "]/td[2]/div[2]/a/strong").text
80          selectedBook.click()
81          window_before = driver.window_handles[0]
82          try:
83              showReview = driver.find_element_by_link_text("전체보기")
84              print("검색하신 책의 리뷰를 찾았습니다.")
85              # bookContent = driver.find_element_by_xpath("//*[@id='container']/div[5]/div[1]/div[3]/div[3]").text
86              bookContent = driver.find_elements_by_class_name("box_detail_article")[0].text
87              print(bookContent)
88              reviewExist = True
89          except NoSuchElementException:
90              print("검색하신 책의 리뷰가 없습니다. 종료합니다")
91              reviewExist = False
92              self.reviewCheck.emit(reviewExist)
93              driver.quit()
94              reviewExist = None
95              search = None
96              selectNum = None
97              status = False
98              checkend = True
99              return None
```

78~99 : 사용자가 선택한 책으로 페이지를 이동해 리뷰 유무 확인, 책 내용 저장.

```
100         showReview.click()
101         window_after = driver.window_handles[1]
102         driver.switch_to.window(window_after)
103 
104         print("현재윈도우 URL : ", driver.current_url)
105         print("변경전윈도우: ", str(window_before))
106         print("변경후윈도우: ", str(window_after))
107 
108         html = driver.page_source
109         bsObject = BeautifulSoup(html, "html.parser")
110         reviews = bsObject.find("ul", {"class": "list_detail_booklog"})
111         pages = bsObject.find("div", {"class": "list_paging"}).find("ul").find_all('li')
112         reviewNum = 1
113         for i in range(1, len(pages) + 1):
114             html = driver.page_source
115             bsObject = BeautifulSoup(html, "html.parser")
116             reviews = bsObject.find("ul", {"class": "list_detail_booklog"})
117             reviewslis = reviews.find_all('li')
118             for review in reviewslis:
119                 reviewC = review.find("div", {"class": "content"})
120                 if reviewC is None:
121                     continue
122                 # savedDir = "./reviewF/"
123                 Path("./" + selectedBookName).mkdir(parents=True, exist_ok=True)
124                 savedDir = "./" + selectedBookName + "/"
125                 savedName = str(reviewNum) + ".txt"
126                 # print(type(reviewC))
127                 reviewContent = reviewC.get_text()
128                 # 데이터 띄어쓰기,줄바꿈 제거
129                 reviewContent = " ".join(reviewContent.split())
130                 reviewContent = reviewContent.replace('>', '').replace('<', '').replace('*', '').replace('-',
131                                                                                                          '').replace(
132                     '#', '').replace('^', '').replace('~', '')
133                 # reviewContent = reviewContent.splitlines()
134                 path = savedDir + savedName
135                 with open(path, "w", encoding="cp949", errors='ignore') as f:
136                     f.write(reviewContent)
137                     print(reviewNum, "번째 리뷰가 저장되었습니다.")
138                 f.close()
139                 reviewNum = reviewNum + 1
140             # 다음페이지로 이동
141             if i < len(pages):
142                 driver.find_element_by_link_text(str(i + 1)).click()
143             # 마지막 페이지 크롤링 후 종료
144             else:
145                 break
146         print("리뷰 추출이 완료되었습니다.")
147         with open(savedDir+"content.txt","w",encoding="cp949",errors='ignore') as f:
148             f.write(bookContent)
149         f.close()
```

100~149 : 리뷰, 책 내용 크롤링 후 txt파일로 저장

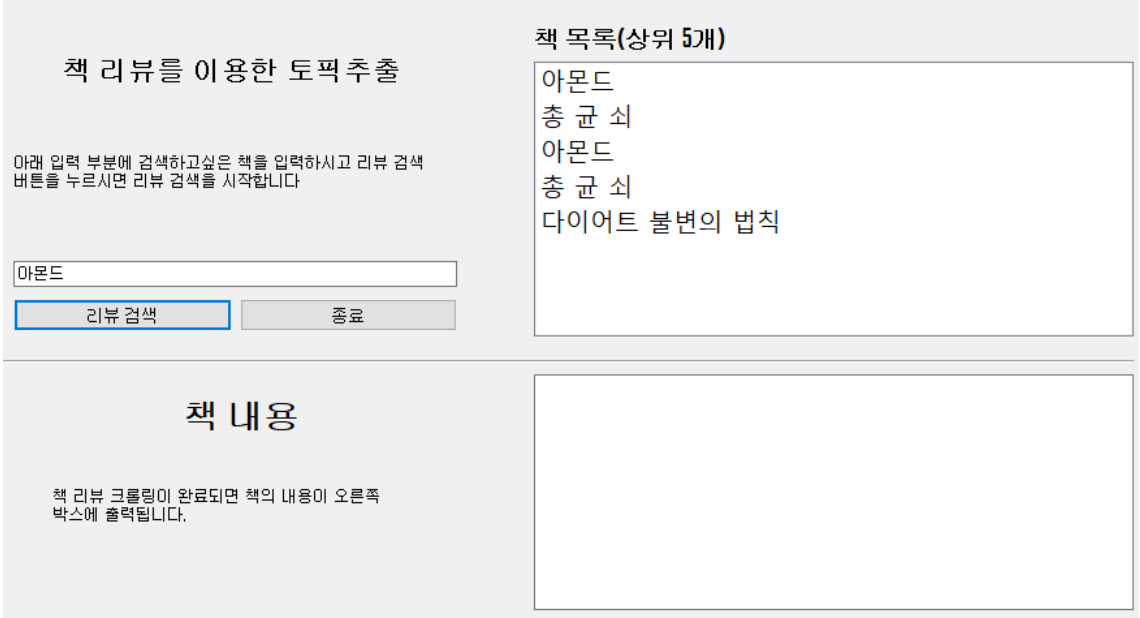
```
161 class Form(QtWidgets.QDialog):
162     def __init__(self,parent=None):
163         super().__init__()
164         self.ui = uic.loadUi("gui.ui")
165         self.ui.search_btn.clicked.connect(self.searchBook)
166         self.ui.quit_btn.clicked.connect(self.quitGUI)
167         self.ui.bookList.itemDoubleClicked.connect(self.select_book)
168         self.ui.show()
169     def quitGUI(self):
170         QtWidgets.QMessageBox.about(self, 'Message', '프로그램을 종료합니다')
171         exit()
172     def searchBook(self):
173         global search
174         global status
175         if status is False:
176             status = True
177             self.worker = Worker()
178             search = self.ui.search_name.text()
179             self.worker.finished.connect(self.update_list)
180             self.worker.runCheck.connect(self.checkEnd)
181             self.worker.reviewCheck.connect(self.checkReview)
182             self.worker.start() # 쓰레드 시작
183     def select_book(self):
184         global selectNum
185         # QtWidgets.QMessageBox.about(self, 'Message', self.ui.bookList.currentItem().text())
186         selectNum = self.ui.bookList.currentRow()+1
187         # print(self.ui.bookList.currentRow())
188         print(self.ui.bookList.currentItem().text())

190     @pyqtSlot(list)
191     def update_list(self, data):
192         # data에 books에 있는 책 리스트가 넘어옴
193         # 만약 책 리스트가 비어있을 시 팝업 출력
194         if len(data) == 0:
195             QtWidgets.QMessageBox.about(self,'Message','검색 결과 없음, 다시 검색하세요  ')
196         # 책 리스트가 있을 시 bookList에 값 추가
197         else:
198             # 책 이름으로 추가
199             for bookName in data:
200                 self.ui.bookList.addItem(str(bookName))
201
202     # 리뷰 하나끝나면 팝업띄우기
203     @pyqtSlot(bool)
204     def checkEnd(self,data):
205         QtWidgets.QMessageBox.about(self,'Message','리뷰 추출이 완료되었습니다.')
206         self.ui.bookContent.setText(bookContent)
207         self.ui.bookList.clear()
208
209     @pyqtSlot(bool)
210     def checkReview(self,data):
211         if data == False:
212             QtWidgets.QMessageBox.about(self,'Message','리뷰가 없습니다. 종료하겠습니다.')
213             self.ui.bookList.clear()
```

161~213 : GUI 불러오기, 이벤트처리

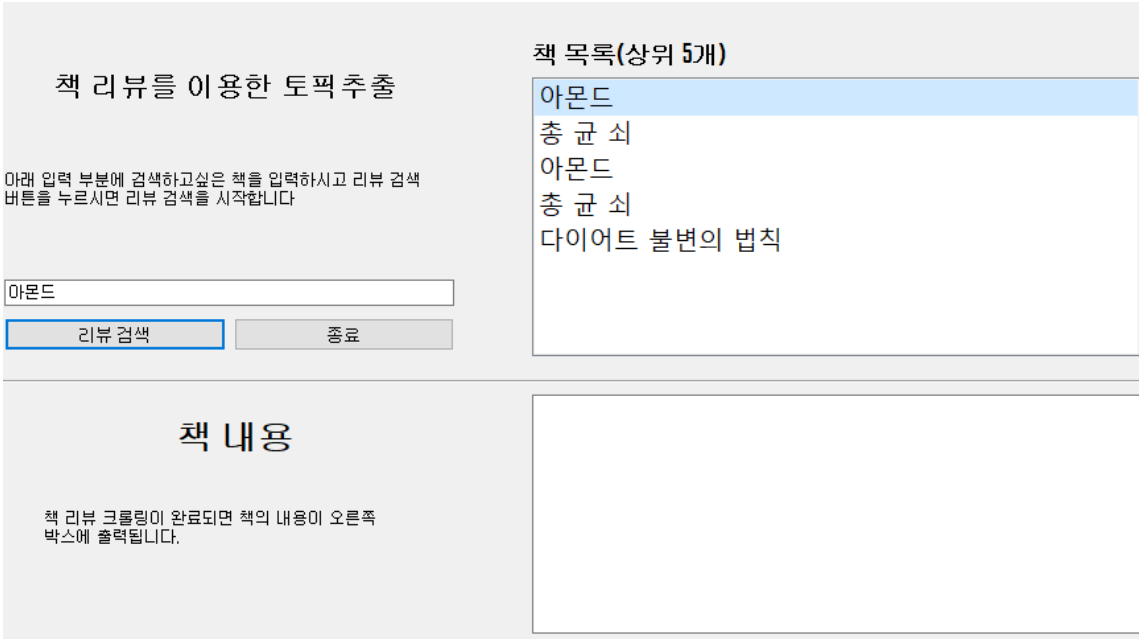
1)-2 데모

- 사용자가 검색하고 싶은 책을 입력하고 '리뷰 검색' 버튼을 누름

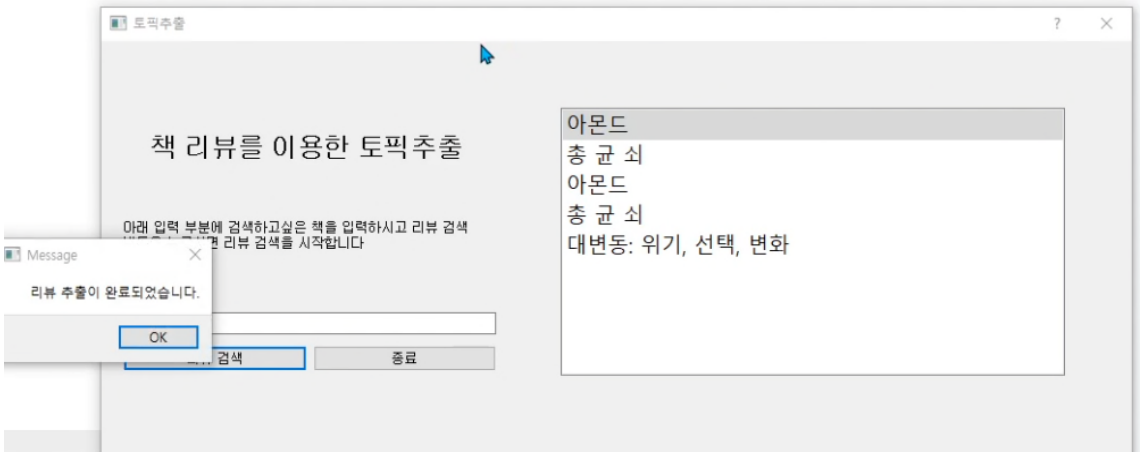
- webdriver가 교보문고 홈페이지로 이동하여 입력받은 책을 selenium을 이용해 자동으로 검색



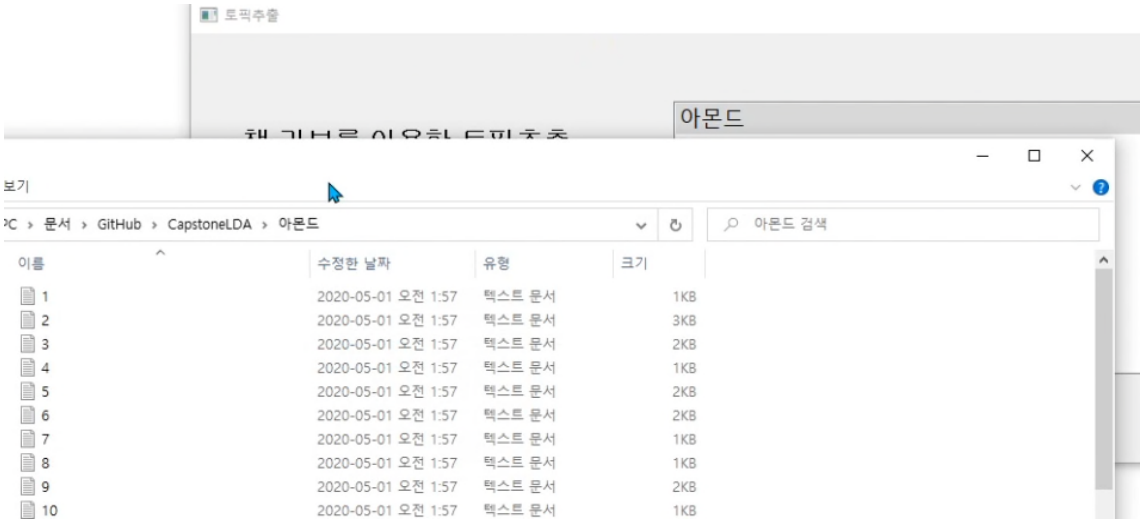
- 검색된 책 목록 상위 5개가 GUI에 출력되고 사용자가 GUI에서 더블 클릭한 책의 리뷰 크롤링을 시작

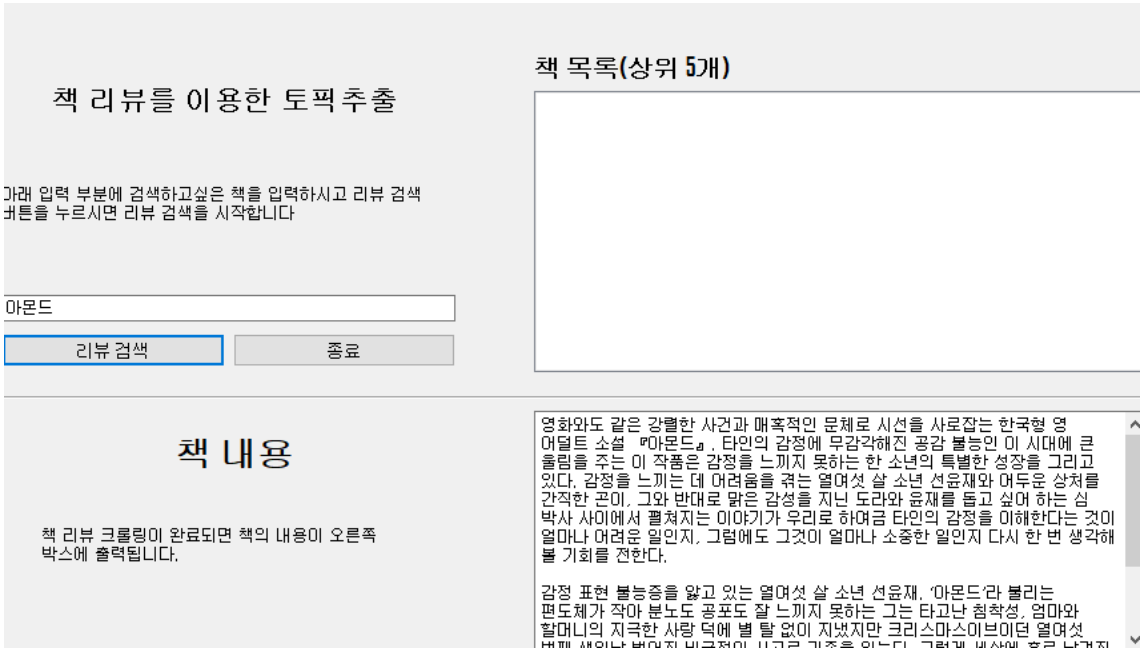


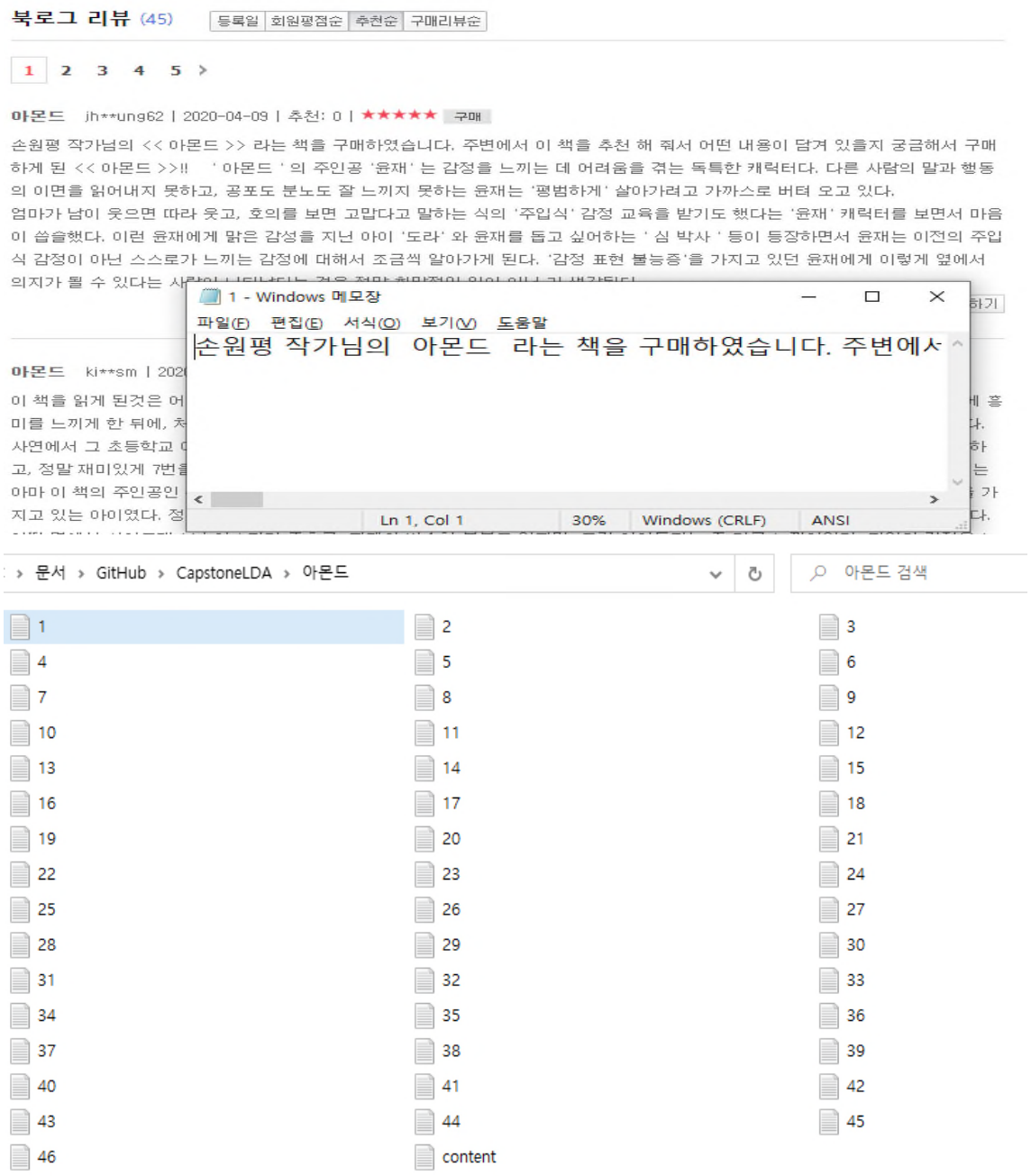
- 북로그 리뷰에 있는 모든 리뷰 내용을 BeautifulSoup를 이용하여 크롤링



- 리뷰 크롤링이 완료되면 자동으로 웹 브라우저를 종료







- 책 리뷰와 내용이 추출되어 txt파일로 저장된 것을 확인 가능 (사용자가 설정한 토픽 추출을 위한 인코딩 방식(ANSI)도 확인가능)

2) 토픽 추출

2)-1. 코드

```
rm(list=ls())
library(plyr)
library(stringr)
library(shiny)
library(ggplot2)
library(lda)
library(stringr)
library(tm)
library(KoNLP)
library(topicmodels)
library(LDAvis)
library(servr)
library(LDAvisData)
library(MASS)
library(wordcloud)
library(wordcloud2)
library(RColorBrewer)
library(treemap)
#Sys.setlocale(category = "LC_CTYPE", locale = "ko_KR.UTF-8")

require(showtext) #R샤이니에서 한글 안깨지게 하는 코드
font_add_google(name='Nanum Gothic', regular.wt=400,bold.wt=700)
showtext_auto()
showtext_opts(dpi=112)

setwd("C:/Users/안윤빈/Desktop/4-1/종설1/gamsung") #불러들일경로 (negative,positive.txt여기에있어야함)
#txt<-readLines("15.txt",warn=FALSE) #감정분석할 텍스트파일 불러오기

path <- file.path("C:/Users/안윤빈/Desktop/4-1/종설1/txt")  # 폴더 경로를 객체로 만든다
kor <- list.files(file.path(path, "애쓰지 않고 편안하게"))  #폴더 경로 중 eng라는 폴더에 있는 파일들 이름을 list-up해서 객체로 만든다.
kor.files <- file.path(path, "애쓰지 않고 편안하게", kor)   #아까 list-up한 파일의 이름들로 폴더의 경로를 다시 객체로 만든다.
content<- readLines(file.path(path,"애쓰지 않고 편안하게/content.txt"))          # content를 따로 저장
#all.files <- c(kor.files, eng.files) 다른 폴더에 있는 파일과 같이 돌릴때 사용
txt <- lapply(kor.files, readLines)    #txt에 이름을 붙여서 새 객체 생성
topic <- setNames(txt, kor.files)      # 텍스트 파일의 목록만큼 데이터를 읽어오고, 그 결과를 list 형태로 저장
topic <- sapply(topic, function(x) paste(x, collapse = " "))  #topic에 있는 리스트들을 공백을 두고 이어붙임
```

- 1~18: 사용할 라이브러리를 불러온다.
- 21~24: 웹 시각화에서 한글 깨짐 방지를 위해 폰트를 불러온다.
- 29~36: 데이터 로드 모듈

```
library(tm) # 텍스트 마이닝 패키지
stop_words <- stopwords("SMART") # 패키지에서 지원하는 불용어 목록
stop_words <- c(stopwords("SMART"), "")
# stopwords("SMART")

# pre-processing:
topic <- gsub("'", "", topic)  # remove apostrophes(')
topic <- gsub("[[:punct:]]", " ", topic)  # replace punctuation(구두점(!@#,.)) with space
topic <- gsub("[[:cntrl:]]", " ", topic)  # replace control characters(제어문자(\n, \t)) with space
topic <- gsub("^[[:space:]]+", "", topic) # remove whitespace at beginning of documents
topic <- gsub("[[:space:]]+$", "", topic) # remove whitespace at end of documents
for(i in (0:100)) {topic <- gsub(i,"",topic)}
for(i in (2000:2020)) {topic <- gsub(i,"",topic)}
topic <- gsub("lineheigh", "", topic)
topic <- gsub("setextparagraphalignlef", "", topic)
topic <- gsub("clas", "", topic)
topic <- gsub("styl", "", topic)
topic <- gsub("setextparagrap", "", topic)
topic <- gsub("있습니", "", topic)
topic <- gsub("malgu", "", topic)
topic <- gsub("gothi", "", topic)
topic <- gsub("합니", "", topic)
topic <- gsub("때문", "", topic)
topic <- gsub("않았", "", topic)
topic <- gsub("있었", "", topic)
topic <- gsub("그럴", "", topic)
topic <- gsub("것같", "", topic)
topic <- gsub("되었", "", topic)
topic <- gsub("없으", "", topic)
topic <- gsub("이럴", "", topic)
topic <- gsub("있도", "", topic)
topic <- gsub("길수", "", topic)
topic <- gsub("하게", "", topic)
topic <- gsub("많았", "", topic)
topic <- gsub("말해", "", topic)
topic <- gsub("[A-Za-z]","",topic)
doc.list <- strsplit(topic, "[[:space:]]+")
```

- 83~119: 데이터 정제 모듈

```r
# 명사 추출
useSejongDic()  # 세종 사전 사용
doc.list <- sapply(doc.list, extractNoun, USE.NAMES=F)
doc.list <- unlist(doc.list)
doc.list <- Filter(function(x){nchar(x)>1}, doc.list)

# compute the table of terms:
# 저장한 용어를 테이블 형식으로 저장
term.table <- table(doc.list)
term.table <- sort(term.table, decreasing = TRUE)

# remove terms that are stop words or occur fewer than 5 times:
# 불용어 또는 5회 미만으로 언급된 단어들을 제거
#del <- names(term.table) %in% stop_words | term.table < 5
del <- term.table < 5
term.table <- term.table[!del]
vocab <- names(term.table)
#write.table(vocab,"아몬드.txt",sep="\t",row.names=FALSE)

# now put the documents into the format required by the lda package:
# 문서를 LDA패키지에 필요한 형식으로 삽입
get.terms <- function(x) {
  index <- match(x, vocab)
  index <- index[!is.na(index)]
  rbind(as.integer(index-1), as.integer(rep(1, length(index))))
}
documents <- lapply(doc.list, get.terms) # 각 문서에 나오는 단어와 빈도수를 리스트 형식으로 저장
#doc.list
#documents
#get.terms()
# Compute some statistics related to the data set:
# 데이터셋과 관련된 일부 통계를 계산
D <- length(documents)  # number of documents (2,000)
W <- length(vocab)  # number of terms in the vocab (14,568)
doc.length <- sapply(documents, function(x) sum(x[2, ]))  # number of tokens(단어) per document [312, 288, 170, 436, 291, ...]
N <- sum(doc.length)  # total number of tokens in the data (546,827)

term.frequency <- as.integer(term.table)  # frequencies of terms in the corpus [8939, 5544, 2411, 2410, 2143, ...]
# MCMC and model tuning parameters:
# MCMC 및 모델 튜닝 매개 변수
K <- 3  # 토픽의 개수 설정
G <- 5000 # 반복 횟수
alpha <- 0.02
eta <- 0.02
```

- 121~137: 자료구조 변환 모듈

- 142~147: 모델 생성 모듈

- 153~164: 통계 계산 모듈

```r
# Fit the model:
# 각 문서에 K개의 토픽을 중 하나를 랜덤하게 할당
# 모든 문서들은 토픽을 갖고, 모든 토픽은 단어 분포를 가지게 됨.
#  각 토픽에 대해, 두 가지를 계산
#     1. 문서 d의 단어들 w 중 토픽 t에 해당하는 단어들의 비율을 계산 (랜덤으로 할당된 토픽 t에 문서들의 어떤 단어들이 가장 많이 언급되었는지)
#     2. 단어 w를 가지고 있는 모든 문서들 중 토픽 t가 할당된 비율 계산 (해당 단어 w가 토픽 t에 적합한지)
#  결과 1*2에 따라 토픽 t를 새로 고른다. (토픽 t에 적합한 단어 w를 선정)
# 이 과정이 충분히 반복되고 나면, 안정적인 상태에 도달
library(lda)
library(topicmodels)
set.seed(357) # sampling
t1 <- Sys.time()
fit <- lda.collapsed.gibbs.sampler(documents = documents, K = K, vocab = vocab,
                                   num.iterations = G, alpha = alpha,
                                   eta = eta, initial = NULL, burnin = 0,
                                   compute.log.likelihood = TRUE)


t2 <- Sys.time()
t2 - t1  # about 24 minutes on laptop

#Terms.Probability<- 10^t(result$theta)

theta <- t(apply(fit$document_sums + alpha, 2, function(x) x/sum(x)))
phi <- t(apply(t(fit$topics) + eta, 2, function(x) x/sum(x)))
```

- 174~182, 206~212: 분석 모듈

```
result <- list(phi = phi,
               theta = theta,
               doc.length = doc.length,
               vocab = vocab,
               term.frequency = term.frequency,encoding='UTF-8')

#x<-data.frame(vocab,term.frequency)
#write.table(x,"LDA_data.txt",sep="\t",row.names=FALSE)
#sorted <- t(apply(phi,1,sort))

library(LDAvis)
options(encoding = 'UTF-8')
# create the JSON object to feed the visualization:
json <- createJSON(phi = result$phi,
                   theta = result$theta,
                   doc.length = result$doc.length,
                   vocab = result$vocab,
                   term.frequency = result$term.frequency,encoding='UTF-8')

serVis(json, out.dir = 'vis', open.browser = TRUE)
#-----------------------------------------------------------------------------------------------------------------------
data(TwentyNewsgroups, package = "LDAvis")

ui <- fluidPage(
  # App title ----
  titlePanel("Review Analysis System"),

  # Sidebar layout with input and output definitions ----
  sidebarLayout(
    # Sidebar panel for inputs ----
    sidebarPanel(
      "summary",textOutput("content"), br()),

    mainPanel(

      # Output: Tabset w/ plot, summary, and table ----
      tabsetPanel(type = "tabs",
                  tabPanel("topic", sliderInput("nTerms", "Number of terms to display", min = 20, max = 40, value = 30), visOutput('myChart')),
                  tabPanel("sentimental analysis",
                           splitLayout(
                             style = "border: 1px solid silver;",
                             cellArgs = list(style = "padding: 4px"),
                             plotOutput(outputId = "sentiment_result"),plotOutput(outputId = "treemap")))
      )

    )))
```

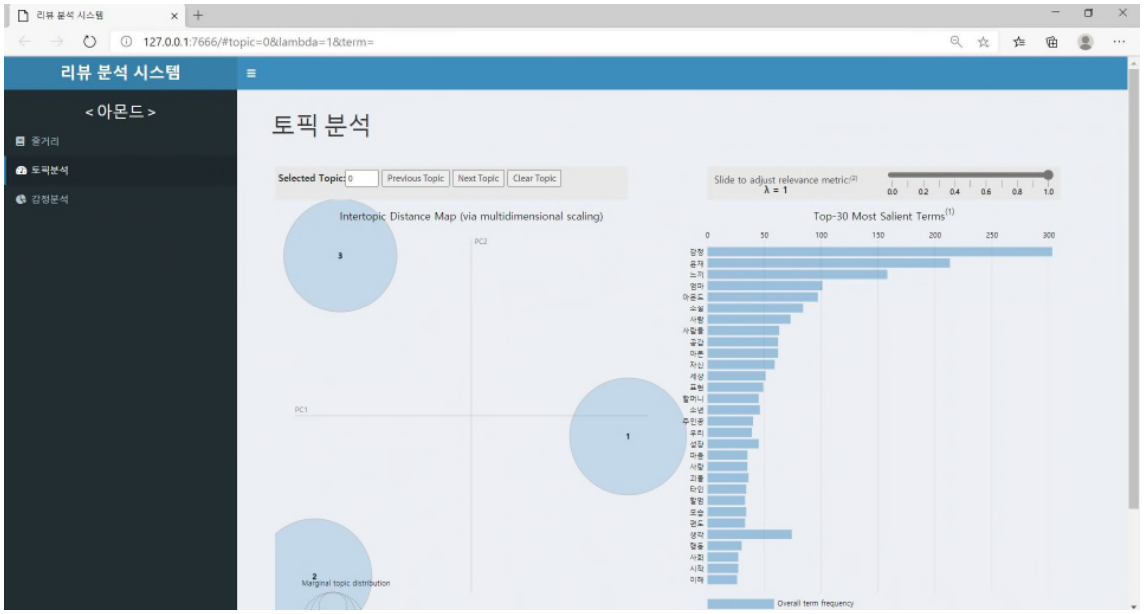
- 214~259: 결과 전송 모듈

```
server <- shinyServer(function(input, output, session) {
  output$myChart <- renderVis({
    if(!is.null(input$nTerms)){
      with(result,
           createJSON(phi = result$phi,
                      theta = result$theta,
                      doc.length = result$doc.length,
                      vocab = result$vocab,
                      term.frequency = result$term.frequency,encoding='UTF-8'))


    }
  })
  output$sentiment_result <- renderPlot({pie(sentiment_result, main="감정분석 결과",col=c("skyblue2","lightcoral","palegreen2"),
                                             label=paste(names(sentiment_percent),'', sentiment_percent,"%"),
                                             border = FALSE, radius=1)})

  output$content<-renderText(content)

  output$treemap <- renderPlot({
    dset<-data.frame(keywords=names(sent.table), sentiment=sent.frequency)#괄호안에 데이터셋 넣으면됨()

    treemap(dset
            ,index=c("keywords")#괄호안에 "키워드" 로 바꾸면됨
            ,vSize=c("sentiment") # 타일의 크기 (언급횟수로 바꾸면 됨)
            ,vColor=c("sentiment") # 타일의 컬러
            ,type="value" # 타일 컬러링 방법
            ,title="Sentimental Topic"
            ,title.legend="Frequency"
            ,fontsize.labels = 9
            ,fontface.labels = c("bold")
            ,fontfamily.labels = "wqy-microhei"
            ,palette = "GnBu" #위에서 받은 팔레트 정보 입력
            ,border.col = "white") # 레이블의 배경색
  })
})
shinyApp(ui = ui, server = server)
```

- 261~297: 시각화 모듈

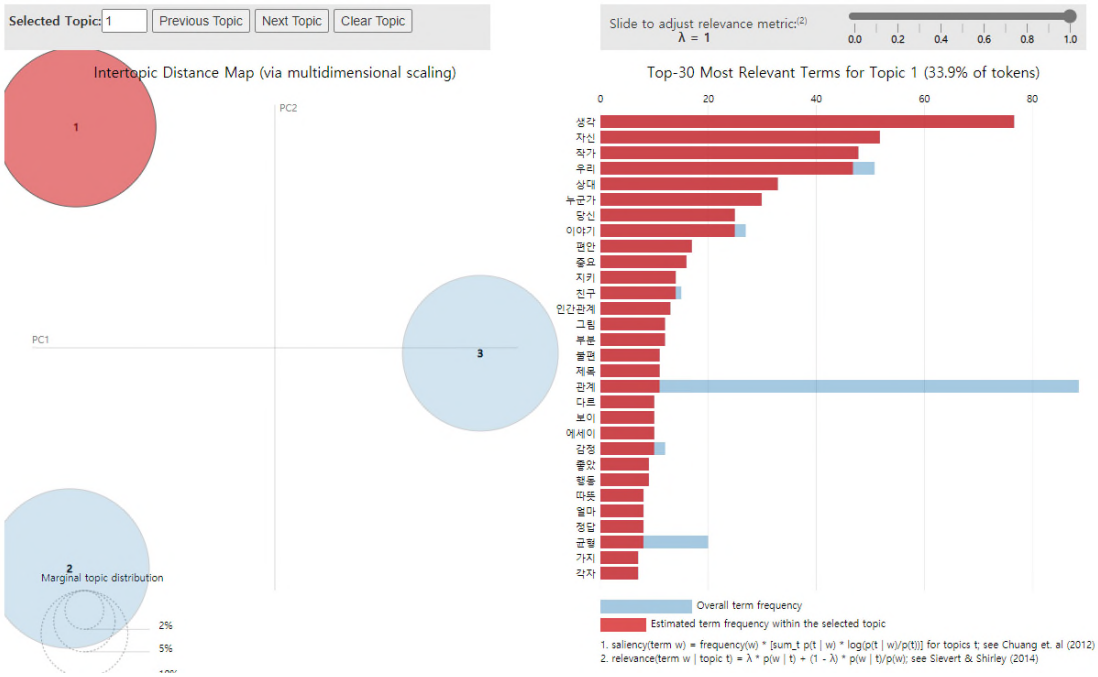
2)-2. 데모 화면

- 오른쪽에는 책의 줄거리, 책 이미지, 데이터 정보가 뜨며 메뉴에서 토픽 추출 결과와 감정 분석 결과를 확인할 수 있다.

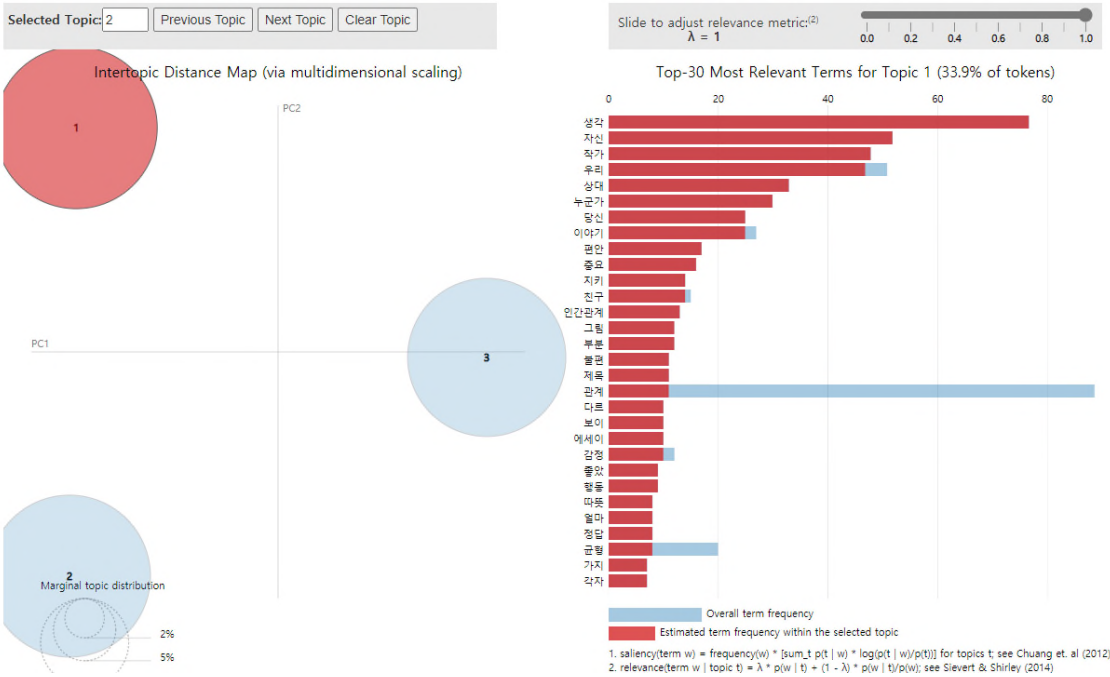


- LDA 알고리즘을 이용하여 책 리뷰에서 토픽을 추출한 결과
- 토픽의 개수(K)를 3으로 설정
- 각 원은 하나의 토픽을 나타낸다.
- 토픽은 코퍼스(단어들의 집합)이며, LDA 알고리즘에 의해 각 토픽에 단어가 할당된 결과이다.

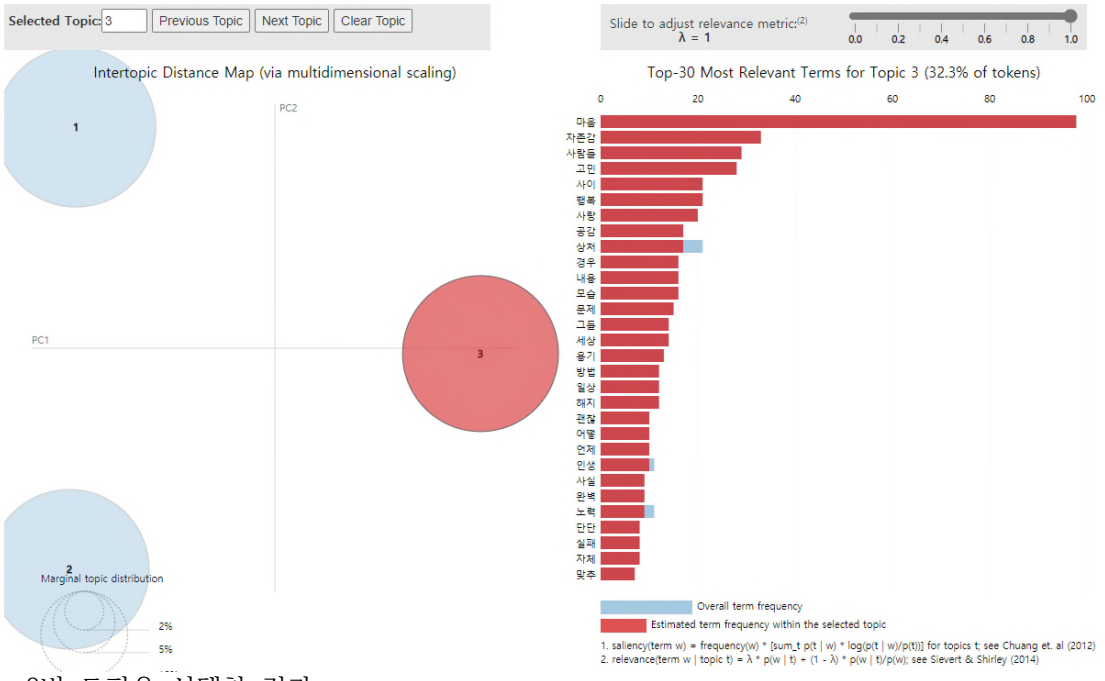
- 각 원의 넓이는 코퍼스 내에서 N개의 전체 토큰들에 대한 비율
- 토픽에서 단어가 발생할 확률 벡터인 P(w|t)에 Principal Component Analysis(PCA)를 적용하여 2 차원의 벡터로 압축
- PCA는 데이터의 분산(variance)을 최대한 보존하면서 서로 직교하는 새 기저(축)를 찾아, 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간으로 변환하는 기법
- 토픽을 선택하지 않았을 때는 가장 중요한 단어 30개(가장 출현 빈도수가 높은 단어)를 보여줌
- ë는 토픽 k에서 용어 w의 확률에 주어진 가중치를 결정
- Previous Topic을 누르면 이전 토픽, Next Topic을 누르면 다음 토픽으로 넘어가며 Clear Topic을 누르면 토픽이 선택되지 않은 초기의 상태로 돌아간다.



- 1번 토픽을 선택한 결과
- 파란색 막대는 코퍼스 내에서 각 용어의 전체적인 빈도수이며, 빨간색 막대는 주어진 토픽에 의해 생성된 용어가 출현한 횟수이다.



- 2번 토픽을 선택한 결과



- 3번 토픽을 선택한 결과

3) 감정분석

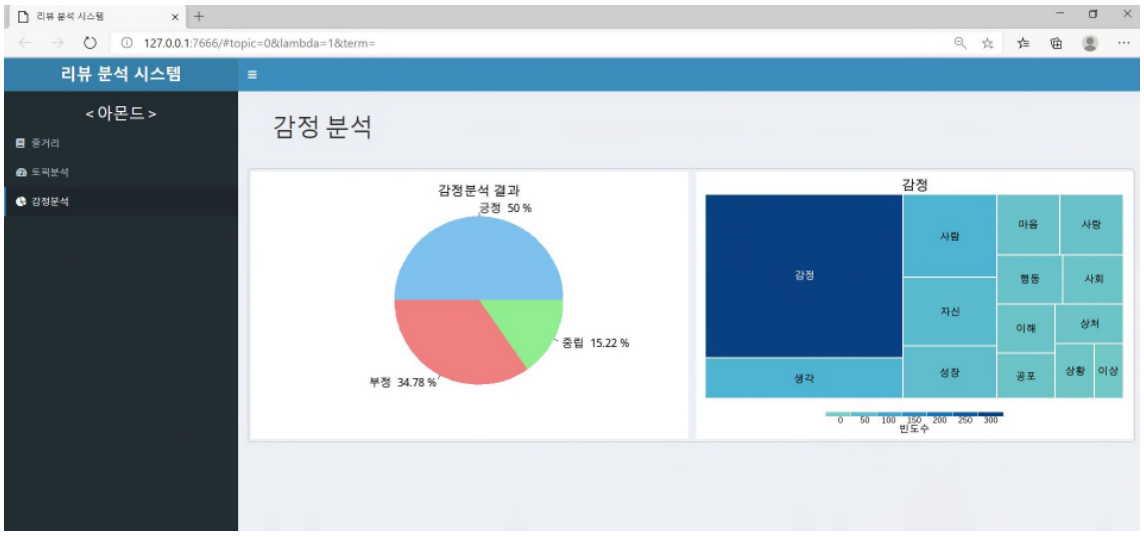
```
positive <- readLines("positive.txt")
positive=positive[-1]
negative <- readLines("negative.txt")
negative=negative[-1]
```

```r
sentimental = function(sentences, positive, negative){

  scores = laply(sentences, function(sentence, positive, negative) {

    sentence = gsub('[[:punct:]]', '', sentence) # 문장부호 제거
    sentence = gsub('[[:cntrl:]]', '', sentence) # 특수문자 제거
    sentence = gsub('\\d+', '', sentence)        # 숫자 제거

    word.list = str_split(sentence, '\\s+')      # 공백 기준으로 단어 생성 -> \\s+ : 공백 정규식, +(1개 이상)
    words = unlist(word.list)                    # unlist() : list를 vector 객체로 구조변경

    pos.matches = match(words, positive)           # words의 단어를 positive에서 matching
    neg.matches = match(words, negative)

    pos.matches = !is.na(pos.matches)            # NA 제거, 위치(숫자)만 추출
    neg.matches = !is.na(neg.matches)

    score = sum(pos.matches) - sum(neg.matches)  # 긍정 - 부정
    return(score)
  }, positive, negative)

  scores.df = data.frame(score=scores, text=sentences)
  return(scores.df)
}

result=sentimental(topic, positive, negative)

result$color[result$score >=1] = "Olive Drab 2"
result$color[result$score ==0] = "Gray 60"
result$color[result$score < 0] = "Orange Red 2"

result$remark[result$score >=1] = "긍정"
result$remark[result$score ==0] = "중립"
result$remark[result$score < 0] = "부정"

sentiment_result= table(result$remark)
sentiment_percent= round(sentiment_result/sum(sentiment_result)*100, 2) # 백분율로 환산
# ------------------------------------------------------------------------------여기까지 감정분석
```

- 38~80 감정분석 모듈

```r
#-------------------------------------------------------------------------------------------------------------
# 토픽 단어들을 단어 감성사전과 매칭
topicwords <- top.topic.words(fit$topics, 20, by.score = TRUE)  # 토픽별로 상위 20개 단어뽑기
topicwords<-c(topicwords[,1],topicwords[,3],topicwords[,3])     # 단어를 리스트로 합치기

search<- names(term.table) %in% topicwords                      # 빈도수 테이블에서 단어 검사
newterm.table <- term.table[search]                             # 단어와 빈도수로 새로운 테이블 생성

new_sentiment <- readLines("단어 감성사전.txt")
new_sentiment=new_sentiment[-1]

newterm.data<-data.frame(newterm.table)
newterm.vec<-rep(newterm.data$doc.list)

sent.matches = match(newterm.vec, new_sentiment)          # 단어를 matching
sent.matches = !is.na(sent.matches)            # NA 제거, 위치(숫자)만 추출

sent.table<-newterm.table[sent.matches]               # 단어와 빈도수로 새로운 테이블 생성
sent.frequency <- as.integer(sent.table)
#length(sent.table)
```

- 184~203: 정제된 감정사전과 LDA 결과를 이용한 Treemap 구현을 위한 매칭 모듈



- rShiny로 구현한 감정분석 관련 부분
- 가장 좌측에는 분석한 책의 간략한 줄거리가 표시된다
- 중앙의 감정분석 탭을 클릭하면 감정분석 결과가 나타난다
- 1권에 책에 대한 15개의 크롤링된 리뷰를 감정분석한 결과 (푸른색= 긍정적이라고 판단된 리뷰, 녹색= 중립적이라고 판단된 리뷰, 붉은색= 부정적이라고 판단된 리뷰)
- 파이차트를 통하여 크롤링한 문서의 리뷰에 대한 평가를 알 수 있다.
- 가장 우측의 Treemap에는 LDA 분석된 토픽중 감정사전에 실린 감정어들을 나타낸다. Treemap에서 차지하는 부분이 클수록 리뷰들 속 많이 언급된 감정어이다.
- Treemap 속 리뷰들의 감정어를 통해 리뷰들의 대략적인 감정이나 내용을 유추해 볼 수 있다.

Ⅲ. 결론

1.연구 결과

본 연구에서는 LDA 알고리즘과 감정분석을 이용하여 교보문고에 있는 책 리뷰의 키워드들을 분석해보았다. 분석한 토픽과 감정단어를 바탕으로 책의 내용 유추하여 책을 읽으려는 독자들에게 도움이 될 수 있을 것이라 예상된다. 또한, 토픽 추출 시스템을 이용하여 리뷰 데이터 수집에 대한 편리성이 증가할 것이며, 기업은 책 리뷰 분석을 통해 책에 대한 독자들의 반응을 검토 가능하여 책의 전망을 파악할 수 있을 것이라 예상된다.

2. 작품제작 소요재료 목록

순번	소요재료
1	노트북

참고자료

순번	참고자료
1	• **동적토픽모델 기법을 적용한 특정 뉴스 토픽의 변화과정 분석** (2019.02, 동국대학교, 배전희)
2	• **일기자료 연구에서 토픽모델링 기법의 활용가능성 검토** (2016.01, 비교문화연구, 22(1), 89~135, 남춘호)
3	• **단어 유사도를 이용한 뉴스 토픽 추출** (2017.11, 정보과학회논문지, 44(11), 1138~1148, 김동욱, 이수원)
4.	• Study on the Topic Mining and Dynamic Visualization in View of LDA Model (2018.12, Modern Applied Science, 13(1), Ting Xie, Ping Qin, Libo Zhu)
5	• Latent Dirichlet Allocation (2003, Journal of Machine Learning Research, Vol. 3, pp. 993−102, Blei, D., A. Ng, and M. Jordan)