데이터 예측 모델과 기계 학습의 응용

4조 여민주, 이호정, 이희주, 최인영, 한지윤

리더스 신규 가입자를 위한 맞춤형 도서 추천 서비스

한양대학교 ERICA
Education Research Industry Cluster @ Ansan

리더스가 추천하는 OO 님이 관심 보일만한 책

1. 리더스란?

리더스란 읽은 책을 캡쳐하거나 마음에 드는 문구에 밑줄을 긋고 감상평을 남길 수 있는 플랫폼이다. 기록을 통해 지금까지 읽은 책을 관리할 수 있으며 독서습관을 들일 수 있게 도움을 준다. 카메라 기능을 이용해 책을 캡쳐하고 기록할 수 있으며 SNS 기능으로 이용자 간 독서기록을 공유함으로써 더 많은 책에 접근할 수 있다. 또한 '북클럽'이라는 독서모임에서 리더스가 추천하는 다양한 주제의 책을 함께 읽을 수도 있다.

2. 리더스의 책 추천 방법

현재 리더스에서 책을 추천받는 방법에는 세 가지가 있다.

- 첫째, <다른 사람들 책장에 많이 들어있는 책>을 확인한다.
- 둘째, <분야별 베스트 도서>를 확인한다.
- 셋째, '팔로워'를 통해 친구의 책 기록을 확인할 수 있다.

이 세가지의 공통점은 모두 내가 아닌 '남에 의한 추천'이라는 것이다. 나의 독서 취향이나 기준이 아닌 '다른 사람', '분야별', '팔로워'에 의한 추천은 나에게 꼭 맞지 않을 수 있다. 그렇기에 가입자의 독서 취향 기준을 정할 필요가 있다.

3. 새로운 추천 방법에 대한 필요성

어플의 기조는 "읽은 책을 관리하고 새로운 책을 발견하세요" 이지만, 가입자에게 남이 읽은 수많은 책 중에서 읽고 싶은 책을 발견해야 하는 수고로움을 덜어주고 싶었다.

데이터 분석을 통해 신규 가입자의 독서 취향을 파악하여 "리더스가 추천하는 oo 님이 관심 보일만한 책" 을 제공하는 것이다. 어플을 사용하기 전 일련의 과정을 거쳐 본인만의 독서 취향 기준을 만든다. 책과 관련한 간단한 정보를 보고 본인에게 맞을 것 같은 책을 선택함으로써 가입자의 취향기준 데이터를 쌓는다. 리더스는 데이터 분석을 통해 가입자의 기준에 따라 관심을 보일만한 책을 추천한다.

4. '신규 가입자 맞춤형 도서 추천 서비스'

신규 가입자에게 랜덤으로 추출한 1000개의 도서 중 45개의 보기를 준 뒤 본인의 취향에 맞는 책을 선택하도록 한다. 이때 책의 간단한 설명이나 작가, 페이지 수 등의 정보를 제공하는데 이는 신규 가입자에게 책에 대한 이해도를 높이는 동시에 데이터 분석의 기본 자료로 사용된다. 즉 신규 가입자가 선택한 책의 정보와 비슷한 맞춤형 도서를 추천하는 것이다.

5. 서비스의 구체적 절차 설명

1) 리더스 앱에서 신규 가입자들에게 랜덤으로 선택된 45권의 책을 한 화면에 9권씩, 총 5번의 선택지를 제공해준다. 신규 가입자들은 책을 선택하면 보이는 간단한 설명을 읽고 개수 제한 없이 관심이 가는 도서를 선택한다.

2) 1)에서 고른 도서의 페이지수, 카테고리를 고려하여 선호할 만한 도서들을 추천하고, 책의 완독률로 순위를 매겨 가입자가 선호할만한 책들을 1순위 ~ 9순위까지 추천해준다.

3) 자신의 관심도에 맞는 책을 추천받은 신규 가입자들은 리더스 앱에 등록된 수천 권의 책들을 일일이 분석하지 않고도 추천된 책들을 통해 그들이 평소 선호하는 도서의 카테고리, 페이지 수 등에 대한 평균적인 결과를 유추해 리더스 앱을 효과적으로 이용할 수 있다.

6. 서비스 도입을 위한 코딩 설명

#####

Import readers DB

새로운 서비스 모델을 구현하기 위해 Readers 의 기존 데이터를 가져온다.

#####

Sys.setlocale("LC_ALL", locale="Korean")

"Korean"으로 로케일을 변경한다.

list.files("2022_05_data")

list.files() 함수를 사용하여 R 에서 원하는 폴더의 파일과 폴더 이름을 가져온다.

```
library(readxl)
library(tidyverse)
```

'readxl'과 'tidyverse' 패키지를 로드한다.

```
# import
user <- read_excel("2022_05_data/01_user.xlsx")
user_cat <- read_excel("2022_05_data/02_user_cat.xlsx")
follow <- read_excel("2022_05_data/03_follow.xlsx")
user_book <- read_excel("2022_05_data/04_user_book.xlsx")
book <- read_excel("2022_05_data/05_book.xlsx")
book_cat <- read_excel("2022_05_data/06_book_cat.xlsx")
cat <- read_excel("2022_05_data/07_cat.xlsx")
scrap <- read_excel("2022_05_data/08_scrap.xlsx")
```

서비스 구현을 위해 필요한 readxl 패키지('user', 'user_cat', 'follow', 'user_book', 'book', 'book_cat', 'cat', 'scrap') 엑셀 파일을 가져온다.

```
library(dplyr)
```

'dplyr' 패키지를 이용하여 데이터 전처리를 위한 작업을 시행한다.

```
# book_id 별로 중복되는 행을 정리
X<-book_cat %>% group_by(book_id) %>%
summarise(new_cat=paste(book_category_id, collapse = ","))
X
```

group_by 와 sumarise 함수를 사용하여 X 에 book_id 변수를 기준으로 그룹화시키고 중복되는 행을 정리한다.

```
# 필요한 데이터 조인
```

```
A <- inner_join(book_cat,cat,by='book_category_id')
A
```

'dplyr' 패키지에서 inner_join 함수를 사용하여 'book_category_id'를 기준으로 'book_cat'과 'cat' 데이터를 병합한다.

```
B<-A%>% group_by(book_id) %>% filter (! duplicated(book_id))
B<-B[,c(1:3)]
```

book_id 열 변수로 데이터프레임을 그룹화한 다음 'filter'와 'duplicated' 함수를 사용하여 중복 행을 제거하기 위해 요소를 필터링해준다.

```
# book 데이터의 첫번째 행 이름을 "book_id"로 변경
names(book)[1]=("book_id")
```

'names' 함수로 book 데이터의 첫 번째 행 변수명을 'book_id'로 변경한다.

```
# book 데이터와 B 데이터를 'book_id'로 이너조인
DATA <- inner_join(book,B,by='book_id')
```

inner_join 함수를 사용하여 'book_id'를 기준으로 'book'과 'B' 데이터를 join 한다.

```
# 1000개의 랜덤데이터 추출
RD<-DATA[sample(nrow(DATA),1000),]
```

리더스의 방대한 데이터를 추려 간단하게 코딩 과정을 보이기 위해 1,000권의 책을 랜덤으로 선별한다. 이를 위해 sample 함수를 사용하여 DATA 에서 1,000개의 랜덤 데이터를 추출한다.

```
# 카테고리의 수가 너무 많아 정리하고자 했으나 어려움이 있어
# 일단, 1000개의 책에 대한 카테고리를 1~5를 랜덤으로 부여함
RD$book_category_id <- sample(c(1:5), 1000, replace=T)
RD
```

카테고리를 분류하기 위해 1,000개의 랜덤 데이터에 1~5를 랜덤으로 부여하고, ‘replace=T’를 사용해 교체가 두 번 발생하지 않도록 한다.

```
# 신규가입자에게 제공할 45개의 랜덤도서 추출

train<-RD[sample(nrow(RD),45),]

train
```

랜덤으로 추출된 1,000권의 책 중 다시 랜덤으로 45권을 선별하여 신규 가입자들에게 제공하기 위해 sample 함수를 사용하여 RD 에서 45개의 랜덤 데이터를 추출하고, 랜덤(train) 데이터에 집어넣는다.

```
# 신규 가입자가 선택하는 상황

# 새로운 변수 (소비자 선택 Y)
```

‘소비자 선택’이라는 새로운 변수를 생성한다.

```
# Y=1 선택 Y=0 선택 안함
```

‘Y=1’은 소비자가 선택한 책, ‘Y=0’은 소비자가 선택하지 않은 책으로 구분한다.

```
# 일단 Y = 0으로 해서 시작

train %>% mutate(Y=0) -> train
```

파생 변수를 추가하기 위해 mutate 함수를 사용한다.

```
# 9개 세트씩 5번 선택지를 준다고 가정
```

리더스 앱 신규 가입자들에게 랜덤으로 선택된 45권의 책을 한 화면에 9권씩, 총 5번의 선택지를 제공한다고 가정한다.

```
# 첫 번째 9개 세트(1:9) -> 2,3,7 선택 했다고 가정

train[c(2,3,7),"Y"]<-1
```

```
# 두 번째 9개 세트(10:18) -> 3,7,9 선택 했다고 가정
train[c(12,16,18),"Y"]<-1
# 세 번째 9개 세트(19:27) -> 1,4,8번 선택 했다고 가정
train[c(19,22,26),"Y"]<-1
# 네 번째 9개 세트(28:36) -> 3,8,9번 선택 했다고 가정
train[c(30,35,36),"Y"]<-1
# 다섯 번째 9개 세트(37:45) -> 4,5,6번 선택 했다고 가정
train[c(40,41,42),"Y"]<-1

# 신규가입자가 45권 중 15권을 선택했다고 가정
table(train$Y)
```

table 함수를 사용하여 랜덤(train) 데이터의 Y 열에 대한 응답 카운트를 해준다.

```
# book_category_id 를 factor 형 변수로 변경
train %>% mutate(Y=as.factor(Y),
book_cat=as.factor(book_category_id)) -> train
glimpse(train)
```

book_category_id 를 범주형 데이터로 표현하기 위해 factor 형 변수로 변경한다.

'train' 데이터프레임의 구조를 확인하기 위해 glimple 함수를 사용하여 가독성을 높인다.

```
# 결정트리모델 시각화
library(rpart)
library(rpart.plot)
```

Decision Tree 분석 및 시각화를 위해 'rpart'와 'rpart.plot' 패키지를 사용한다.

```
tree<-rpart(Y~book_cat+page,
data=train,
method='class')
```

모델 학습에 필요한 train 데이터를 사용하여 모델을 생성한다.

```
rpart.plot(tree)
```

'rpart.plot' 패키지를 사용하여 결정 트리 모델을 시각화한다.

```
# 학습된 데이터를 토대로 책별 신규가입자가 읽을 확률을 예측함
RD %>% select(book_id, book_category_id, page) -> RD2
RD2
```

RD 데이터프레임에서 'book_id', 'book_category_id, page'에 대한 변수를 추출하여 'RD2'에 집어넣는다.

```
RD2$book_cat<-as.factor(RD2$book_category_id)
```

RD2 데이터프레임의 'book_category_id' 변수를 범주형 데이터로 표현하기 위해 fartor 형 변수로 변경한다.

```
glimpse(RD2)
```

'RD2' 데이터프레임의 구조를 확인하기 위해 glimple 함수를 사용하여 가독성을 높인다.

```
# 예측을 활용하여 신규가입자를 위한 책 추천 시스템의 기초가 될 데이터프레임을 형성
predict(tree, newdata=RD2)
```

predict 함수를 사용하여 예측치와 실측치를 분석한다.

```
PR<-predict(tree, newdata=RD2)
```

예측을 수행할 새로운 데이터를 생성해 PR 에 집어넣는다.

```
PR<-data.frame(PR)
PR[,'book_id']<-RD$book_id
PR[,'title']<-RD$title
```

0.51이상의 확률을 읽을 것이라고 가정

READ<-subset(PR,X1>=0.51)

READ

PR 데이터프레임에서 '0.51 이상의 확률'에 대한 부분 집합을 추출하기 위해 subset 함수를 사용한다.

완독비율 구하기 (book_id 별로 '읽음' 상태가 되어 있는 책의 비율)

Book_ratio<-user_book %>%

select(book_id, read_status) %>%

user_book 데이터프레임에서 'book_id', read_status'에 대한 변수를 추출하여 'Book_ratio'에 집어넣는다.

mutate(book_read_done=ifelse(read_status %in% "READ_STATUS_DONE", 1, 0)) %>%

group_by(book_id) %>%

summarise(ratio=sum(book_read_done)/n())

group_by 와 sumarise 함수를 사용하여 book_id 변수를 그룹화시키고 '읽음' 상태가 되어 있는 책의 비율을 정리한다.

book_id 를 기준으로 READ 데이터에 완독비율 데이터 합치기

BEST<- left_join(READ,Book_ratio,by='book_id')

left join 함수를 사용하여 왼쪽에 있는 'READ'를 기준으로 'book_id'를 병합한다.

완독률을 내림차순으로 정렬

BEST$ratio<-sort(BEST$ratio, decreasing = T)

sort 함수를 사용해 'BEST$ratio'를 내림차순으로 정렬한다.

완독률이 1인 책들만 Best Of Best 로 선정

BOB <-BEST %>% filter (ratio==1)

filter 함수를 사용하여 ratio 가 1인 책들을 필터링하고, ‘BOB’에 집어넣는다.

Best Of Best 책들 중 9권만 랜덤으로 선정

BOB<-BOB[sample(nrow(BOB),9),]

BOB

필터링 된 BOB 의 데이터에서 9권의 책을 랜덤으로 선정한다.

#####

5. 결론 및 기대효과

신규가입자에게 랜덤으로 추출한 선택지를 고르게 하고 선택지의 특성과 정보를 반영해 신규가입자가 관심 가질 확률이 높은 책을 추천하고자 했다. 남이 좋아하는 것을 따라하는 것이 아닌 나의 취향에 맞고 좋아할만한 책을 추천받을 수 있다면 리더스라는 앱을 사용함에 있어 좀 더 맞춤화됐다는 느낌을 받을 수 있을 것이다. 더 나아가 추천받은 책이 마음에 든다면 리더스라는 앱의 서비스 만족도가 높을 것이며 앱 사용의 충성심도 강해질 것이다.

“리더스가 추천하는 oo 님이 관심 보일만한 책” 기능은 어플에서 (추천도서 > 다른 사람들 책장에 많이 들어 있는 책)과 함께 (추천도서 > 내 책장에 들어갈 만한 책)이라는 이름으로 추가될 수 있다. 다른 사람들의 독서 취향과 비교함으로써 요즘 유행하는 책이나 자신의 독서습관을 확인할 수 있다. 또한 신규 가입 시 이뤄지는 일회성 추천이 아닌 지속적으로 취향의 범위를 넓혀가며 독서의 다양성을 추구하도록 서비스가 이뤄질 수 있다.

6. 한계점과 보완점

리더스가 제공한 45개의 랜덤한 책이 가입자 마음에 들지 않을 수 있고 아무리 책 설명을 제공했더라도 간단한 정보만으로 책을 제대로 파악하지 못할 수 있다. 따라서 가입자의 취향을 제대로 반영하지 못할 수 있다. 또한 코딩에서 카테고리를 분류할 때 명확한 기준이 아닌 임의로 나누었다는 한계를 지닌다.

이러한 한계점을 보완하기 위해서 먼저 가입자의 취향을 정확히 분석하기 위해 최대한 다양한 취향 선택권을 제시하고 그 중에서도 적어도 1개쯤 취향이 담긴 선택지가 있어야한다. 또한, 한두 달 간격으로 추천 서비스를 제공하고 이전 서비스에 대한 평가나 설문조사를

실시함으로써 개선점을 찾아나간다. 코딩 과정에서 카테고리를 나눌 명확한 기준을 세운다면 좀 더 정확하게 가입자의 취향을 파악할 수 있을 것이다.

서비스 시행에 있어 가장 중요한 것은 아무리 좋은 서비스가 있더라도 지속적으로 소비자의 니즈를 충족시키지 못한다면 소용없다. 신규 가입자의 취향에 맞는 모든 도서를 추천했다 하더라도 꾸준히 앱을 사용할 수 있도록 '지속적 추천 서비스'도 이뤄져야 한다. 즉, 자신의 취향 도서를 다 섭렵했더라도 자연스럽게 더 넓은 범위의 취향을 찾을 수 있도록 유도하는 알고리즘이 있으면 좋을 것 같다.