졸업프로젝트 보고서

원클릭: 컨텐츠 기반 필터링을 이용한 음식 추천프로그램

2013171046 심재훈

김동승 교수님 지도

고려대학교 공과대학 전기전자전파 공학부

1. Introduction

❑ Problem Statement

❍ 우리는 점심시간이나 저녁시간 때 무엇을 먹으러 갈지 고민을 많이 한다. 심지어 혼자 밥을 먹을 때도 무엇을 먹을지 몰라 한참을 생각한다. 가끔은 IT의 힘을 빌려 인터넷 검색을 해보지만 수 많은 맛집 사이에서 우리는 결국 선택을 해야만 한다. 여러 가지 선택지 사이에서 이것 저것 고려해보고 고르는 과정은 에너지를 많이 소비한다. 반면에 제시된 선택지에 Yes or No 와 같은 두 개중 하나를 선택하는 것은 비교적 수월하다. 이와 같은 음식 메뉴 하나를 추천해주는 프로그램을 만들고 싶다.

❑ Pains and Needs

❍ 질 좋은 추천 시스템의 바탕은 많은 량의 데이터이다. 하지만 데이터에 의존하지 않고도 추천할 수 있는 방식이 있는데 이는 content-based Filtering(CBF) 이다. 하지만 CBF를 위해서는 content 즉 음식에 관한 정보를 잘 알고 있어야 한다. 해당 음식이 매운지, 짠지, 중식인지 한식인지와 같은 속성을 잘 알고 있어야 좋은 추천을 할 수 있다.

❑ Importance

❍ 추천 시스템은 여러 산업에서 응용이 가능하다. 웹서핑, 온라인 쇼핑, 서점 등 이미 많은 분야에서 사용되고 있지만 성능을 올릴 틈은 아직 많다. 이번에는 널리 이용되는 추천 시스템을 음식이라는 분야에 응용하려고 한다.

2. Related Work (Project)

- **풀고자 하는 연구 내용(Problem)을 다른 논문/프로젝트들의 저자들은 어떻게 풀고 있는가? 해당 문제를 풀기 위해서 기존에는 어떤 접근법(approach)들이 존재했는가? 그것들의 장단점은 무엇인가?**

  추천 알고리즘은 여러 산업에서 쓰인다. 대표적으로 아마존과 같은 인터넷 쇼핑몰에서 상품을 추천할 때 쓰인다. 아마존 같은 경우는 수 많은 고객데이터를 분석해 이를 바탕으로 신규 유저나 기존 유저에게 상품을 추천해준다. 이는 기존에 쌓아온 데이터를 활용한 것이다. 이 방식을 이용하면 캐시된 데이터가 거의 없거나 아예 없는 신규 유저에게도 적은 데이터만으로 양질의 추천을 해줄 수 있다. 반면에 아마존과 같은 빅데이터 분석이 없어도 캐시된 데이터가 풍부한 기존 유저에게는 그 데이터를 이용해 좋은 추천을 해 줄 수 있다. 예를 들면 음악 추천 프로그램 같은 경우다. 음악 같은 경우 1~2시간 동안 20곡 이상을 들을 수 있는데 여기에 등록된 장르, 아티스트와 같은 분류를 이용하면 금방 맞춤형 추천을 해줄 수 있다. 전자의 같은 경우 Collaborative Filtering 방식으로 사용자간 유사도 Matrix를 이용해 많은 량의 데이터가 필요하다. 후자의 경우는 Content-based Filtering으로 Content(음악) 자체를 직접 분석해 이 분석 정보 바탕으로 추천을 해준다. 이외에도 관계간 숨은 속성을 찾아내 이를 활용해 추천하는 방식이 있다. 이는 요즘 트렌드인 Neural Network를 이용한다. 이 방식 또한 수 많은 종류와 많은 량의 데이터가 바탕이 된다.(3번에서 자세히 설명)

3. Solution Approach (Main Idea) / 성능 평가를 위한 실험 계획

- **아이디어란 주어진 문제를 효과적인 체계적으로 풀어나가기 위한 일련의 처리과정을 기술하는 것 입니다. What is your main idea to the problem?**

  - 1). Content-Based Filtering(CBF)

    컨텐츠 기반 필터링 알고리즘은 추천 컨텐츠의 속성을 바탕으로 사용자에게 컨텐츠를 추천해준다. 음식의 경우, 음식의 카테고리 그리고 맛이 그 속성이다. 카테고리는 크게 6가지로 각각 고기, 한, 중, 일식, 분식, 기타류로 나누었다. 맛의 경우에는 맵다, 면 종류이다, 밥 종류이다, 달다, 느끼하다 와 같은 것으로 총 12가지로 나누었다. 음식마다 한 가지 카테고리에 속하고 맛의 경우에는 5개까지 속할 수 있다. CBF 특성상 적은 량의 데이터로도 충분한 성능이 나오기 때문에 협업 필터링 알고리즘과 같은 알고리즘보다는 이 방식을 선택했다.

  - 2). Evaluation

    위와 같은 방식으로 추천을 하게 될 경우 음식의 취향이 성별, 나이, 사는 곳, 시간대와 같은 특징에 따라 크게 달라질 수 있다는 점을 반영 못하게 된다. 그래서 알고리즘 개발 시, 이 변수들을 통제하고 맞춤형으로 개발할 필요가 있다. 20대 남성을 12명을 대상으로 30가지 음식의 선호도를 Yes or No선택하게 하였다. 이 데이터를 train, test set으로 나누어 CBF에 학습시키고 이를 test set으로 검증한다. 단순 임의 추출했을 때와 대비해 어느 정도 성능인지 확인해본다. 성능이 좋아질 때까지 계속 반복적으로 이를 시행하고 알고리즘을 수정한다.

원버튼 음식 추천 프로그램(이하 원버튼)의 핵심은 단순함에 있다. 이 프로그램의 사용자는 Yes or No 중 하나의 선택만 하면 된다. 그 외에 복잡한 알고리즘은 프로그램 안에 숨겨져 있다. 일상생활의 고민을 줄여주기 위한 프로그램이므로 이 단순함이 사용자들에게 매력적일 것이라고 본다.

프로그램은 다음과 같이 구현할 생각이다.

1. 사용자는 프로그램을 실행 후 중앙의 버튼을 눌러 음식을 추천 받는다.

1.*음식의 추천 알고리즘 추후 설명

2. 음식을 추천 받으면 사용자는 Yes or No 를 누를 수 있다.

2* Yes를 누르게 되면 사용자는 기존 알고 있던 음식점을 가거나 지도API를 이용해 검색된 음식점을 갈 수 있다. 여기서 광고를 통해 해당 음식점을 나열해주는 식으로 수익창출을 기대해 볼 수 있다.

2** No를 누르게 되면 해당 음식을 제외한 다른 음식을 추천해준다.

3. Yes를 누르기 전까지 반복한다.

음식의 추천 알고리즘은 다음과 같이 구현할 생각이다.

처음으로 사용자가 프로그램을 이용하면 목록에 있는 음식을 임의 추출해서 보여준다.

사용자가 Yes or No 선택을 하면 그 데이터(*)를 로그에 저장한다.

* 로그에 저장할 데이터는

1. 사용자 정보(나이, 성별, ID)

2. 음식(해당 음식, 음식의 카테고리)

3. Yes or No(1 or 0)

4. 시간

5. 기후(날씨, 온도)

추후 사용자가 프로그램을 이용 시 로그에 있는 데이터를 바탕으로 음식을 가중치(*) 더해서 추출해 보여준다.

* 가중치를 정하는 방법

가중치를 정하는 방법은 알고리즘의 핵심이 될 것이다. 일단 이를 정하는 한가지 방법으로는 기존 목록의 가중치를 전부 100으로 설정해놓고, Yes가 나온 음식의 가중치는 +30, No가 나온 음식의 가중치는 -10을 하는 것이다. 음식이 A,B,C,D 4개가 있다고 하면 총 가중치의 합은 400이다. 여기에 A음식의 경우 Yes, C와 D음식의 경우 No 선택지를 골랐다고 하자. 그러면 A,B,C,D 각각의 가중치는 130,100,90,90이다. 다음에 사용자가 음식을 추천 받는 경우 A를 추천 받을 확률은 130/410 = 31.7퍼센트, C를 추천 받을 확률은 22퍼센트가 된다.

사용자가 프로그램을 이용하는 순간 우리는 사용자가 yes 라고 답할 법한 음식을 추천해줘야 한다. 이를 위해서는 여러가지 머신러닝(통계적) 기법들을 이용할 수 있는데 우리는 Yes or No와 같은 이항결과를 모델링 해야 하므로 로지스틱 회귀가 적절한 방법일 것 같다(1). 어떤 머신러닝 기법을 이용할 것인지는 추후 더 연구해볼 사항이다.

추천 알고리즘 종류 및 적용 가능성

추천 알고리즘은 크게 세가지로 나눌 수 있다. Collaborative filtering(CF), Content-based filtering(CBF), Model-based Collaborative Filtering(MCF) 이 세 가지이다. [1]마지막 MCF가 제일 최근에 나온 알고리즘으로 일반적인 상황에서 성능이 제일 좋다. 일단 이와 같은 알고리즘을 적용하기 위해서는 데이터가 많이 축적되어 있어야 한다. 현재 진행하고자 하는 프로젝트는 기반 데이터가 없으므로 처음부터 이 세가지 알고리즘을 적용하지는 못한다. 다만 추후 알고리즘을 적용한다는 목표를 가지고 이 프로젝트를 진행하면 좋을 것이다. 이를 위해서 추천 알고리즘에는 어떤 데이터가 이용되는지, 어떤 식으로 데이터를 쌓아가면 좋을지 알아볼 것이다. CF는 A아이템을 구매한 사람이 B아이템도 높은 확률로 구매한다면 A와 B를 묶어 추후 A를 구매하는 사람에게 B를 추천하는 방식이다. 이와 같은 연관성을 바탕으로 Matrix를 만들거나 k-Nearest-Neighbor 알고리즘을 이용해 비슷한 성질의 집단을 찾을 수 있다. 가령 A를 산 사람이 B를 사는 경향이 있고, B를 산 사람은 C를 사는 경향이 있으면 A,B,C 모두를 한 집단으로 묶을 수 있다. CF는 세가지 단점이 있다. 첫 번째로 행렬 계산 같은 경우 연산 량이 많아 처리 속도가 느리다. 추천 프로그램 같은 경우 빠르게 결과를 보여주어야 하므로 큰 단점이 될 수 있다. kNN의 경우 O(kdn) 와 O(nd+kn) 중 빠른 것이 시간 복잡도이다[2]. k, n은 30 이하, d는 1로 추정할 때 알고리즘 처리 속도는 매우 빠를 것으로 보인다. 여기서 k는 추정 집단 수, n은 cardinality of testing data로 행의 수, d는 dimension이다. 두 번째 단점으로는 기존 데이터에 없는 새로운 유형의 데이터의 경우 연관관계를 찾기 힘들어 추천에 반영이 안 된다는 점이다. 하지만 음식 같은 경우 퓨전 음식을 제외하고는 새로운 음식보다 기존 음식을 찾는 경우가 대다수이다. 나중에 유저 층을 늘려나갈 때, 인도 음식, 중국 음식 등을 포함시킬 경우 이와 같은 단점이 부각될 수 있다. 세 번째 단점으로는 대중성이 높은 항목만을 추천해줄 가능성이 높다는 점이다. 추천 항목의 선택지가 많을 경우 이 같은 단점이 두드러지는데, 카테고리 태그를 이용해 추천된 카테고리 내에서 임의추출 한다면 이 단점을 보완할 수 있어 보인다.

---

[1]http://www.kocca.kr/insight/vol05/vol05_04.pdf pg1

[2]https://stats.stackexchange.com/questions/219655/k-nn-computational-complexity

CBF는 항목 자체를 분석해내서 유사한 항목을 추천해주는 알고리즘이다. 라면은 면 종류이고 저렴하고 조금 매운 음식이다. 이와 같이 항목을 분석해내고 유사한 면 종류이면서 저렴한 또 다른 음식인 우동을 추천해 줄 수 있다. 음악 추천의 CBF 알고리즘에서는 음악의 음 데이터를 분석해 어느 장르인지, 템포는 빠른지 느린지, 시끄럽거나 조용한 음악인지 분석했을 것이다. 하지만 음식의 경우 이를 분석하려면 음식의 성분, 요리법 등이 필요하다. 이와 같은 데이터는 구하기 힘들고 처리하기도 힘들어 보인다. WHO 보고서에 나오는 데이터를 이용하면 몇몇 음식의 성분 데이터는 구할 수 있다.[3] CBF 알고리즘 접근 방식은 음식에 대한 구체적인 데이터를 필요로 하는데 외국 음식을 포함한 여러 음식의 종류를 생각한다면 현실적으로 좋지 않아 보인다. 음식의 분석을 끝내고 나면, 사용자가 좋아한 음식을 바탕으로 비슷한 음식을 추천해 줄 수 있다. 이는 비교적 적은 량의 사용자 데이터를 이용하므로 초기 단계의 추천 알고리즘으로는 매우 적합하다! 초기 버전의 음식 추천 프로그램으로 간단하게 대표 음식 몇 가지를 수동 분석해보고(보통의 경우 군집분석(Clustering analysis), 인공신경망(Artificial neural network), tf-idf(term frequencyinversedocument frequency) 알고리즘을 이용한다) [4] 사용자의 YES나 NO 선택지를 바탕으로 해당 카테고리의 Weight를 조정해 추천을 해줄 수 있다. 음식의 수동 분석은 두 가지 행(음식의 종류, 음식의 맛)으로 간단히 하려고 한다.(1) 라면을 예를 들면 면, 매움 이다. 사용자가 라면에 YES 선택을 한다면 다음 음식 추천의 경우 면 카테고리와 매움 카테고리에 더 높은 weight을 가지고 추천해주면 된다.

마지막으로 MCF는 LDA(Latent Dirichlet Allocation), 베이지안 네트워크(Bayesian Network)와 딥 러닝 알고리즘을 이용해 데이터간 숨겨진 특성을 발견하고 모델을 만드는 방식이다. 예를 들면 라면, 불닭볶음, 우동이 있다면 라면과 불닭볶음을 좋아하는 사람의 경우 숨은 속성인 매운 맛을 좋아한다는 것을 알아낸다거나 라면과 우동을 좋아하는 사람의 경우 면을 좋아한다는 것을 알아내는 것과 같다. 이는 앞의 세가지 알고리즘을 이용해서 알아낸다. 보통은 숨은 속성이 무엇인지(매운 맛인지 면 종류인지)는 알고리즘이 알려주지는 않는다. 다만 이와 같은 속성이 있고 이와 같은 모델링을 통해 추천을 정교하게 해줄 수 있게 된다. 베이지안 네트워크와 딥 러닝 알고리즘을 살펴보자. 베이지안 네트워크는 데이터간 acyclic graphical model을 만들어주는데 비교적 적은 량의 데이터로도 만들 수 있다. 데이터가 작은 경우 모델의 성능은 ANN, 딥러닝에 비해 우수할 때가 많다. [5]딥 러닝의 경우 많은 분야에서 월등한 성능을 보이는데 학습에 요구되는 데이터 량이 무지 크다. 그러므로 음식 추천 프로그램의 경우 베이지안 네트워크가 더 적합하다. 추후 데이터가 많이 쌓이게 된다면 딥 러닝으로 모델링을 진행할 수 있을 것이다.

---

[3]https://raw.githubusercontent.com/CSIT-GUIDE/FYP-2016/master/1803_Kundan_FoodRecommedationSystemBasedOnContentFilteringAlgorithm.pdf

[4]http://www.kocca.kr/insight/vol05/vol05_04.pdf, pg3

[5]http://www.bayesia.com/hubert-bayesian-networks-and-small-data

프로그램 개발 전략

초기에는 쌓아둔 데이터가 없고 사용자가 없기 때문에 이를 대비해 프로그램을 만들어야 한다. 사용자가 늘어나고 쌓아둔 데이터가 많아진다면 추천 알고리즘에 변화를 줄 수 있다. 그 전까지는 비교적 간단한 추천 알고리즘을 이용하고 다른 서비스에 집중하는 것이 좋아 보인다. 일단 CBF 문단의 (1)에서처럼 음식의 간단한 수동 분석으로 추천 알고리즘을 개발한다. 음식의 종류 또한 대표적인 음식 20~30 가지로 한정할 것이다.

사용자들이 초기 버전의 프로그램을 이용하게 되면 데이터가 쌓일 것이다. 이 데이터를 관리하기 위한 데이터베이스를 체계적으로 만들 필요가 있다. 데이터베이스에 들어갈 데이터로는

1. 사용자 정보(id_code, id, passowrd, age, sex)

2. 음식 정보(food_id, category, food, type)

3. 음식선택 데이터(id_code, food_id, 선호여부(Yes or No), 시간, GPS, 날씨 및 기후) 가 있다.

날씨 및 기후는 외부에서 GPS와 시간을 이용해 끌어와야 할 데이터다.

사용자ID를 통해 1, 3을 Join할 수 있고 음식ID을 통해 2,3을 Join할 수 있다. 쌓여진 데이터가 사용자 별 30개이상, 그리고 50명이상의 사용자가 생기면 새로운 알고리즘을 개발해 적용해 볼 수 있을 것이다.

음식 속성: 달다, 느끼하다, 맵다, 짜다, 따뜻하다, 차갑다, 면, 밥, 인스턴트, 저렴하다, 시다, 고소하다

4. Experiment / 성능 평가 / 및 결론

- ❑ **가설 만들기(Hypotheses) / 가설검증/**

실제 데이터를 이용해서 Evaluation 하기

1. 사용자에게 임의로 추출된(중복o) 120개의 항목에 대해 Yes or No 응답데이터를 수집.

2. train, test set을 3대1의 비율로 나누기.

(1,2 실제로는 중복없이 30개의 항목에 대해 조사했고 12명을 대상으로 설문조사를 마쳤다.)

3. train set에서 사용자의 음식 preference weight list(1)를 알고리즘을 통해 뽑아낸다.

4. train set에서 yes or no 비율(2)을 계산한다.

5. test set에서 yes or no 비율(2)이 train set에서와 같게 나오게끔

6. preference weight list(1)에서 weight 기준점(3)을 설정.

7.이 기준(3)을 이용해 각 test set에서 yes, no 응답을 예측.

8. 예측된 값과 실제 값의 Accuracy(4)를 계산한다.

9 . 랜덤(yes or no비율(2)은 같게)으로 yes or no 응답했을 때의 accuracy(5)를 계산한다.

10. 두 accuracy 값(4),(5)을 비교, 이 차이값을 이용해볼 수 있다. 이 차이값이 클 수록 알고리즘의 성능은 좋은 것.

Mean Absolute Error의 경우 평점간 비교라 O,X 응답데이터에 적용이 안됨.

사용자한명(저)에 대해 알고리즘 평가가 이루어질 수 있고

같은 방식으로 사용자 n명에 대해서도 이루어 질 수 있다.

데이터 수집의 어려움으로 12명까지는 데이터 수집 했다.

n명의 경우 n개 case에 대해 성능평가후 평균값을 이용한다.

20대남성(제주변 사람들)에 한해 수집했고 이에 따라

음식 추천 알고리즘은 일반적인 20대 남성의 경우에 잘 적용될 것이다.

CBF 알고리즘:

1.속성별로 1/0 표시해준다.

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	food	느끼하다	맵다	짜다	고소하다	면	따뜻하다	달다	시다	밥	차갑다	저렴하다	인스턴트	고기	중식	일식	분식	한식	기타
2	소고기	1												1					
3	돼지고기	1												1					
4	닭갈비	1	1	1										1					
5	치킨	1		1	1									1					
6	스테이크	1												1					
7	짜장면			1	1	1									1				
8	짬뽕		1		1	1	1								1				
9	탕수육							1	1						1				
10	라멘		1			1	1									1			
11	조밥								1	1						1			
12	돈부리	1					1	1								1			
13	참치회										1					1			
14	우동				1	1	1									1			
15	라면		1			1	1					1	1				1		
16	떡볶이		1	1								1					1		
17	순대				1							1					1		
18	빙수							1			1						1		
19	케이크							1									1		
20	김밥				1					1		1					1		
21	만두	1										1					1		
22	보쌈			1	1													1	
23	냉면		1			1			1		1							1	
24	김치찌개		1	1			1			1								1	
25	곱창	1	1	1														1	
26	부대찌개		1	1			1			1								1	
27	파스타	1			1														1
28	피자	1		1									1						1
29	햄버거	1		1									1						1
30	쌀국수				1	1	1												1
31	빵	1										1							1

2. 속성의 개수를 세고, 속성값 / (속성 총갯수 ** 0.5) 로 변환해준다.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	food_id	category	food	느끼하다	맵다	짜다	고소하다	면	따뜻하다	달다	시다	밥	차갑다	저렴하다	인스턴트	고기	중식	일식	분식	한식	기타	합
2	1	고기	소고기	0.707107												0.707107						2
3	2	고기	돼지고기	0.707107												0.707107						2
4	3	고기	닭갈비	0.5	0.5	0.5										0.5						4
5	4	고기	치킨	0.5		0.5	0.5									0.5						4
6	5	고기	스테이크	0.707107												0.707107						2
7	6	중식	짜장면			0.5	0.5	0.5									0.5					4
8	7	중식	짬뽕		0.447214		0.447214	0.447214	0.447214								0.447214					5
9	8	중식	탕수육							0.57735	0.57735						0.57735					3
10	9	일식	라멘		0.5			0.5	0.5									0.5				4
11	10	일식	조밥								0.57735	0.57735						0.57735				3
12	11	일식	돈부리	0.5					0.5	0.5								0.5				4
13	12	일식	참치회										0.707107					0.707107				2
14	13	일식	우동				0.5	0.5	0.5									0.5				4
15	14	분식	라면		0.408248			0.408248	0.408248					0.408248	0.408248				0.408248			6
16	15	분식	떡볶이		0.5	0.5								0.5					0.5			4
17	16	분식	순대				0.57735							0.57735					0.57735			3
18	17	분식	빙수							0.57735			0.57735						0.57735			3
19	18	분식	케이크							0.707107									0.707107			2
20	19	분식	김밥				0.5					0.5		0.5					0.5			4
21	20	분식	만두	0.57735										0.57735					0.57735			3
22	21	한식	보쌈			0.57735	0.57735													0.57735		3
23	22	한식	냉면		0.447214			0.447214			0.447214		0.447214							0.447214		5
24	23	한식	김치찌개		0.447214	0.447214			0.447214			0.447214								0.447214		5
25	24	한식	곱창	0.5	0.5	0.5														0.5		4
26	25	한식	부대찌개		0.447214	0.447214			0.447214			0.447214								0.447214		5
27	26	기타	파스타	0.57735			0.57735														0.57735	3
28	27	기타	피자	0.5		0.5									0.5						0.5	4
29	28	기타	햄버거	0.5		0.5									0.5						0.5	4
30	29	기타	쌀국수				0.5	0.5	0.5												0.5	4
31	30	기타	빵	0.57735										0.57735							0.57735	3
32																						108

3. DF, IDF를 구한다.

Document Frequency ,Inverse Document Frequency

DF	12	9	10	9	7	8	4	3	4	3	6	3	5	3	5	7	5	5
IDF	0.954243	1.079181	1.033424	1.079181	1.188326	1.130334	1.431364	1.556303	1.431364	1.556303	1.255273	1.556303	1.334454	1.556303	1.334454	1.188326	1.334454	1.334454

```
DF = base.count()[2:]
IDF = DF.sum()/ DF
IDF = np.log10(IDF)
```

4. IDF 값 * 속성값 * USER 응답 값(1, 0으로 이루어짐)을 통해 USER1 row를 구할 수 있다.

이를 각 음식값과의 Sumproduct를 구하면 PREDUSER1 값을 구할 수 있다.

이 값을 normalize한다.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	food_id	category	food	느끼하다	맵다	짜다	고소하다	면	따뜻하다	달다	시다	밥	차갑다	저렴하다	인스턴트	고기	중식	일식	분식	한식	기타		USER1	PREDUSER1
2	1	고기	소고기	0.674751												0.943601							1	7.069594032
3	2	고기	돼지고기	0.674751												0.943601							1	7.069594032
4	3	고기	닭갈비	0.477121	0.539591	0.516712										0.667227							0	7.640040162
5	4	고기	치킨	0.477121		0.516712	0.539591									0.667227							1	7.799987708
6	5	고기	스테이크	0.674751												0.943601							1	7.069594032
7	6	중식	짜장면			0.516712	0.539591	0.594163									0.778151						0	6.020670612
8	7	중식	짬뽕		0.482625		0.482625	0.531435	0.505501								0.696						1	7.774393852
9	8	중식	탕수육							0.826398	0.898532						0.898532						1	5.630139312
10	9	일식	라멘		0.539591			0.594163	0.565167									0.667227					1	8.169232218
11	10	일식	초밥								0.898532	0.826398						0.770447					1	6.906240068
12	11	일식	돈부리	0.477121					0.565167	0.715682								0.667227					1	8.949128147
13	12	일식	참치회										1.100472					0.943601					1	6.471942723
14	13	일식	우동				0.539591	0.594163	0.565167									0.667227					1	8.329179764
15	14	분식	라면		0.440574			0.485132	0.461457					0.512463	0.635358				0.485132				1	8.060281886
16	15	분식	떡볶이		0.539591	0.516712								0.627636					0.594163				0	5.722949457
17	16	분식	순대				0.623066							0.724732					0.68608				0	5.592352523
18	17	분식	빙수							0.826398			0.898532						0.68608				1	6.060747035
19	18	분식	케이크							1.012127									0.840273				0	4.457092974
20	19	분식	김밥				0.539591					0.715682		0.627636					0.594163				1	6.404884271
21	20	분식	만두	0.550932										0.724732					0.68608				1	6.635884921
22	21	한식	보쌈			0.596647	0.623066													0.770447			0	4.667997778
23	22	한식	냉면		0.482625			0.531435			0.696		0.696							0.596786			1	7.923582029
24	23	한식	김치찌개		0.482625	0.462161			0.505501			0.640125								0.596786			1	6.756736456
25	24	한식	곱창	0.477121	0.539591	0.516712														0.667227			1	6.547634857
26	25	한식	부대찌개		0.482625	0.462161			0.505501			0.640125								0.596786			0	6.756736456
27	26	기타	파스타	0.550932			0.623066														0.770447		1	7.326274145
28	27	기타	피자	0.477121		0.516712									0.778151						0.667227		0	6.723188862
29	28	기타	햄버거	0.477121		0.516712									0.778151						0.667227		1	6.723188862
30	29	기타	쌀국수				0.539591	0.594163	0.565167												0.667227		1	7.768455016
31	30	기타	빵	0.550932										0.724732							0.770447		1	7.16929423
32																								
33																								
34			USER1	5.585536	2.967629	2.012297	3.264053	3.330491	3.733126	2.368478	2.493063	2.182205	2.695003	2.589563	1.413509	3.498031	1.594531	3.715729	2.451455	1.860799	2.875348			

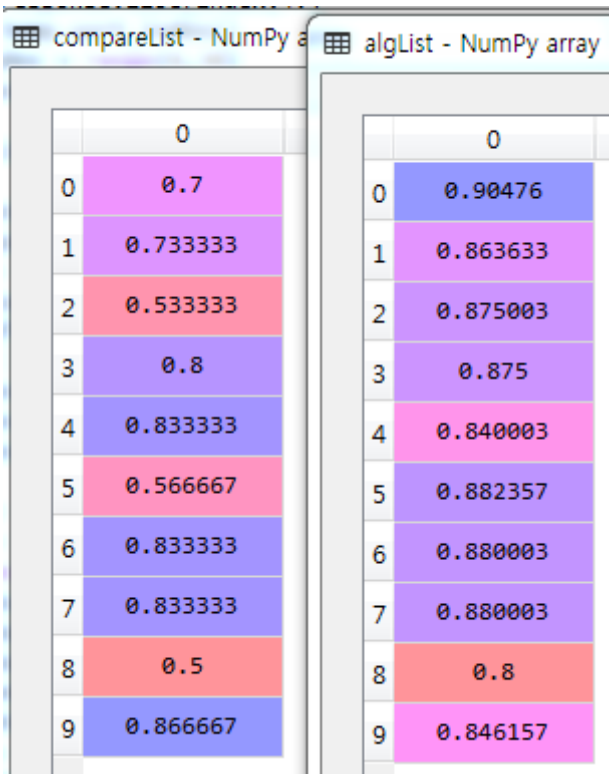
```
base["sum"] = base.count(axis=1)-2

for column in base.columns[2:-1]:
    base[column] = base[column]/ base["sum"]**0.5

base = base.iloc[:,:-1]

for index in base.index:
    base.iloc[index,2:] = base.iloc[index,2:] * IDF
```

```python
import pandas as pd
import numpy as np
from pathlib import Path
home = str(Path.home())
PATH = home + "\\Desktop\\ee_grad_project\\"

basePath = PATH + "dataFile\\weighted2.xlsx"
responsePATH = PATH + "dataFile\\food_selection_responses2.xlsx"

weight = pd.read_excel(basePath)
response = pd.read_excel(responsePATH)

responseList = []
for index in response.index:
    temp = response.iloc[index,4:]
    temp = temp.sort_index()
    temp.index = range(0,30)
    responseList.append(temp)

preferenceList = []

for response1 in responseList:
    w = responseList[1]
    profile = []
    for column in weight.columns[2:]:
        profile.append((weight[column] * w).sum())

    preference = (weight.iloc[:,2:] * profile).sum(axis=1)
    preference = preference / preference.sum()
    preferenceList.append(preference)
```

5. 알고리즘의 성능 평가를 시행해본다.

```python
compareList = []
pointList = []
for i in range(len(responseList)):
    tempData = weight.iloc[:,[0,1]].copy()
    tempData["response"] = responseList[i]
    tempData["preference"] = preferenceList[i]
    positiveCount = tempData["response"][tempData["response"]==1].count()
    positRate = positiveCount / tempData["response"].count()
    calData = tempData.sort_values("preference",ascending=False)[:positiveCount]
    evaluateRate = calData["response"][calData["response"]==1].count() / positiveCount
    finalPoint = round(evaluateRate - positRate,5)
    compareList.append(positRate)
    pointList.append(finalPoint)

compareList = np.array(compareList)
pointList = np.array(pointList)
algList = compareList + pointList
#pointList = pd.Series(pointList[pointList!=0])
#pointListSamp = pointList.sample(frac=0.7)
print(round(pointList.mean(),4) * 100,"%",sep="")
```

설문 참여자의 모든 응답을 이용해 accuracy 값을 계산한다.



compareList - NumPy a

	0
0	0.7
1	0.733333
2	0.533333
3	0.8
4	0.833333
5	0.566667
6	0.833333
7	0.833333
8	0.5
9	0.866667

algList - NumPy array

	0
0	0.90476
1	0.863633
2	0.875003
3	0.875
4	0.840003
5	0.882357
6	0.880003
7	0.880003
8	0.8
9	0.846157

yes / total의 비율이다.

전체중 몇 %를 yes로 답했는지를 compareList에 저장했다.

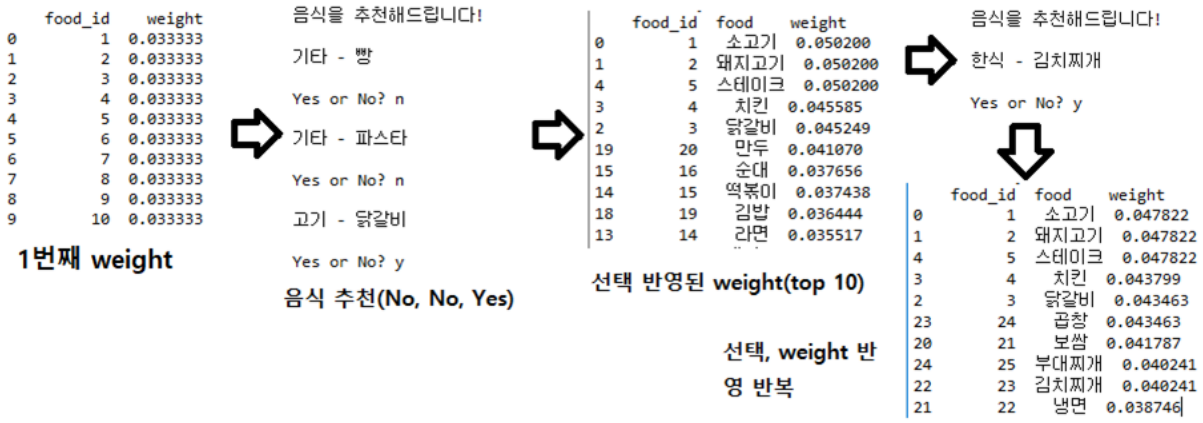
4, 2명의 설문 응답자의 경우 30가지 모든 음식에 yes를 답해 의미가 없는 데이터가 되었다. (이를 제외함)

알고리즘을 거쳐 좋아하는 음식을 예측해본 경우 맞은 비율을 오른쪽 리스트에 저장했다.

```
In [5]: "%.4f" % compareList.mean()
Out[5]: '0.7200'

In [6]: "%.4f" % algList.mean()
Out[6]: '0.8647'
```

평균적으로 14.47% 비율이 올라갔으며, index 9(2% 감소)빼고는 모든 값이 올랐다.

content의 요소를 수정하면 이 비율이 증가하며 category(고기, 중식, 한식, 기타, 일식, 분식) 이 6가지만을 이용했을 경우 정확도는 11.03% 올랐다. 12명의 데이터만을 이용하는 것이라 overfitting의 문제가 생길 확률이 높다. 그래서 가지고 있는 모든 속성을 이용한 14.47% 알고리즘 성능을 최종 성능으로 봐야 할 것이다.



weight을 가지고 음식을 추천해준다. yes 답변을 할 때까지 추천을 해주며, 답변에 따라 weight를 조정한다.

이번 졸업프로젝트로 음식 추천 프로그램을 개발했다. 추천 알고리즘은 최대한 간단히 만들고 날짜, GPS 등을 이용해 편의기능 개발에 집중하려 했으나 개발 방향을 바꾸어 추천 알고리즘의 성능에 집중하기로 했다. 개발을 하면서 db 또한 직접 만들어 보고 중복된 데이터가 없도록 효율적으로 관리하는 법도 배웠다. 또한 CBF에 대해서 확실히 알게 됐고, 실제 데이터 수집부터 개발까지 혼자 힘으로 할 수 있어 의미 있었다.

5. Reference 및 별첨

http://www.kocca.kr/insight/vol05/vol05_04.pdf pg1

https://stats.stackexchange.com/questions/219655/k-nn-computational-complexity

https://raw.githubusercontent.com/CSIT-GUIDE/FYP-2016/master/1803_Kundan_FoodRecommedationSystemBasedOnContentFilteringAlgorithm.pdf

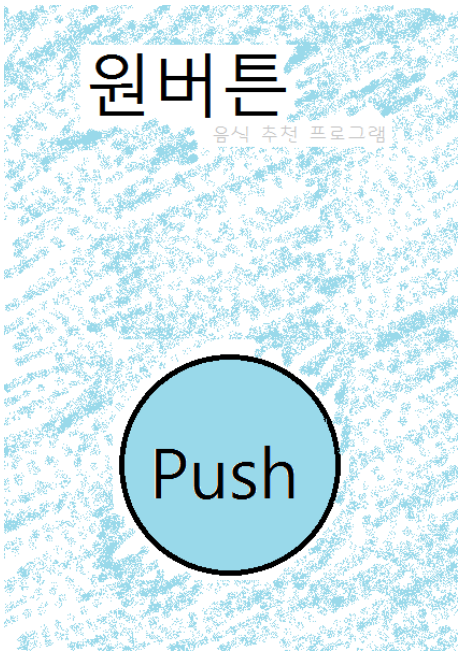
http://www.kocca.kr/insight/vol05/vol05_04.pdf, pg3

http://www.bayesia.com/hubert-bayesian-networks-and-small-data

(1)https://www.nextobe.com/#!머신러닝-엔지니어가-알아야-할-10가지-알고리즘/mhqg1/5902967aaf7c916ff7ea682c로지스틱 회귀

https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/

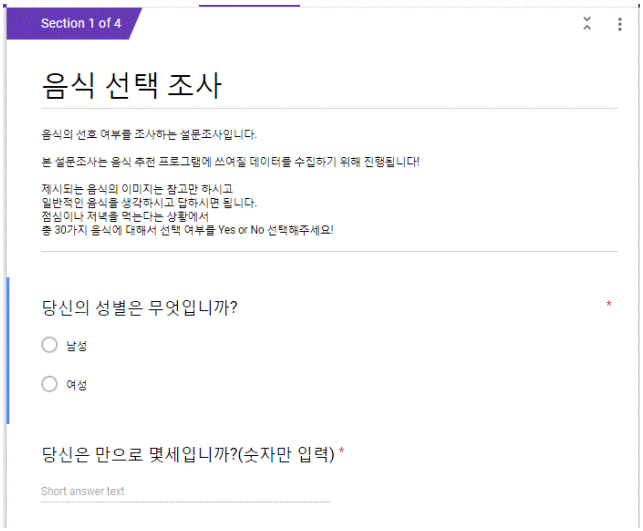
앱으로구현시 예시 화면



메인화면

선택 화면

지도 보여주기



설문조사1

설문조사2

6. 별첨(Source, data files description)

#data files description

#base.xlsx, base2.xlsx: 음식의 속성을 적은 테이블

#food_data.xlsx,food_data2.xlsx: food_id, category, food, type 의 테이블

#food_recommendV2.xlsx :카테고리별 음식종류 테이블

#food_selection_responses1.xlsx ,food_selection_responses2.xlsx: 음식 선호 설문조사 응답표

#id_db_log.xlsx : id, password 로그(age, sex도 저장)

#output_log.xlsx, output_log2.xlsx: id_code, food_id, choice, 음식 추천후 추천의 선택지 로그 기록

#weighted.xlsx, weighted2.xlsx, : base.xlsx 를 CBF 알고리즘을 통해 변환한 table. 속성별 weight가 table로 저장되어 있다.

#Source code

#pickFunctionV3.py

yes_or_no loop 함수, 음식을 하나 추출하는 모듈

#makeNewAccountAndLoginV1.py

새로운 계정을 만들고 로그인하는 기능

#pickFoodWithGivenWeight.py

주어진 weight를 가지고 음식을 고른다.

#foodDataWeightFromLogV2.py

로그에서 food weight를 계산해낸다.

#pickFunctionWithWeightV1.py

foodDataWeightFromLogV2.py에서 calWeightFromLog를 가져온다.

weight를 가져와 이 weight 비율대로 음식을 골라준다.

#mainV1.py

db를 load하고, 음식을 추천해준다. 코드를 통합함.

#baseWeightCal.py

base.xlsx 를 이용해 weight.xlsx를 만드는 코드

#weightUserCal.py

weight.xlsx에서 음식별 weight를 가지고 온다. food_selection_responses2.xlsx 에서 설문지 응답표를 가지고 오고 이를 바탕으로 설문응답자의 음식별 preference 점수를 계산한다. 이 점수를 가지고 추천 성능을 판별한다.