**졸업 프로젝트- 원버튼 음식 추천 프로그램**

2013171046 전전전 심재훈

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 이지선다를 하는 것은 비교적 수월하다. 그래서 나는 Yes or No로 선택할 수 있게 음식 메뉴 하나를 추천해주는 프로그램을 만들고 싶다. 선택 가능한 메뉴를 등록해놓고 여기서 임의 추출해 제시해주는 프로그램은 그 자체만으로도 유용할 수 있다. 이를 단순임의추출에서 더 나아가 사용자에 따라 각 메뉴에 가중치를 두고 이를 고려한 표본 추출하는 식으로 사용자의 만족도를 높일 수 있을 것이다.

우리는 점심시간과 저녁시간에 따라 입맛이 다를 수 있는데 이는 시간(Timestamp)을 이용해 점심과 저녁에 따라 메뉴를 추천해 줄 수 있다. 또한 비가 오면 파전을 먹거나 더운 날 냉면이나 삼계탕을 먹듯, 날씨에 따라 메뉴가 달라지기도 한다. 이를 프로그램 실행 당시 날씨 데이터에 따라 메뉴 추천을 해줄 수 도 있다. 이외에도 나이와 성별에 따라 추천을 달리 해줄 수 있다.

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. 원버튼 음식 추천 프로그램(이하 원버튼)의 경우 음악 추천 프로그램과 같은 방식을 이용할 것이다.

**구현 방법**

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 이용시 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 추천받는 경우 A를 추천받을 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]](#footnote-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]](#footnote-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 음식를 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.

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]](#footnote-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 frequency inverse document frequency) 알고리즘을 이용한다) [[4]](#footnote-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]](#footnote-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.

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 종류또한 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(사용자ID, 비밀번호, 나이, 성별)

2. 음식 정보(음식ID, 음식명, 카테고리1, 카테고리2,...)

3. 음식선택 데이터(사용자ID, 음식ID, 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 사용자별 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

**평가 방법**

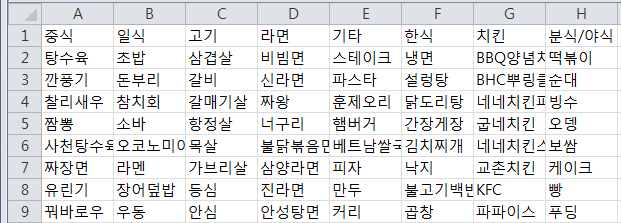
평가 방법으로는 보통의 경우 train dataset 과 test dataset으로 나누어 train dataset에서 나온 모델을 바탕으로 test dataset에서 어느 정도 성능이 나왔는지를 수치화해 측정한다. 그러나 우리는 이를 위한 dataset이 없기 때문에 다른 평가 방법을 이용하기로 한다. 30번 이상의 추천에 Yes or No에 응답한 사용자에 한해 10번의 테스트 항목 추천을 해준다. 이 항목 중에 사용자가 10개 중 몇 개를 Yes로 답했는지의 비율을 평가 방법의 척도로 이용한다. 이 중 중복된 항목의 경우 이를 제외하고 비율을 계산한다. 항목 추천을 단순임의추출로 했을 때의 경우의 비율을 계산해 앞의 값과 비교한다.

**간단한 코드**

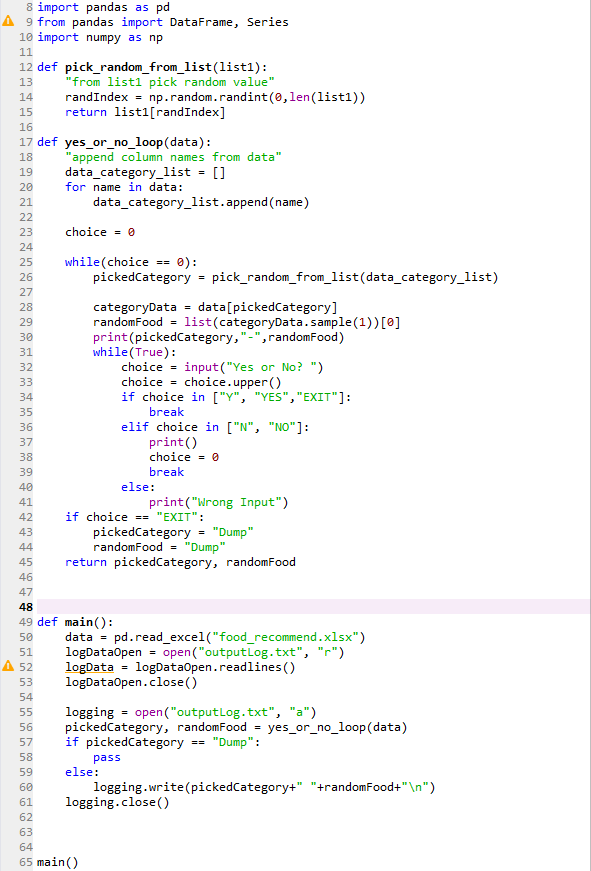
아래는 프로그램을 간략히 구현해 본 것이다.



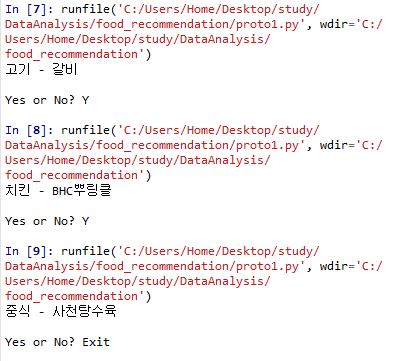
음식 목록의 바탕이 될 만다라트 (출처는 못구했다)



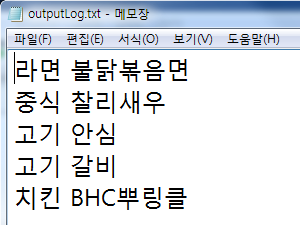
음식 만다라트를 테이블로 작성



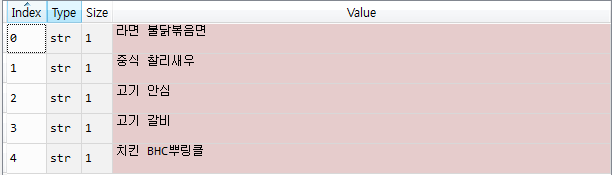
간단한 임의 추출 및 로그 저장 프로그램



실행 결과

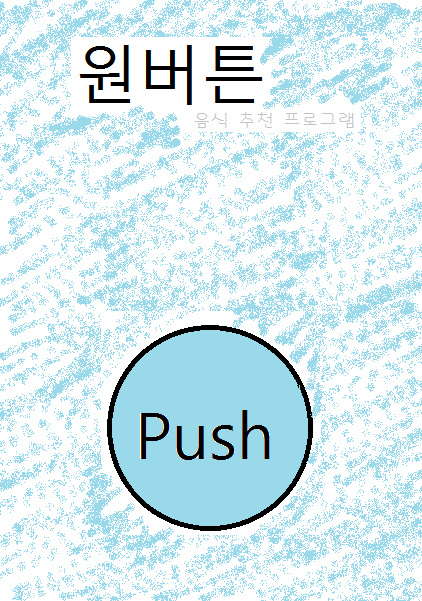


로그 목록



로그 데이터

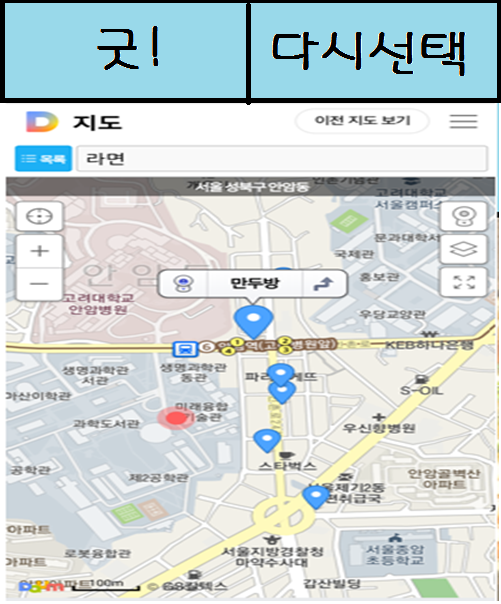
구현 화면 예시



메인화면



선택 화면



지도 보여주기

**Reference:**

(1)https://www.nextobe.com/#!머신러닝-엔지니어가-알아야-할-10가지-알고리즘/mhqg1/5902967aaf7c916ff7ea682c 로지스틱 회귀

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. http://www.kocca.kr/insight/vol05/vol05\_04.pdf pg1 [↑](#footnote-ref-1)
2. https://stats.stackexchange.com/questions/219655/k-nn-computational-complexity [↑](#footnote-ref-2)
3. https://raw.githubusercontent.com/CSIT-GUIDE/FYP-2016/master/1803\_Kundan\_FoodRecommedationSystemBasedOnContentFilteringAlgorithm.pdf [↑](#footnote-ref-3)
4. http://www.kocca.kr/insight/vol05/vol05\_04.pdf, pg3 [↑](#footnote-ref-4)
5. http://www.bayesia.com/hubert-bayesian-networks-and-small-data [↑](#footnote-ref-5)