오피니언 마이닝과 머신러닝을 이용한 페이스북 인기 게시물 예측 시스템

안현우, *문남미
호서대학교, 컴퓨터소프트웨어과
grabroe@naver.com, *mnm@hoseo.edu

Prediction System of Facebook's popular post using Opinion Mining and Machine Learning

Hyeon-woo An, *Nammee Moon
Hoseo University

요 약

페이스북 SNS 플랫폼에서 제공하는 데이터 수집 프로토콜을 이용해 콘텐츠들의 인기 점수와 사용자 의견들을 수집하고 수집된 정보를 가공하여 기계학습을 진행한다. 오피니언 데이터를 학습함으로 인해 인간의 관점을 모방하게 되며 결과적으로 콘텐츠의 질을 판단하는 요소로써 작용하도록 한다. 데이터의 수집은 페이스북 측에서 제공하는 Graph API 와 Python 을 이용하여 진행한다. Graph API 는 HTTP GET 방식의 프로토콜을 이용하여 요청 하고 JSON 형식으로 결과를 반환한다. 학습은 Multiple Linear Regression 과 Gradient Descent Algorithm(GDA)을 사용하여 진행한다. 이후 학습이 진행된 프로그램에 사용자 의견 데이터를 건네주면 최종 인기 점수를 예측하는 시스템을 설명한다.

1. 서론

페이스북과 같은 SNS 플랫폼에서 인기 게시물이 갖는 파급력은 상당히 높다. 모 브랜드가 SNS 에서 시작된 트랜드를 마케팅하기 위해 '트렌드 즐겨찾기' 라는 특화존을 설치한 것처럼 작금의 소비 트랜드를 좀 더 빠르게 알 수 있다는 것은 경쟁력의 상승이라고 보아도 무방하다[1]. 그러나 이미 인기 있는 콘텐츠들을 통해 특정 소비 트랜드가 시작되었음을 아는 것은 효용성이 떨어진다. 이러한 상황 속에서 인기 콘텐츠를 미리 알 수 있다는 것은 트랜드의 시작을 예측할 수 있다는 말과 같다. 하지만 콘텐츠의 질적인 요소를 판단하기 위해서는 인간의 관점이 필요한데 인력을 사용하여 플랫폼에 등록된 방대한 콘텐츠들을 선별하는 방법은 매우 비효율적이다.

이러한 문제점을 해결하고 인기 게시물을 예측하기 위한 방법과 그를 적용한 프로그램을 설명한다.

2. 본론

인기 게시물을 예측하기 위해서는 먼저 콘텐츠의 품질을 나타내는 척도가 있어야 한다. 이는 페이스북의 좋아요 수와 댓글 수를 합친 수(이하 관심 수)를 사용한다. 공유 수의 경우 테스트 결과 앞의 두 점수와 독립적으로 높거나 낮은 결과를 보여 제외하였다.

학습은 표본이 갖는 특징을 토대로 목적 값을 유추하기 위한 변수를 구해가는 과정이다. 여기서 목적하는 값은 관심 수이며 표본의 특징으로 사용되기 위한 아래와 같은 제약사항을 두어 탐색했다.

- 수집 가능한 항목이어야 한다.
- 관심 수와 연관된 항목이어야 한다.
- 시간 흐름에 따른 실시간 분석은 사용할 수 없다.
- 수로 표현 가능한 항목이어야 한다.

이러한 특징을 모두 만족하는 항목으로 콘텐츠 게시 시간 이후 특정 시간 동안의 댓글 등록 수를 선택했다. 지정된 항목은 예측에 필요한 특정 시간 길이의 배열로 저장되고, 학습은 Multiple Linear Regression[2]과 Gradient Descent Algorithm(GDA)[3]을 사용하여 진행된다.

학습에 앞서 예측을 위한 Hypothesis 식을 설정한다. 이 식을 통해 나온 값은 예측값을 의미하며 식은 아래와 같다.

$$H(X) = XW \quad (1)$$

X 는 예측의 기반이 되는 값, 시간에 따른 댓글 등록 수 배열을 뜻한다. [y][시간]의 길이를 갖는다. W 는 가중치 값이며 첫 값으로는 임의의 값이 들어가며 행렬 곱을 위해 [시간][y]의 길이를 갖는다.

H(X)를 통한 예측 시 오차(cost 또는 loss)값을 계산하기 위한 함수 cost 를 만든다.

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left(H\left(x_1^{(i)},x_2^{(i)},x_3^{(i)}\right) - y^{(i)}\right)^2 \quad (2)$$

표본 전체의 예측값과 실제 목적 값인 y 와의 차이 평균을 구하는 함수가 cost 라 할 수 있다. 가중치 W 로 인한 예측값이 목표값 y 와 연관되도록 오차 최소화(minimize) 과정을 실행해야 한다. 이는 GDA 를 이용하여 설정할 수 있다. GDA 는 cost 함수 값과 W 로 표현된 그래프에서 기울기를 구해 cost 가 더 작은 값으로 수렴할 수 있도록 다음 W 값을 설정해주는 알고리즘이다.

수집한 표본 데이터를 사용해 댓글 등록 수 배열을 X 에 넣고 관심 수를 y 에 넣어 training 시킨다.

이때 학습 반복 횟수와 GDA 의 알파 상수를 조절해 학습 효율을 조절할 수 있다.

위 과정 중 초기 Hypothesis 식을 설정하는 과정을 제외한 과정은 반복되는 과정으로써, 주기가 지날수록 minimize 과정에 의해 cost(W)값이 줄어들며 W 값은 임의의 값에서 점점 실제 목적 값과 연관된 배열로 바뀌어 나간다.

데이터의 수집은 페이스북 측에서 제공하는 Graph API 와 Python 을 이용하여 진행한다. Graph API[4]는 HTTP GET 방식의 프로토콜을 이용하여 요청 하고 JSON 형식으로 결과를 반환한다. URL 요청 구조는 다음 그림과 같다.

```
(1)
https://graph.facebook.com/v2.9/'Page ID'/feed
    (2)?access_token='AppID및 Access Token'
    (3)&fields=
            message,
            created_time,
            likes.limit().summary(true),
            comments.limit(200).summary(true).filter(stream),
            created_time,
            reactions.summary(true).filter(stream),shares.summary(true)
    (4)&since=2017-01-01
      &until=2017-06-07
```

그림 1. 페이스북 요청 URL 구조

(1):수집하려는 페이지의 ID.
(2):app id 토큰과 access 토큰 정보.
(3):수집하고자 하는 필드.
(4):수집하고자 하는 기간.

위와 같은 요청 URL 을 보내면 아래와 같이 항목 별로 분리된 JSON 구조의 결과를 반환 받는다.

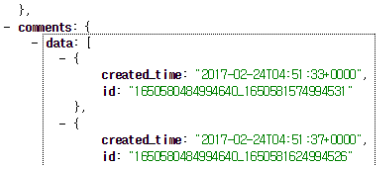
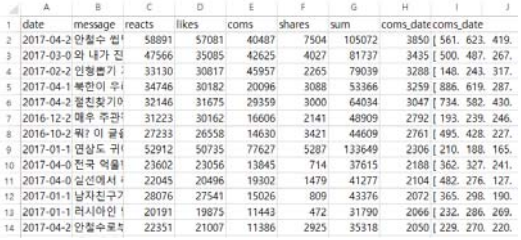


그림 2. JSON 구조의 반환 결과

반환된 결과를 확인해 보면, Comments 항목에 created_time 이라는 게시 시간이 기록되어 있음을 알 수 있다. 이제 이 시간을 이용해서 콘텐츠 게시 이후 24 시간 동안의 댓글 등록 상황을 비롯한 기타 정보들을 파싱하여 csv 파일로 저장하여야 한다. 이때 수집 속도를 늘리기 위해 약간의 트릭이 필요한데, 댓글 조사에 있어서 step 을 적용하는 것이다. 예를 들어 그림 2 와 같은 콘텐츠의 경우 2929 개의 댓글이 등록되어 있는데 이를 전부 요청하고 파싱한다면 시간이 꽤나 소비될 것이다. 이에 step 이라는 상수를 설정하고 해당 값만큼 건너뛰면서 조사하는 방법을 사용한다면 보다 적은 시간 내에 결과를 출력할 수 있을 것이다.

이 step 상수는 다른 의미로도 필요하다. 바로 댓글 비중에 따른 가중치 역할로써 조사 빈도를 늘리거나 줄일 필요가 있기 때문이다. 약 1000 개 가량의 표본 중 댓글 수가 좋아요 수에 비해 아주 높거나 낮은 콘텐츠들을 주관에 따라 해석한 결과 내용이 자극적인 요소를 띌 경우 관심 표현인 좋아요 기능보다 댓글 등록을 더 우선시 하는 현상이 있었다. 이런 경우 콘텐츠의 질을 판단하기 위해 다른 평균적인 콘텐츠에 비해 더 낮은 빈도로 조사하도록 하고 반대의 경우 더 높은 빈도로 조사하여 두 양상 사이의 균형을 맞추어 줄 수 있을 것이다.

좋아요 수와 댓글 수, 시간별 댓글 수 등 콘텐츠의 각 정보를 수집한 결과 아래와 같은 csv 파일을 얻을 수 있다.



	A	B	C	D	E	F	G	H	I	J
1	date	message	reacts	likes	coms	shares	sum	coms_date	coms_date	
2	2017-04-2	안철수 씹	58891	57081	40487	7504	105072	3850	[561.	623. 419.
3	2017-03-0	와 내가 진	47566	35085	42625	4027	81737	3435	[500.	487. 267.
4	2017-02-2	인형뽑기	33130	30817	45957	2265	79039	3288	[148.	243. 317.
5	2017-04-1	북한이 우	34746	30182	20096	3088	53366	3259	[886.	619. 287.
6	2017-04-2	절친찾기	32146	31675	29359	3000	64034	3047	[734.	582. 430.
7	2016-12-2	매우 주관	31223	30162	16606	2141	48909	2792	[193.	239. 246.
8	2016-10-2	뭐? 이 글	27233	26558	14630	3421	44609	2761	[495.	428. 227.
9	2017-01-1	연상도 귀	52912	50735	77627	5287	133649	2306	[210.	188. 165.
10	2017-04-0	전국 역올	23602	23056	13845	714	37615	2188	[362.	327. 241.
11	2017-04-0	실센에서	22045	20496	19302	1479	41277	2104	[482.	276. 127.
12	2017-01-1	남자친구	28076	27541	15026	809	43376	2072	[365.	298. 190.
13	2017-01-1	러시아인	20191	19875	11443	472	31790	2066	[232.	286. 269.
14	2017-04-2	안철수로	22351	21007	11386	2925	35318	2050	[229.	270. 220.

그림 3. JSON 구조의 반환 결과

이제 상기에 서술한 학습 흐름에 따라 파싱 된 파일의 정보 중 coms_date 라는 시간별 댓글 등록 수 배열이 X 로써 자리하게 된다. 만약 조사 시간을 24 시간이라 한다면 목적 값과 예측값은 관심 수 하나만을 표현하므로 X 의 크기는 [1][24]가 될 것이며 W 의 크기는 행렬 곱을 위해 [24][1]이 될 것이다. 학습 시에는 y 값인 sum 에 해당하는 배열이 cost 비교를 위한 리스트로 제공될 것이고, 가중치인 W 에는 X 와 같은 크기의 임의의 배열이 들어가고, 학습이 끝난 이후 W 는 얻어진 가중치 값을 통해 예측을 진행할 것이다. GDA 의 알파 값은 cost 함수의 결과에 따라 경험적으로 얻어질 것이다.

학습은 Python 과 Tensorflow[5] 라는 인공지능 라이브러리를 이용해 진행한다. 또 csv 로 저장된 학습 데이터를 읽어와야 하기에 csv 라이브러리와 파싱에 필요한 re 라이브러리가 필요하다.

학습의 핵심 코드는 아래와 같다.

```
tf.set_random_seed(777)  # for reproducibility
times = 24
X = tf.placeholder(tf.float32,shape = [None,times])
Y = tf.placeholder(tf.float32,shape = [None,1])
W = tf.Variable(tf.random_normal([times,1]), name='weight')
hypothesis = tf.matmul(X, W)
cost = tf.reduce_mean(tf.square(hypothesis - Y))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.000001)
train = optimizer.minimize(cost)
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
```

그림 4. 학습 핵심 코드

코드를 설명하자면 난수 시드를 설정하고, 조사한 특정 시간이 24 시간이었음을 명시하고 X, Y, W 를 설정한다. placeholder 는 이후 사용자 피드백으로 대체될 수 있는 공간을 설정하는 기능이다. W 에 설정된 Variable 인자들에 의해 W 는 [times,1] 크기 배열에 임의의 값을 저장하게 된다. 이후 hypothesis 를 통해 예측 함수를 설정하고, cost 함수를 통해 오차 평균을 구하는 식을 설정, GDA 와 알파 값 0.000001 을 사용하는 optimizer 또한 설정하였다. 이후 optimizer.minimize(cost)를 등록함으로써 cost 를 minimize 하는데 사용될 것이다.

예측에 더 적합한 가중치 배열이 학습됨에 따라 W 값은

임의의 값이 아닌 학습 결과로써의 가중치 배열이 들어갈 것이다.

tensorflow 는 각 노드라고 불릴 수 있는 변수, 개체들이 연결되는 형태로 설계하고 어떤 노드를 sess.run 시키면 하위 노드들이 동작하며 회귀적으로 상위 노드에 값을 반환하는 식으로 설계되어 있다.

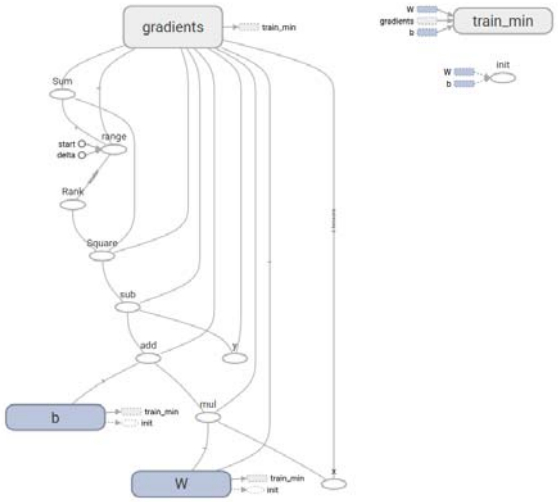


그림 5. Tensorflow 동작 방식

즉 위 핵심 코드에서 sess.run(train)을 통해 최상위 노드를 실행시킨다면 GDA 인 optimizer 가 실행되고, cost 함수와 hypothesis 가 실행되며 마지막으로 X, W 에 대한 값을 hypothesis 에 반환하며 계산을 시작한다. 이때, placeholder 로 설정된 X 와 Y 에 대한 정보를 트레이닝 데이터에서 피드백형식으로 제공해주어야 한다.

학습이 끝나면, 즉 적합한 가중치 배열을 얻고 예측을 위한 동작만 필요할 때에는 train 이 아닌 hypothesis 만을 실행하면 된다. 이 때문에 코드는 학습 코드와 약간 다른 모습을 갖는다. 예측에 필요한 가중치가 준비되어 있어야 하며 기존 Y 값에 대한 피드백이 필요치 않게 된다.

```
tf.set_random_seed(777)  # for reproducibility
times = 24
W = [[ 23.21560097],[ 14.66111851],[ 12.88269615],[ -0.23741183],[  9
W = tf.Variable(W, name='weight')
X = tf.placeholder(tf.float32,shape = [None,times])
hypothesis = tf.matmul(X, W)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

그림 6. 예측 핵심 코드

3. 실험 및 결과

학습과 예측에 필요한 댓글 표본 수가 100 을 넘지 못하면 표본으로 사용하기 힘들다고 판단하여 표본에서 제외시켰다. 학습은 cost 함수의 결과값을 줄여나가는 방식으로 반복 진행하였으며, 예측 실험 도중 댓글 비중에 따라 예측 점수에 가중치를 부여하는 것이 높은 예측률을 보임을 확인했고 가장 적합한 가중치 값을 구하기 위해 시뮬레이션 코드를 작성하여 실행시켜 본 결과 댓글 비중이 1.7~3 일 때 가장 높은 효율을 보인다는 사실과 함께 아래와 같은 가중치 식을 얻을 수 있다.

$$mulx = |com_{rate} \quad 2.45| \; | \; 0.53 \qquad (3)$$

실험에 필요한 데이터는 6 개의 서로 다른 페이스북 페이지에서 수집되었고 최종 관심 수라 할 수 있을 만큼의 시간을 3 개월이라 판단했으며 그 전까지의 데이터만을 다루었다. 1023 개의 레코드로 이루어져 있으며 평균 관심 수는 6971.496, 표준편차는 4477.408, 레코드들의 관심 수 비율은 아래 그래프와 같다.

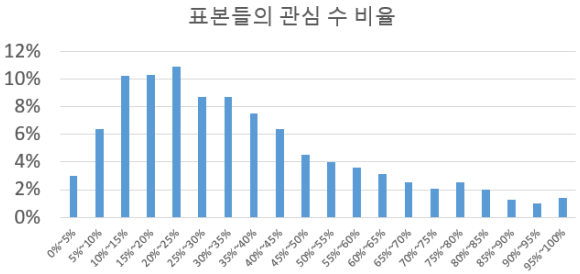


그림 7. 표본들의 관심 수 비율 그래프

만약 인기 게시물의 관심 수를 10000 이라 제한하면, 테스트 결과 화면은 아래와 같다. 예측 수에 대한 오차율은 29.725%이며 총 228 개의 인기 게시물 중 86.842%에 해당하는 인기게시물을 선정해 낼 수 있었다.

```
-----
avg diff :  0.29725298003996925
hit rate :  0.868421052631579  suc :  198      fail :  30
>>>
```

그림 8. 결과 화면

```
        if int(hy_val[0][i][0]) > 10000 and int(org_arr[i]) > 10000:
            suc = suc + 1
        elif int(org_arr[i]) > 10000:
            fail = fail + 1

print ('avg diff : ', sumofdiff/len(hy_val[0]))
print ('hit rate : ',suc/(fail+suc), ' suc : ', suc, 'wtfail : ', fail)
```

그림 9. 인기 게시물의 관심 수 제한, 출력 코드

오차율은 1023 개 각각에 대한 예측 오차율로, 인기게시물을 포함한 다른 모든 게시물에 대한 오차율 평균이다.

4. 결론

예측하기 위한 24 시간이 선행적으로 필요하고, 변화 추이를 뚜렷이 관측하기 위해 댓글 표본 수가 어느 정도 이상이어야 한다는 점은 명백한 단점이지만 사람이 하기 힘든 예측을 오차율 30% 내외로 할 수 있다는 점은 명백한 장점이라 볼 수 있다.

비록 본 논문에서는 공유 수를 다루진 않았지만, 테스트 중 공유 수가 두 개의 합산보다 높은 특이 케이스가 몇 발견되었고 이에 대해 인기 게시물이 아니라고 단정할 수 없으므로 이러한 예외를 처리하기 위한 방법이 필요할 것이다. 또한, 가중치 W 는 각각의 페이지 성향에 많은 영향을 받는 모습을 보였는데, 이에 대해 뚜렷한 해법이나 이해가 따라오지 못했다. 만약 각 페이지 성향에 맞는 가중치를 따로 구하고 적용할 수 있다면 더 높은 정확도를 기대할 수 있을 것이다.

[이 논문은 2017 년 대한민국 교육부와 한국연구재단의 중견연구자지원사업의 지원을 받아 수행된 연구(한국연구재단-2017 년-2017008886)임]

참고문헌

[1] 고서진, and 이재영. "SNS 네트워크

특성이 소비자의 감정 반응 및소비 행동에 미치는 영향." 소비문화연구 19 (2016): 133-155.

[2] Grégoire, G. "Multiple linear regression." European Astronomical Society Publications Series 66 (2014): 45-72.

[3] Govan, Anjela. "Introduction to optimization." North Carolina State University, SAMSI NDHS, Undergraduate workshop. 2006.

[4] Weaver, Jesse, and Paul Tarjan. "Facebook linked data via the graph API." Semantic Web 4.3 (2013): 245-250.

[5] Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).