TEAM13 Final Report

강동헌
SungKyunKwan University, Undergraduate

송은기
SungKyunKwan University, Undergraduate

윤영진
SungKyunKwan University, Undergraduate

이시윤
SungKyunKwan University, Undergraduate

ABSTRACT

대한민국에서 가장 유명한 질문답변 서비스는 네이버에서 제공하는 지식인 서비스이다. 지식인 서비스는 2022 년 서비스 20 주년을 맞이했을 정도로 오래되었고 그만큼 다양한 사용자 편의를 지원하지만 질문에 대한 예상 답변시간을 알려주는 서비스는 존재하지 않는다. 이에 본 프로젝트에서는 지식인 답변시간에 영향을 주는 변수들을 설정해서 지식인 게시글의 답변이 작성되는 시간을 예측하는 모델을 통해서 변수들의 영향력을 계산하였다. 계산결과 제목과 본문의 내용 보다는 답변시에 받게 되는 내공과 질문글의 시간대 및 답변이 달리는 시간대가 주된 의미를 가진다는 것을 알 수 있었다.

1. 서론

네이버 지식인은 대한민국에서 가장 활발하고 유명한 Q&A 서비스이다. 미국의 경우 Quora 라는 지식인과 유사한 서비스가 있지만 2002 년에 첫 서비스를 시작한 지식인에 비해 많이 늦은 2009 년부터 시작됐다.

지식인의 작동 방식은 질문자가 궁금증을 질문 글로 작성하고 '내공'이라는 포인트를 해당 질문 글의 답변 보상으로 설정하면, 사용자들이 해당 질문 글에 답변을 달고, 질문자가 채택하여 질문자 채택 답변으로 선정되면 '내공'이라고 불리우는 포인트를 가져가는 서비스이다.

지식인은 2022 년 기준으로 서비스한지 20 년이된 장수 서비스인 만큼 다양한 사용자 편의성을 갖추고 있다. 작성자가 질문글을 작성하면 자동으로 카테고리 및 태그를 작성해주는 기능부터 작성한지 시간이 많이 지나서 묻혀 버린 글의 경우는 제한된 횟수동안 '끌올' 이라는 기능을 통해서 다시 글이 상단에 노출될 수 있도록 도와주는 기능도 존재한다. 또한, 전문가들과 협업해서 법률, 의학등 전문적인 정보를 제공하기도 한다.

지난 질문글을 다시 끌어올려주는 기능은 있지만, 애초에 질문글을 작성할 때 예상되는 답변시간을 알려주는 서비스는 존재하지 않는다. 질문에 따라서는 답변시간이 중요하지 않은 사소한 질문들도 있지만, 갑자기 눈이 안보이게 되거나 신체의 일부에 감각이 없는 등 건강에 관련돼서 중대하고 긴급한 질문도 있을 수 있다. 119 등을 이용해서 도움을 받을 수도 있지만 사람의 심리상 큰병이나 큰일이 아니길 빌면서 지식인 등에 질문글을 올리는 경우 또한 존재한다.

이러한 문제점을 해결하기 위해서 지식인 질문글로부터 답변이 달리는 예상 시간을 예측하는 방법을 개발하기로 하였다.

답변 시간을 예측할 수 있다면 긴박함을 요하는 질문의 경우 답변이 짧게 달리도록 질문글을 수정하거나 답변 예상시간이 길 경우 우선적으로 병원을 찾아가는 등 다양한 부가적인 효과가 있을 것으로 예상한다.

2. 개요

질문과 답변시간에 대한 모델을 세우기 앞서서 질문과 답변시간의 관계에 대한 가설을 세워 보았다.

1. 유사한 카테고리는 답변시간이 유사할 것이다.
2. 얻을 수 있는 내공이 많을수록 답변시간이 짧을 것이다.
3. 질문의 난이도가 낮을수록 답변시간이 짧을 것이다.
4. 사람들이 많이 활동하는 시간대일수록 답변시간이 짧을 것이다.

5. 답변이 자주 달리는 시간대일수록 답변시간이 짧을 것이다.

위의 가설을 검증하기 위해서 지식인으로부터 데이터를 수집하고 수집의 데이터로부터 모델을 이용해서 가설의 검증을 하는 것이 본 프로젝트의 주요 목표이다.

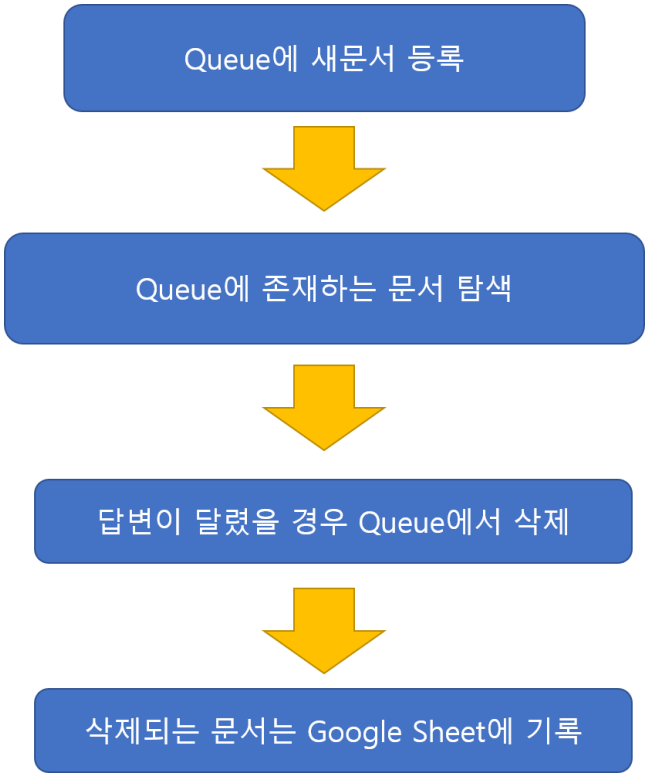
3. 데이터 수집

지식인은 자체적으로 검색서비스를 제공하고 있지만 질문글이나, 질문글의 통계 정보를 공개적으로 제공하지는 않는다. 따라서 우리는 직접 크롤러를 개발해서 지식인 질문글 정보를 수집하였다.

크롤러는 python 의 beutifulsoup4.0 을 이용하여 개발했고,

작동환경은 우분투 18.04 가 설치된 라즈베리파이 서버이다.

3.1 데이터 크롤러



[그림 1] 크롤러 작동방식

크롤러는 그림과 같은 방식으로 작동하며 매 30 분마다 자동으로 실행돼서 데이터를 수집한다.

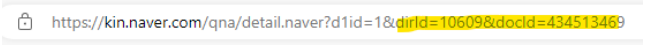
30 분 간격으로 실행시킨 이유는 지식인 자체적으로 1 시간 이후의 데이터는 분정보를 생략해서 분단위 데이터를 얻을 수 없기 때문이다.

A. Queue 에 새문서 등록

https://kin.naver.com/qna/list.naver 사이트로부터 Queue 에 존재하지 않는 문서를 찾아서 queue 에 등록시킨다.

B. Queue 에 존재하는 문서 탐색

지식인의 문서는 2 개의 ID 값으로 정해지는 docID 값과 dirID 값이 해당 값은 문서의 URL 에서 구할 수 있으면 이 두 값들을 key 값으로 이용해서 문서를 탐색한다.



C. 답변이 달렸을 경우 Queue 에서 삭제

답변이 달린 경우 Queue 에서 대당 문서의 dirID, docID 값을 삭제하고, 문서의 데이터를 저장한다.

D. 삭제되는 문서를 Google sheet 에 기록

[그림 2] gspread 작동방식

저장된 문서 데이터를 자동으로 기록하기 위해서 구글에서 제공하는 gspread 라는 구글 스프레드시트 API 를 사용하였다.

gspread 의 경우 한시간동안 호출할 수 있는 횟수 제한이 있었다. 지식인의 질문량이 워낙 많아서 이 제한 횟수를 넘는 경우가 종종 있었는데, 그런 경우 제한 횟수까지 도달한 순간 이미 충분한 양의 데이터가 수집되었기 때문에 횟수 이상의 데이터들은 수집하지 않는 방식으로 하였다.

Data Feature

수집한 데이터의 features 는 다음과 같다.

[표 1] 데이터별 자료형

수집 데이터	타입
Upload Time	Date (시간)
Answer Time	Date (시간)
Title	String(자연어)
Content	String(자연어)
Answer Count	Int
Views	Int
Points	Int
TTA (Time To Answer)	Int (Upload Time – Answer Time)

[표 2] 데이터별 수집한 내용

수집 데이터	내용
Upload Time	질문글이 작성된 시간
Answer Time	답변이 작성된 시간
Title	질문글 제목
Content	질문글 본문
Answer Count	답변 개수
Views	질문글 조회수
Points	질문글 내공
TTA (Time To Answer)	첫번째 답변까지 걸린 시간(분)

전체 데이터 개수는 3 일치 10253 개의 데이터를 수집하였다.

4. Experiment Design

수집한 데이터들을 가설에 맞게 전처리할 필요가 있었기 때문에 전처리 과정을 진행하였다.

4.1 Preprocessing

1. 유사한 카테고리는 답변시간이 유사할 것이다.

A. 카테고리별 one-hot encoding

첫 번째로는 카테고리를 단순히 one-hot encoding 하는 방법이다.

[표 3] one-hot encoding table

1	컴퓨터 통신	**2**	게임
3	엔터테인먼트, 예술	**4**	경제
5	쇼핑	**6**	사회, 정치
7	건강	**8**	생활
9	여행	**10**	스포츠, 레저
11	교육, 학문	**12**	지역 &플레이스

우선 위의 방식대로 인코딩한 one-hot encoding vector 를 추가해서 3 layer DNN 모델에 학습을 시도했다. 결과는 loss 가 발산하여 학습이 제대로 이루어 지지 않는 것을 확인할 수 있었다.

```
Epoch : 0, loss : 1.1556, valid : 1.5592
Epoch : 1, loss : 1.8150, valid : 2.8101
Epoch : 2, loss : 1.9131, valid : 2.1118
Epoch : 3, loss : 1.0163, valid : 3.0368
Epoch : 4, loss : 0.9657, valid : 3.1917
Epoch : 5, loss : 2.2565, valid : 2.3038
Epoch : 6, loss : 2.5882, valid : 3.2965
Epoch : 7, loss : 3.3264, valid : 2.0165
Epoch : 8, loss : 0.9448, valid : 3.0014
Epoch : 9, loss : 2.0317, valid : 2.7349
```

[그림 3] one-hot encoding 결과

문제에 대한 원인으로는 크게 2 가지 이유를 추측했다.

첫번째는 대분류가 의미를 가지지 못할 정도로 너무 광범위하다는 것이다. 대분류 12 가지 밑에 수백가지의 소분류 들이 있기 때문에 대분류로 나누는 것은 적합하지 못하다는 생각했다.

두번째는 One-hot encoding 자체적으로 가지는 단점인 같은 카테고리는 표현할 수 있지만 다른 카테고리 별 유사도를 표현할 수는 없고 여기서 유의미한 정보를 없어졌기 때문에 이런 문제가 생겼다고 생각했다.

B. Ko-Electra 를 이용한 tokenization

위의 실험을 통해서 대분류 카테고리의 한계를 찾아냈고 이를 극복하기 위해서 제목과 본문 내용을 사용하는 방식으로 제목과 본문 내용을 tokenizer 로 문장을 토큰화 하는 방법을 선택하였다.

해당 방법을 사용한 이유는 지식인의 카테고리 시스템이 수동으로 정하지 않은 경우 사용자의 질문글의 제목과 본문내용을 바탕으로 자동으로 카테고리를 생성해주는 방식이기 때문에 제목과 본문 내용을 벡터화 시킬 수만 있으면 모델이 이로부터 유사도 혹은 답변시간과의 관계성을 학습할 것이라고 생각했기 때문이다.

따라서, 제목과 본문을 인코딩하여 모델의 feature 로 사용하기로 하였다. 사전 학습된 Ko-Electra tokenizer 를 사용하여 제목과 본문을 이어붙인 문장들을 토큰화하였다.

다른 tokenizer 대신에 Ko-Electra 를 사용한 이유는 Ko-Electra 가 기존에 BERT 모델보다 우수하다고 판단했기 때문이다. Kevin Clark, 2020 [1]의 연구에 따르면 Electra 는 기존 BERT 모델의 한계를 지적하며 나온 모델로, BERT 는 MLM(masked language model)을 이용하여 학습을 하는데 이 경우 15%만이 MASK 처리되므로 전체 토큰에 대해 15%만이 loss 가 발생하고 학습 효율성이 떨어진다. 그리고 이를 해결하기 위해 다른 학습 method 인 replaced token detection 를 사용하는 모델이 우리가 사용한 Electra 이다.

이는 BERT 와 달리 모든 input 토큰에 대해 학습이 가능한 장점을 지녔다. Kevin Clark 의 연구에 따르면 Electra-Small 모델이 하나의 GPU 로 사전 학습하는데 4 일밖에 걸리지 않았다고 언급하였는데 그럼에도 GPT 와 BERT-Small 보다 높은 GLUE score 기록하였고 BERT 에 비해 적은 컴퓨팅 자원을 사용하는 것이 google colab 외의 컴퓨팅 자원을 활용할 수가 없는 우리 팀의 상황에 매우 걸맞다고 생각하였다.

이렇게 Electra 의 다양한 장점으로부터 우리는 한국어 버전인 Ko-Electra 을 사용해서 질문글의 제목과 본문 내용을 tokenization 하기로 하였다.

데이터의 통계적 정보로 제목과 본문을 전부 합쳤을 때 가장 짧은 경우의 길이는 7 이었고 가장 긴 경우의 길이는 7000 이상이었다. 이를 바탕으로 tokenizer 를 설정할 때 생성되는 토큰의 길이가 Ko-Electra 의 최대 길이인 512 개가 되도록 설정하였다. 제목과 본문이 토큰화된 벡터인 512 차원을 그대로 사용하기에는 다른 feature 들의 정보가 묻힐 수도 있고 무의미한 features 가 많을 수 있기 때문에 input 으로 넣기 전에 sparse representation 을 dense representation 으로 바꾸는 작업을 할 필요가 있었다.

송은영,2019 [2]의 연구에 따르면 PCA 를 사용하여 차원축소를 진행할 경우 word embedding 시 사용되면 모델 학습시에 더 낮은 차원으로 성능을 유지하거나 향상하는데 영향을 끼친다. 따라서 우리는 PCA 를 사용해서 embedding vector 의 차원 축소를 진행하였다. 축소하는 차원의 값은 1 과 20 두가지 경우를 시도하였고, 보다 좋은 쪽의 값을 사용하였다.

Text embedding 의 차원을 작은 값만 사용한 이유는 전처리를 통해서 구한 다른 feature 들의 차원이 전부 작아서 text embedding 이 큰 차원 값을 가질 경우 모델이 다른 features 를 제대로 학습하지 못할 우려가 있어서 작은 차원 값을 사용하였다.

2. 얻을 수 있는 내공이 많을수록 답변시간이 짧을 것이다.

이 경우 따로 전처리는 거치지 않고 정수 값을 그대로 사용하되 MLP 에 넣기 전에 MinMax scaling 을 거쳤다.

3. 질문의 난이도가 낮을수록 답변시간이 짧을 것이다.

질문의 난이도는 질문의 학문적인 난이도와 질문의 이해의 난이도 이렇게 2 가지로 나누어서 접근하였다.

질문의 학문적인 난이도의 경우에는 text embedding

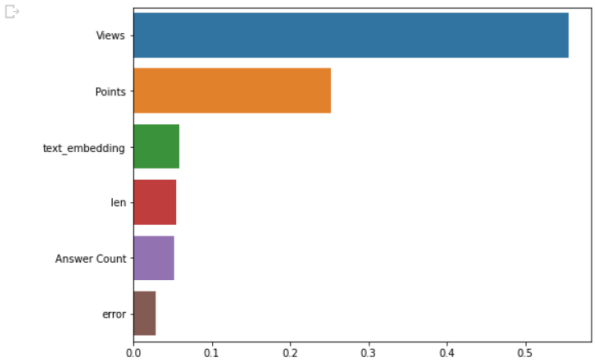
과정에서 모델이 학습하게 될 것이라고 가정했다. 추가적으로 질문의 이해 난이도에 관해서 실험을 설계하였다. 질문의 이해 난이도가 높다는 것은 “질문이 정확히 무엇을 묻는 것인지 알기 어렵다” 를 의미한다.

이를 판단하는 feature 로 제목과 본문의 맞춤법 오류와 제목과 본문의 전체 길이를 feature 로 설계하였다. 맞춤법

오류가 많을수록 사람이 이해하기 힘들고, 길이가 길수록 한눈에 질문의 핵심을 알기 어렵다는 가정이었다.

맞춤법이 얼만큼 틀렸는지를 계산하기 위해서 네이버 맞춤법 검사기 기반의 한글 맞춤법 검사 라이브러리인 “py-hanspell”을 사용하여 text 의 오류 정도를 추출하였다.

py-hanspell 은 맞춤법이 틀린 부분의 개수를 출력해준다. 만약 1 군데가 틀렸으면 1 이 3 군데가 틀렸으면 3 이 출력되는 방식이다.



[그림 4] Random Forest Regressor 맞춤법 feature 의 중요도

그렇게 설계된 features 로 Random Forest Regressor 를 학습했을 때의 결과이다.

위의 그림에서 Points 는 내공, text_embedding 은 Ko-Electra 로 embedding 한 word vectors, len 은 제목과 본문을 포함한 길이이고 error 가 맞춤법 오류를 의미한다.

그래프의 크기는 각각의 feature 가 얼만큼 유의미한 영향을 주었는지를 의미한다.

위의 실험 결과로부터 맞춤법의 오류는 거의 아무런 영향이 없고 오히려 noise 라고 생각되어서 제목과 본문의 길이인 len 만 남겨두고 error 의 경우 feature 에서 제외하였다

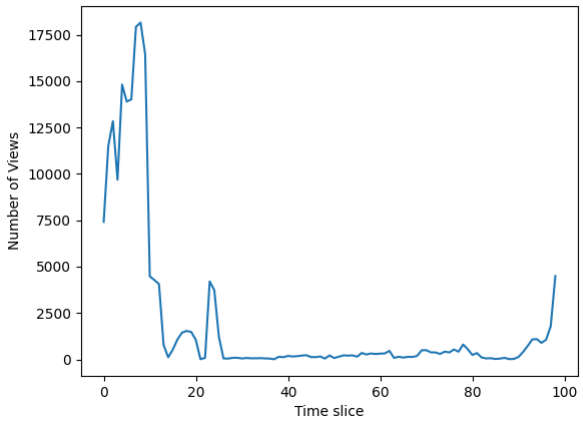
4. 사람들이 많이 활동하는 시간대일수록 답변시간이 짧을 것이다

사람들의 활동량을 정량적으로 분석하기 위해서 수집한 데이터로부터 2 가지를 가정하였다.

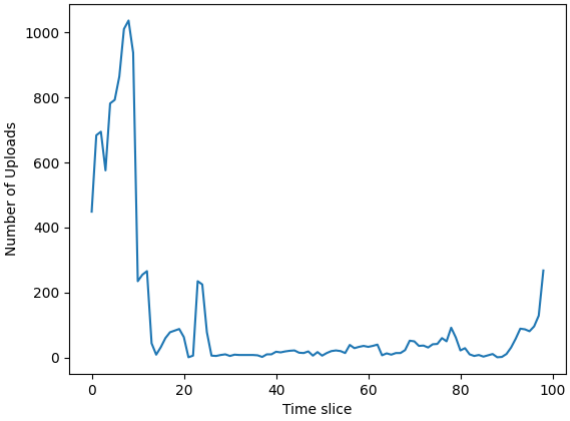
첫번째는 “사람들이 많이 활동할수록 단위 시간당 작성된 질문글의 수가 많을 것이다.” 이다.

두번째는 “사람들이 많이 활동할수록 단위 시간당 조회수의 총합이 더 높을 것이다.” 이다.

위의 2 가지 가정을 통해서 24 시간을 5 분 단위로 쪼개고 upload time 과 views 값을 이용해서 시간별로 값을 비교하였다.



[그림 5] 단위 시간당 조회수의 총합

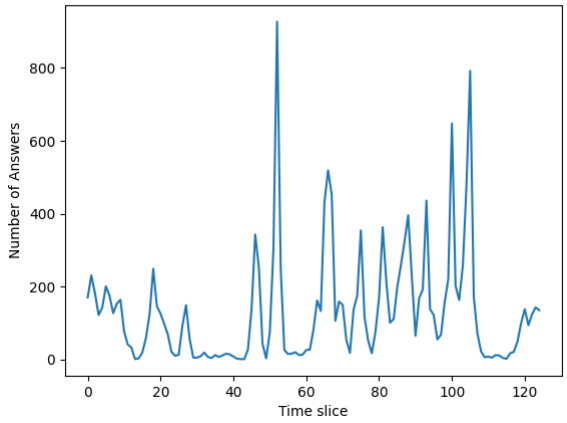


[그림 6] 단위 시간당 질문글의 수

두 변수가 동일한 형태를 하고 있는 것으로 보아 두 변수가 사람들의 활동량에 관해서 서로 같은 영향력을 미친다고 생각하였다

5.답변이 자주 달리는 시간대일수록 답변시간이 짧을 것이다.

사람들의 활동량을 분석하는 방식과 동일하게 24 시간을 5 분단위로 쪼개서 단위 시간당 작성되는 답변의 개수를 계산하였다.



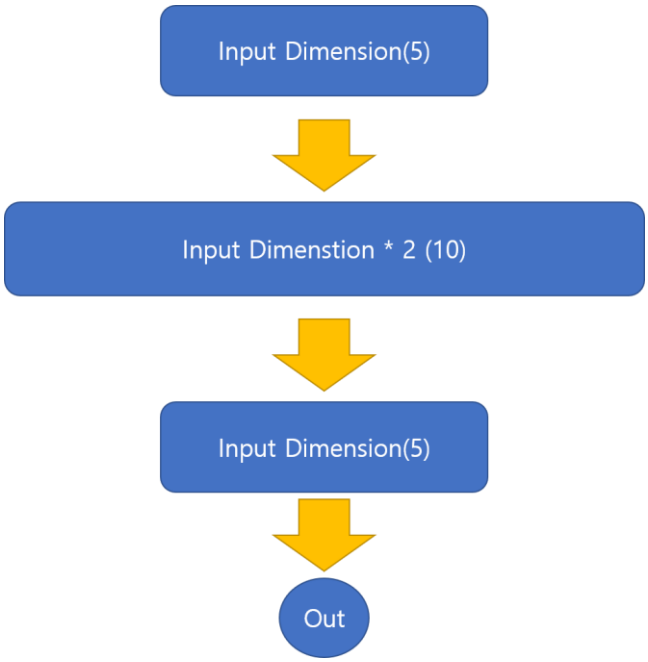
[그림 7] 단위시간당 답변의 수

사람들의 활동량과는 그래프의 형태가 다르지만, 특정시간대에 더 많은 답변이 달리고 특정 시간대는 답변의 개수가 줄어주는 것을 볼 수 있다.

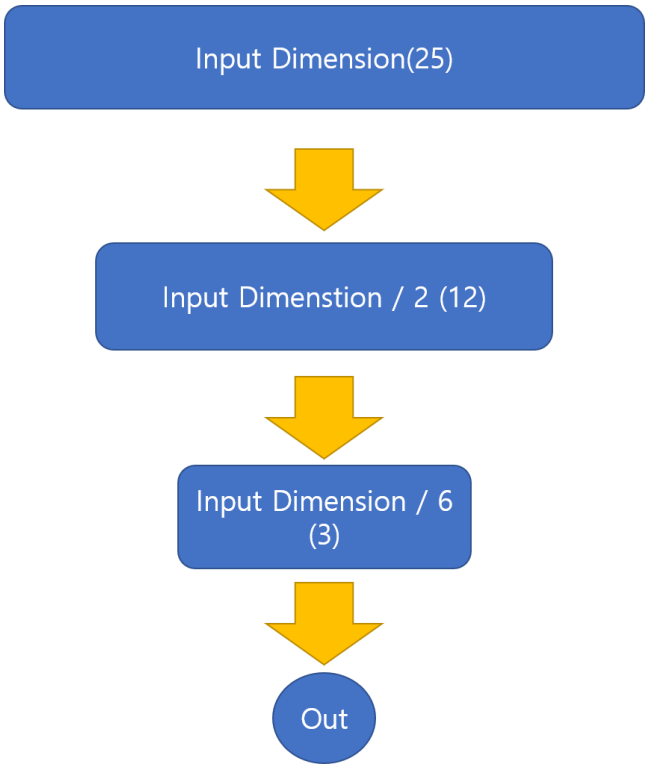
4.2 Model 설계

모델의 경우 sklearn 에서 기본적으로 제공하는 random forest regressor 를 제안하고 baseline 으로는 팀 자체적으로 설계한 MLP 와 linear regressor 를 사용하였다.

MLP 의 구조는 다음과 같다.



[그림 8] text embedding 을 사용하지 않는 MLP



[그림 9] text embedding 을 사용하는 MLP

5. Evaluation

Regression model 에서는 classification 과는 다르게 맞다 틀리다가 아니라 "얼만큼 다르냐"를 중점으로 점수를 매겨야 한다. 그렇기 때문에 loss 를 기반으로 하는 평가 방법을 사용하는데, 이 경우 상대적인 loss 를 비교할 기준이 필요하다.

본 프로젝트에서는 sklearn 의 regressor model 들이 사용하는 평균과 비교하는 score 방식을 사용하였다.

[수식 1] Score 수식

$$Score = 1 - \frac{\sum(x_{real} - x_{pred})^2}{\sum(x_{real} - x_{mean})^2}$$

예측 값이 평균과 유사할수록 의미 없는 regressor 이기 때문에 score 은 0 이 되고 실제 값과 유사할수록 1 에 가까워진다.

6. Results

[표 4] 실험 결과

Score(test)	RFR	MLP	Linear
Dataset without Embedding	0.9182	0.3492	0.4514
Dataset with Embedding (pca(n_..) = 1)	0.3362	0.1908	
Dataset with Embedding (pca(n_..) = 20)	0.8807	0.2119	

가장 좋은 결과 값은 text embedding 없이 Random Forest Regressor 로 학습했을 경우이다.

예상외의 결과로 Random Forest Regressor, MLP 모두 text embedding 을 했을 경우 오히려 결과가 부정적으로 바뀌었다.

그리고 PCA 차원의 경우 20 차원이 1 차원으로 축소한 경우보다 결과가 좋았다.

7. Analysis

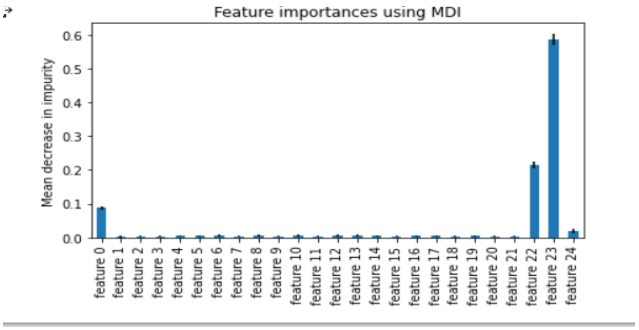
데이터의 차원이 낮고 낮은 차원에 비해서 형태가 복잡해서 MLP 나 Linear regressor 보다 Random Forest Regressor 의 성능이 더 높게 나왔다고 생각한다.

Text embedding 을 적용했을 경우 결과가 부정적으로 바뀌는 이유는 토큰화된 내용을 모델이 제대로 학습하지 못하였거나, 수집한 데이터의 양이 Ko-Electra 의 정확도를 높일 정도로 많은 양이 아니었기 때문이라고 생각한다.

PCA 차원의 경우 1 차원으로 축소하는 경우는 너무 많은 정보를 소실해서 의미가 없는 noise 처럼 작용하기 때문에 결과가 더 안좋다고 생각한다.

다음은 Random Forest Feature Importance (MDI)를 사용하여 각 feature 별로 중요도를 시각화 하였다. Feature 0 는 내공, feature 1~20 은 20 차원의 word embedding, feature 21 은 제목과 본문의 길이, feature 22 는 질문이 올라온 시간, feature 23 은 답변이 달린 시간, feature 24 는 조회수를 의미한다.

시각화의 결과로 보았을 땐 기대와 달리 제목과 본문의 embedding 이 영향이 없었으며 내공, 질문이 올라간 시간, 답변이 달린 시간이 중요한 feature 인 것으로 나타난다.



[그림 10] Random Forest Feature Importance

[Reference]

[1] Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning, 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators

[2] 송은영, 최회련, 이홍철. 2019. Word Embedding 에 PCA 를 적용한 개체명 인식 모델을 위한 효율적인 학습방법 연구

[3] https://docs.google.com/spreadsheets/d/1SxvvmRzz8IMVg2nOkCsbR7wF5SpXY6jFFAyDsLkIeOo/edit#gid=0