네이버 영화평을 이용한 긍정/부정 단어사전 만들기

학번 : K2020509 이름: 배기석

1. 프로젝트 목표

❖ 수업시간에 배운 것을 의미 있는 실습을 통해 복습하자
❖ 자연어처리와 ML 과의 경계를 확인하자.

실습순서

1. 네이버 영화평 200,000개 에서 출현단어의 빈도수 확인

2. 학습을 시킨 후 긍정과 부정에 대한 문장입력으로 분류성능 확인

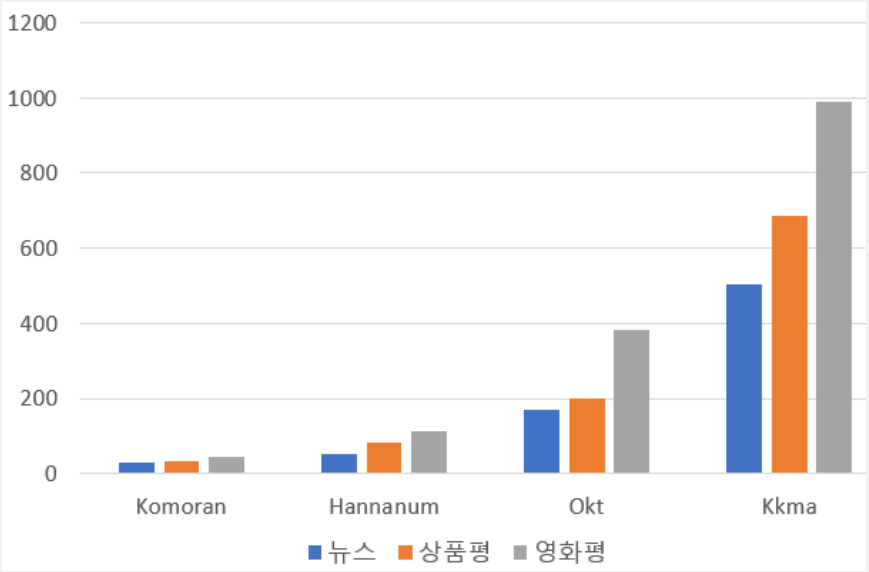
3. 네이버 영화평 200,000개에서 조사와 문장부호를 제거한 상태에서 출현단어의 빈도수 확인

4. 학습을 시킨 후 긍정과 부정에 대한 문장입력으로 분류성능 확인

5. 100% 분류를 통한 긍정 및 부정 단어사전 작성

2. 실습 – 네이버영화평에서 단어의 출현 빈도 확인

- ❖ nltk.Text.vocab().most_common를 이용한 출현 빈도별 토큰분석
- ❖ 맞춤법 수정기능 필요
- ❖ 품사 태깅 필요

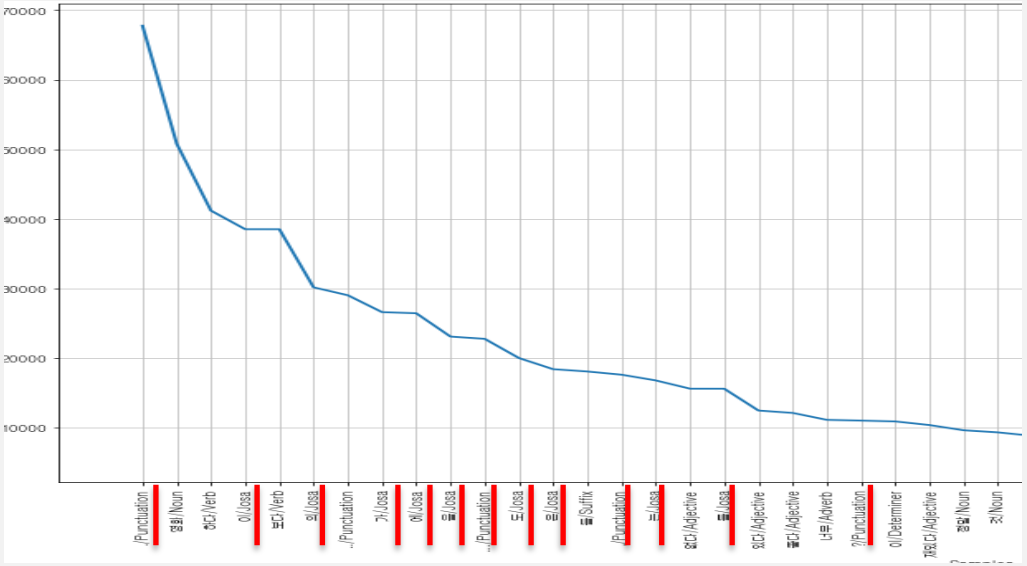


도메인에 따라 문장 유형에 따라 조금씩 다르지만 아래와 같은 결론을 내립니다.

빠른 속도와 보통의 정확도를 원한다면 "Komoran" 또는 "Hannanum"
(이번 분석 중 Komoran의 놀라운 발전에 감짝 놀랐습니다.)
속도는 느리더라도 정확하고 상세한 품사 정보를 원한다면 "Kkma"
어느 정도의 띄어쓰기 되어 있는 "인터넷" 영화평/상품명을 처리할 땐 "Okt"
(만약 띄어쓰기가 없다면 느린 처리속도는 감수해야함)

Okt.pos(phrase, norm=True, stem=True)

- Norm은 정규화 옵션
- Stem은 근어 옵션



https://m.blog.naver.com/PostView.nhn?blogId=wideeyed&logNo=221337575742&proxyReferer=https%2F%2Fwww.google.com%2F

2. 실습 – 학습의 방법 및 분류성능

❖ Keras를 이용한 학습 (Dense와 유닛 그리고 활성화 함수의 이해)

```
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
from tensorflow.keras import losses
from tensorflow.keras import metrics

model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss=losses.binary_crossentropy,
              metrics=[metrics.binary_accuracy])

model.fit(x_train, y_train, epochs=10, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
predict_pos_neg("올해 최고의 영화! 세 번 넘게 봐도 질리지가 않네요.")
predict_pos_neg("배경 음악이 영화의 분위기랑 너무 안 맞았습니다. 몰입에 방해가 됩니다.")
predict_pos_neg("주연 배우가 신인인데 연기를 진짜 잘 하네요. 몰입감 ㅎㄷㄷ")
predict_pos_neg("믿고 보는 감독이지만 이번에는 아니네요")
predict_pos_neg("주연배우 때문에 봤어요")
```

```
[올해 최고의 영화! 세 번 넘게 봐도 질리지가 않네요.]는 99.22% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^
[배경 음악이 영화의 분위기랑 너무 안 맞았습니다. 몰입에 방해가 됩니다.]는 91.84% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;
[주연 배우가 신인인데 연기를 진짜 잘 하네요. 몰입감 ㅎㄷㄷ]는 96.14% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^
[믿고 보는 감독이지만 이번에는 아니네요]는 68.46% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;
[주연배우 때문에 봤어요]는 69.67% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;
```

Dense layer는 입력과 출력을 모두 연결하는 기능을 합니다. 예를 들어 입력 뉴런이 8개이고 출력이 8개 있다면 총 연결선은 64개 (8*8=64) 입니다.
이렇게 입력뉴런과 출력뉴런을 모두 연결한다고 해서 전결합층이라고 불리고,
케라스에서는 Dense 라는 클래스로 구현되어 있습니다.

https://tykimos.github.io/2017/01/27/MLP_Layer_Talk/

Sigmoid
입력값을 0과 1 사이의 값으로 변환하여 출력 (이진분류에 적합)하며, 레이어가 깊어질수록 적용하기 어려운 문제가 있다.

ReLU (rectified liear unit)
0이하의 값은 다음 레이어에 전달하지 않고, 0이상의 값은 그대로 출력하며, 주로 CNN을 학습 시킬 때 많이 사용됩니다.

https://yeomko.tistory.com/39

2. 실습 – 학습의 방법 및 분류성능

❖ Keras를 이용한 학습 (Dense와 유닛 그리고 활성화 함수의 이해)

```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss=losses.binary_crossentropy,
              metrics=[metrics.binary_accuracy])

model.fit(x_train, y_train, epochs=10, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/10
293/293 [==============================] - 7s 23ms/step - loss: 0.3856 - binary_accuracy: 0.8352
Epoch 2/10
293/293 [==============================] - 6s 21ms/step - loss: 0.3158 - binary_accuracy: 0.8644
Epoch 3/10
293/293 [==============================] - 6s 21ms/step - loss: 0.2924 - binary_accuracy: 0.8778
Epoch 4/10
293/293 [==============================] - 6s 20ms/step - loss: 0.2724 - binary_accuracy: 0.8885
Epoch 5/10
293/293 [==============================] - 6s 20ms/step - loss: 0.2535 - binary_accuracy: 0.8975
Epoch 6/10
293/293 [==============================] - 6s 20ms/step - loss: 0.2338 - binary_accuracy: 0.9073
Epoch 7/10
293/293 [==============================] - 6s 20ms/step - loss: 0.2144 - binary_accuracy: 0.9170
Epoch 8/10
293/293 [==============================] - 6s 20ms/step - loss: 0.1959 - binary_accuracy: 0.9252
Epoch 9/10
293/293 [==============================] - 6s 20ms/step - loss: 0.1793 - binary_accuracy: 0.9321
Epoch 10/10
293/293 [==============================] - 6s 20ms/step - loss: 0.1647 - binary_accuracy: 0.9380
1563/1563 [==============================] - 4s 2ms/step - loss: 0.4180 - binary_accuracy: 0.8520
```

```
results
```

```
[0.4179568290710449, 0.8520200252532959]
```

```
def predict_pos_neg(review):
    token = tokenize(review)
    tf = term_frequency(token)
    data = np.expand_dims(np.asarray(tf).astype('float32'), axis=0)
    score = float(model.predict(data))
    if(score > 0.5):
        print("[{}]는 {:.2f}% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^\n".format(review, score * 100))
    else:
        print("[{}]는 {:.2f}% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;\n".format(review, (1 - score) * 100))
```

```
predict_pos_neg("올해 최고의 영화! 세 번 넘게 봐도 질리지가 않네요.")
predict_pos_neg("배경 음악이 영화의 분위기랑 너무 안 맞았습니다. 몰입에 방해가 됩니다.")
predict_pos_neg("주연 배우가 신인인데 연기를 진짜 잘 하네요. 몰입감 ㅎㄷㄷ")
predict_pos_neg("밀고 보는 감독이지만 이번에는 아니네요")
predict_pos_neg("주연배우 때문에 봤어요")
```

```
[올해 최고의 영화! 세 번 넘게 봐도 질리지가 않네요.]는 98.86% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^

[배경 음악이 영화의 분위기랑 너무 안 맞았습니다. 몰입에 방해가 됩니다.]는 73.25% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;

[주연 배우가 신인인데 연기를 진짜 잘 하네요. 몰입감 ㅎㄷㄷ]는 99.71% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^

[밀고 보는 감독이지만 이번에는 아니네요]는 76.21% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^

[주연배우 때문에 봤어요]는 94.73% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;
```

시간이 오래 걸리지만 Input_shape가 추측확률을 결정짓는 요소가 되기 때문에 10,000을 유지

확률 0.85 로써 조심스러운 수치이긴 하지만 단어사전 구분 용 85% 에서 다시 80%를 선택하는 작업

비교용 데이터

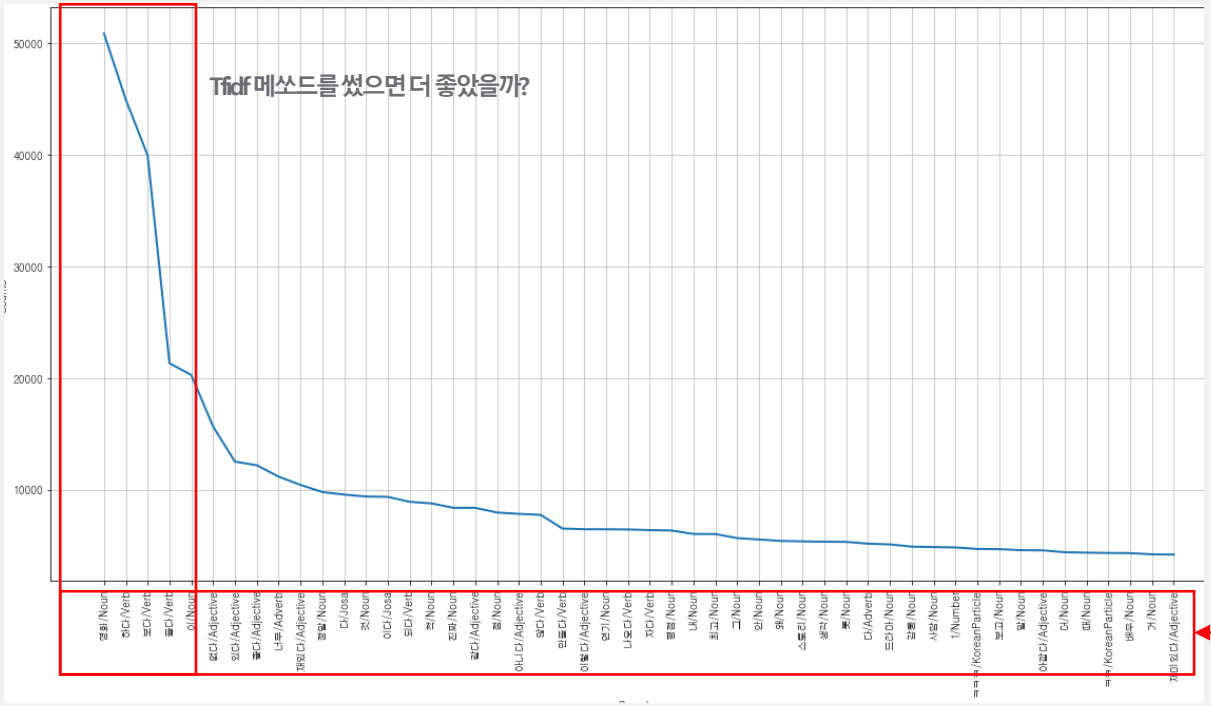
2. 실습 – 문장부호 제거방법

❖ Okt를 이용한 품사확인

❖ 조사와 문장부호를 제거한 학습용 data 생성

```
okt = Okt()
print (okt.pos(u'동해물과 백두산이 마르고 닳도록 하느님이 보우하사.'))
```

```
[('동', 'Modifier'), ('해물', 'Noun'), ('과', 'Josa'), ('백두산', 'Noun'), ('이', 'Josa'), ('마르고', 'Noun'), ('닳도록', 'Verb'), ('하느님', 'Noun'), ('이', 'Josa'),
('보우', 'Noun'), ('하사', 'Noun'), ('.', 'Punctuation')]
```




```
# 명사, 동사, 형용사, 부사 등 의미있는 단어를 남긴다.
okt = Okt()
def str_filter(review):
    str_result = ""
    for t in okt.pos(review, norm=True, stem=True):
        if (t[1] =="Josa" or t[1] == "Punctuation"):
            str_result = str_result
        else:
            str_result = str_result + " " + t[0]
    return str_result
```

2. 실습 – 동일한 학습과 분류성능

❖ Keras를 이용한 학습 (Dense와 유닛 그리고 활성화 함수의 이해)

```
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
from tensorflow.keras import losses
from tensorflow.keras import metrics

model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss=losses.binary_crossentropy,
              metrics=[metrics.binary_accuracy])

model.fit(x_train, y_train, epochs=10, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
predict_pos_neg("올해 최고의 영화! 세 번 넘게 봐도 질리지가 않네요.")
predict_pos_neg("배경 음악이 영화의 분위기랑 너무 안 맞았습니다. 몰입에 방해가 됩니다.")
predict_pos_neg("주연 배우가 신인인데 연기를 진짜 잘 하네요. 몰입감 ㅎㄷㄷ")
predict_pos_neg("믿고 보는 감독이지만 이번에는 아니네요")
predict_pos_neg("주연배우 때문에 봤어요")
```

```
[올해 최고의 영화! 세 번 넘게 봐도 질리지가 않네요.]는 99.53% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^
[배경 음악이 영화의 분위기랑 너무 안 맞았습니다. 몰입에 방해가 됩니다.]는 92.94% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;
[주연 배우가 신인인데 연기를 진짜 잘 하네요. 몰입감 ㅎㄷㄷ]는 99.48% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^
[믿고 보는 감독이지만 이번에는 아니네요]는 58.15% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;
[주연배우 때문에 봤어요]는 74.49% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;
```

조사를 제거하면 조금 더 긍정에 가까운 확률이 나옴

2. 실습 – 동일한 학습과 분류성능

❖ 조사가 없을 경우의 분류성능

```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss=losses.binary_crossentropy,
              metrics=[metrics.binary_accuracy])

model.fit(x_train, y_train, epochs=10, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/10
293/293 [==============================] - 7s 22ms/step - loss: 0.3972 - binary_accuracy: 0.8266
Epoch 2/10
293/293 [==============================] - 7s 23ms/step - loss: 0.3272 - binary_accuracy: 0.8603
Epoch 3/10
293/293 [==============================] - 7s 22ms/step - loss: 0.3055 - binary_accuracy: 0.8717
Epoch 4/10
293/293 [==============================] - 7s 22ms/step - loss: 0.2893 - binary_accuracy: 0.8811
Epoch 5/10
293/293 [==============================] - 7s 22ms/step - loss: 0.2726 - binary_accuracy: 0.8890
Epoch 6/10
293/293 [==============================] - 7s 22ms/step - loss: 0.2557 - binary_accuracy: 0.8971
Epoch 7/10
293/293 [==============================] - 7s 23ms/step - loss: 0.2391 - binary_accuracy: 0.9046
Epoch 8/10
293/293 [==============================] - 7s 22ms/step - loss: 0.2225 - binary_accuracy: 0.9117
Epoch 9/10
293/293 [==============================] - 7s 22ms/step - loss: 0.2068 - binary_accuracy: 0.9193
Epoch 10/10
293/293 [==============================] - 7s 22ms/step - loss: 0.1922 - binary_accuracy: 0.9256
1563/1563 [==============================] - 5s 3ms/step - loss: 0.4393 - binary_accuracy: 0.8380
```

```
results
```

```
[0.4392968714237213, 0.8380399942398071]
```

확률 0.84 로써 반복되는 조사가 없어지니 분류의 정확률이 떨어짐

```
def predict_pos_neg(review):
    token = tokenize(review)
    tf = term_frequency(token)
    data = np.expand_dims(np.asarray(tf).astype('float32'), axis=0)
    score = float(model.predict(data))
    if(score > 0.5):
        print("[{}]는 {:.2f}% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^\n".format(review, score * 100))
    else:
        print("[{}]는 {:.2f}% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;\n".format(review, (1 - score) * 100))
```

```
predict_pos_neg("올해 최고의 영화! 세 번 넘게 봐도 질리지가 않네요.")
predict_pos_neg("배경 음악이 영화의 분위기랑 너무 안 맞았습니다. 몰입에 방해가 됩니다.")
predict_pos_neg("주연 배우가 신인인데 연기를 진짜 잘 하네요. 몰입감 ㅎㄷㄷ")
predict_pos_neg("믿고 보는 감독이지만 이번에는 아니네요")
predict_pos_neg("주연배우 때문에 봤어요")
```

```
[올해 최고의 영화! 세 번 넘게 봐도 질리지가 않네요.]는 99.53% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^

[배경 음악이 영화의 분위기랑 너무 안 맞았습니다. 몰입에 방해가 됩니다.]는 92.94% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;

[주연 배우가 신인인데 연기를 진짜 잘 하네요. 몰입감 ㅎㄷㄷ]는 99.48% 확률로 긍정 리뷰이지 않을까 추측해봅니다.^^

[믿고 보는 감독이지만 이번에는 아니네요]는 58.15% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;

[주연배우 때문에 봤어요]는 74.49% 확률로 부정 리뷰이지 않을까 추측해봅니다.^^;
```

긍정적일 확률이 근소한 차이로 올라감

3. 100% 긍정, 부정 단어 추출 (단어사전 작성)

- ❖ nltk.Text.vocab().most_common를 이용한 출현 빈도별 토큰분석
- ❖ 명사, 동사, 형용사, 부사를 중심으로 결과비교
- ❖ 80% 이상 긍정단어와 80%이상 부정단어 추출하여 사전 파일 작성

80% 이상 긍정적인 단어사전
(조사가 포함된 파일로 제작)

명사
동사
형용사
부사
조사
문장부호 등

80% 이상 부정적인 단어 사전
(조사가 제거된 파일로 제작)

3. 100% 긍정, 부정 단어 추출 (단어사전 작성)

- ❖ nltk.Text.vocab().most_common를 이용한 출현 빈도별 토큰분석
- ❖ 명사, 동사, 형용사, 부사를 중심으로 결과비교
- ❖ 80% 이상 긍정단어와 80%이상 부정단어 추출하여 사전 파일 작성

```python
def make_dictionary(word):
    token = tokenize(word)
    tf = term_frequency(token)
    data = np.expand_dims(np.asarray(tf).astype('float32'), axis=0)
    score = float(model.predict(data))
    if(score > 0.8):
        score = score * 100
        spoint = str(score)
        strReturn = token[0] + "/" + spoint
    else:
        strReturn = "fault"

    return strReturn

# print (len(set(text.tokens)))
checkwords = set(text.tokens)
arrcheckwords = list(checkwords)
# print ((arrcheckwords[0].split("/"))[0])
positive_dictionary = open("./data/positive_dictionary.txt","w",encoding="utf-8")
for t in arrcheckwords:
    result = make_dictionary((t.split("/"))[0])
    if result != "fault":
       positive_dictionary.write(result+"\n")
positive_dictionary.close()
```

명사
동사
형용사
부사
조사
문장부호 등

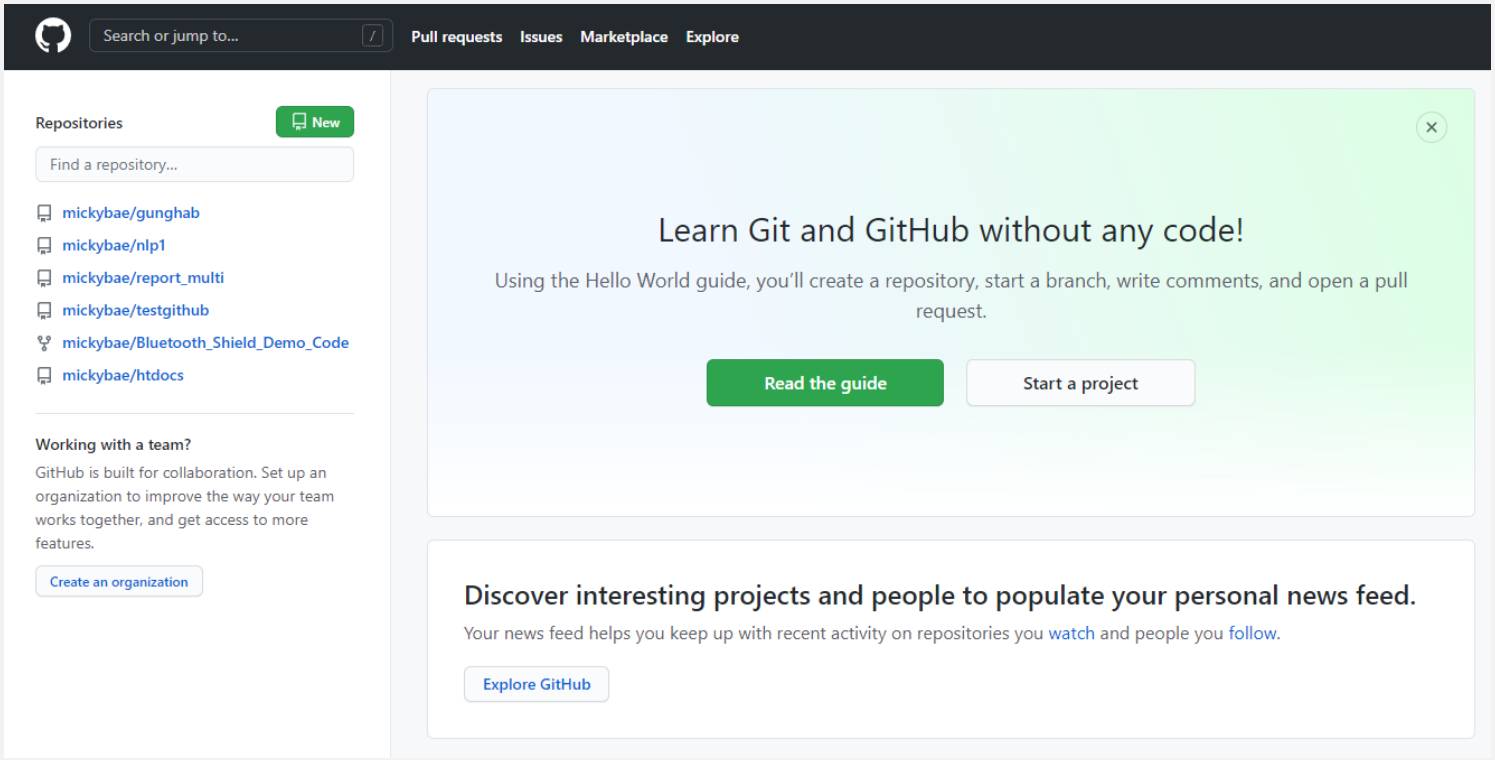
```python
def make_dictionary(word):
    token = tokenize(word)
    tf = term_frequency(token)
    data = np.expand_dims(np.asarray(tf).astype('float32'), axis=0)
    score = float(model.predict(data))
    if(score < 0.2):
        score = (1-score) * 100
        spoint = str(score)
        strReturn = token[0] + "/" + spoint
    else:
        strReturn = "fault"

    return strReturn

# print (len(set(text.tokens)))
checkwords = set(text.tokens)
arrcheckwords = list(checkwords)
# print ((arrcheckwords[0].split("/"))[0])
negative_dictionary = open("./data/negative_dictionary.txt","w",encoding="utf-8")
for t in arrcheckwords:
    result = make_dictionary((t.split("/"))[0])
    if result != "fault":
       negative_dictionary.write(result+"\n")
negative_dictionary.close()
```

positive_dictionary.txt

negative_dictionary.txt

4. 사용소스 및 데이터 공개

❖ Git-hub 사용하기



온라인 공유 게시

- 프로젝트 개요
- 사용소스
- 원본데이터
- 긍정사전
- 부정사전

감사합니다.