GitHub Commit Message Log의 긍정과 부정 의미 분류를 통한 프로젝트 진행 상황 분석

1. 임슬기(19910714)
2. lsg7240@naver.com
3. 임슬기(19910714) ./ 윤영(19880418) / 김민수(19900102)

Github은 세계에서 가장 큰 오픈소스 개발자 커뮤니티의 본거지이며 2015년 기준으로 1,200만 명이 넘는 사람들이 Github에서 3,100만 건 이상의 프로젝트에 기여해왔다.

- Github 로고

Git과 Github은 개발자들이 가장 많이 사용하는 협업 도구 중 하나이다. Git은 개발자들이 개발할 때 소스코드의 버전을 관리해주는 분산 버전관리 시스템(DVCS, Distributed Version Control System) 중 가장 유명하고 Github은 Git의 데이터를 저장하는 서버 역할을 한다.

오픈소스 소프트웨어이며 리눅스를 만든 리누스 토발즈와 주니오 하마노가 개발했다. Git을 이용하면 누가 어떤 코드를 수정했는지 기록하고 추적할 수 있다. 많은 사람들이 함께 소프트웨어를 개발할 때 무척이나 유용하다.

Git 저장소에 커밋 메시지가 뒤죽박죽으로 되어있다고 상상해보자. 당장 그런 커밋 메시지를 보는 것만으로도 머리가 지끈지끈 아파올것이다.

명료하고 좋은 커밋 메시지는 같이 일하는 팀 동료뿐만 아니라 나중에 자신이 볼 때도 많은 도움이 된다. 언제 어떤 기능이 추가되었는지, 혹은 언제 왜 코드가 변경되었는지 쉽게 알아볼 수 있기 때문이다.

연구 주제 수립

개발자라면 누구나 Git에 커밋(Commit)을 할 때 커밋 메시지(Commit message)를 어떻게 작성해야 할지 한번쯤 고민해보았을 것이다. 지금까지 GitHub에는 수천만 건의 커밋 메시지가 작성되었고 지금도 수많은 커밋 메시지가 작성되고 있다.

그렇다면 수많은 개발자들이 작성한 GitHub의 Commit Message Log에서 가장 많이 쓰인 단어와 비속어, 감탄사는 무엇인지 분석해볼 수 있다.

비속어와 감탄사는 사람이 감정을 표현하는 단어이다. 비속어와 함께 쓰인 문장의 단어들은 대체로 부정적 의미를 지니고 있고, 감탄사와 함께 쓰인 단어들은 긍정적 의미를 내포하고 있다.

그렇다면 긍정과 부정의 의미를 가진 단어를 분류해보고, 커밋 메시지 로그에서 어떤 단어가 많이 쓰이는지에 따라 프로젝트 진행 상황을 분석해볼 수 있지 않을까 하는 가설에서 이번 연구 분석을 진행하게 되었다.

GitHub Commit message 데이터셋

분석에는 https://www.kaggle.com/github/github-repos에서 제공하는 'GitHub Repos'의 데이터를 이용하였다.

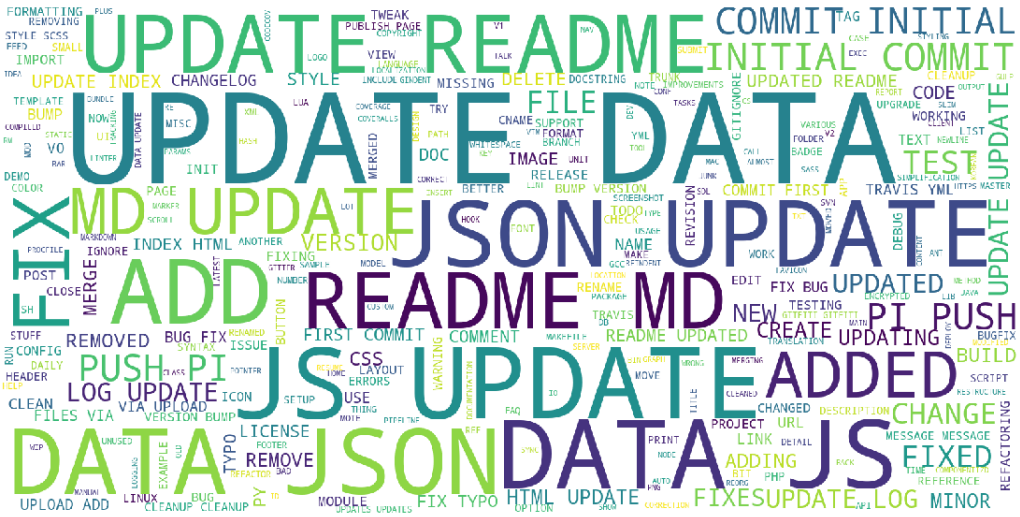
- 데이터 출처 : https://www.kaggle.com/github/github-repos
- 이 데이터세트는 현재까지 출시된 GitHub 활동 중 가장 큰 소스로 구성됨
- 총 145,000 개 이상의 고유 커밋과 280만 개 이상이 오픈소스 GitHub 저장소의 전체 스냅 샷이 포함
- GitHub 레포지토리의 커밋 로그를 분석하여 '커밋 메시지에 쓰인 단어들의 긍정과 부정 의미 분류'을 통해 인사이트를 얻어볼 수 있지 않을까 가설을 세워본다.

Data Source 설명

· commits : commit message, repository name, committer 등의 데이터를 포함한 테이블
· contents : id 와 size, content 등의 데이터를 포함한 테이블
· files : repo_name, ref, path 등의 데이터를 포함한 테이블
· languages : repo_name, language 데이터를 포함한 있는 테이블
· license : repo_name, license 데이터를 포함한 테이블
· sample_commits : commit, author, message 등의 데이터를 포함한 샘플 테이블
· sample_contens : id, content, smaple_repo_name 등의 데이터를 포함한 샘플 테이블
· sample_files : repo_name, ref, path 등의 데이터를 포함한 샘플 테이블
· sample_repos : repo_name, watch_count 등의 데이터를 포함한 샘플 테이블

커밋 메시지에서 가장 빈도 수가 높은 단어는 무엇일까?

개발자들이 커밋 메시지를 작성할 때 가장 많이 사용하는 단어가 무엇인지 분석해보았다. 분석 결과는 워드 클라우드(word cloud) 형태로 나타냈다. 워드 클라우드는 자주 나타나는 단어를 크게 보여줌으로써 직관적으로 텍스트를 파악하는데 도움을 준다. 개발자들이 커밋 메시지를 작성할 때 가장 많이 사용하는 단어들은 UPDATE, DATA, JS, JSON, README 등으로 나타났다.



커밋 메시지에서 가장 많이 쓰인 비속어는 무엇일까?

그렇다면 일반적인 단어들이 아닌 개발자들이 쓰는 커밋 메시지에서 가장 많이 쓰인 비속어는 무엇일지 흥미가 생긴다. 흔히 쓰이는 영어 비속어 10개를 추려서 비속어 리스트를 만들었다.

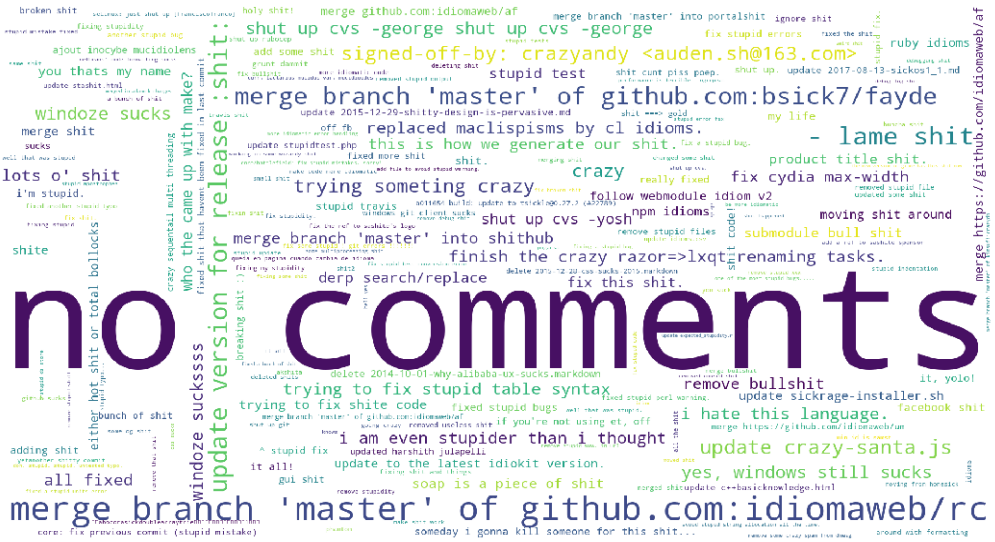
slang = {'sick', 'jerk', 'fuck', 'crazy', 'shut up', 'shit', 'stupid', 'suck', 'damn', 'idiot'}

해당 비속어가 속한 커밋 메시지를 분석해보자. 카운트(hits) 수를 기준으로 상위 30개를 분석하였다.

	message	hits
0	no fucking comments	1238
1	piece of shit	955
2	stupid typo	380
3	fix stupid typo	299
4	stupid	286
5	fix stupid bug	256
6	i'm an idiot	222
7	stupid mistake	214
8	dammit	206
9	stupid fix	204
10	more shit	182
11	fix shit	177
12	stupid bug	176
13	merge branch 'master' of github.com:idiomaweb/rc	174
14	fixed stupid bug	171
15	fix stupid mistake	155
16	fuck it	151
17	fixed stupid typo	144
18	fuck this	134
19	shut up cvs	133
20	fixed shit	132
21	fuck you	114
22	fix stupid error	113
23	fixed stupid mistake	110
24	bullshit	108
25	idiot	106
26	some shit	106
27	fuck this shit	105
28	i'm stupid	101
29	fixed stupid error	99

많은 개발자들이 커밋 메시지를 작성할 때 비속어를 쓴다는 결론을 내릴 수 있다.
아마도 커밋 메시지를 작성할 당시 기분이 좋지 않거나, 자신의 코드가 엉망이었거나, 일이 너무 많았기 때문일 수도 있다.

가장 많이 쓰인 단어는 'no fucking comments' 에 쓰인 fucking 이었다.
그렇다면 fucking 과 가장 많이 함께 쓰인 단어를 분석해보자.



'fucking' 과 함께 쓰인 단어로 'no comments', 'piece of shit', 'stupid typo' 등이 있다. 부정적인 비속어와 함께 쓰인 단어들을 분석해보면 개발자들을 힘들게 하는 것이 어떤 것인지 파악할 수 있다.

커밋 메시지에서 가장 많이 쓰인 감탄사는 무엇일까?

이번에는 커밋 메시지에서 가장 많이 쓰인 감탄사는 무엇인지 분석해보자. 기분이 좋을 때 자주 쓰이는 영어 감탄사 10개를 추려보았다.

Interjection = {'bravo', 'good', 'amazing', 'awesome', 'wonderful', 'well-done', 'great', 'good job ', 'fantastic', 'gorgeous'}

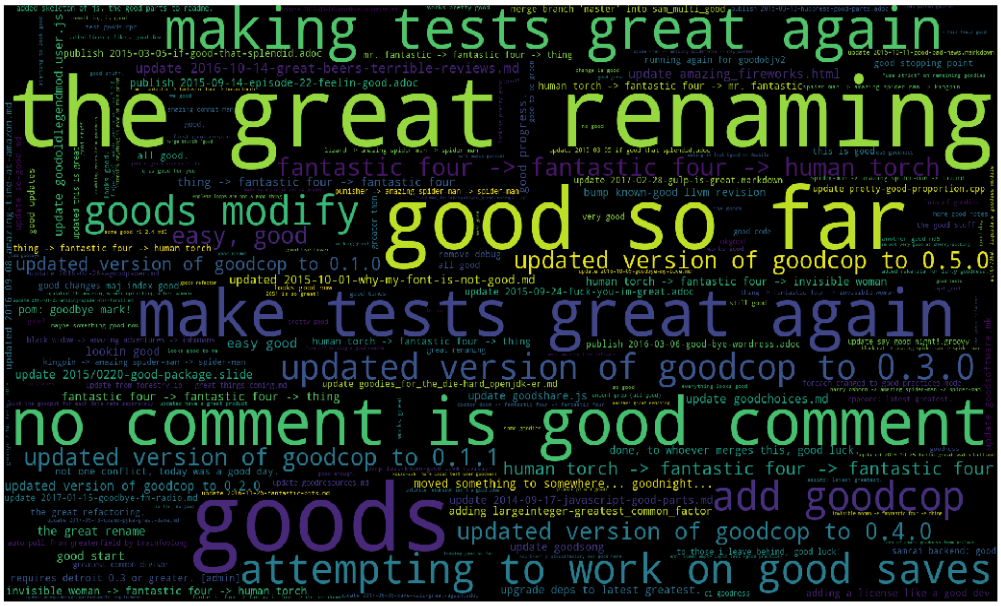
해당 감탄사가 속한 커밋 메시지를 분석해보자. 카운트 수(hits)를 기준으로 상위 30개를 출력하였다.

	message	hits
0	add font awesome	325
1	add font-awesome	301
2	added font awesome	248
3	awesome	217
4	font awesome	216
5	added font-awesome	178
6	add fontawesome	174
7	delete font-awesome.min.css	161
8	looking good	156
9	delete fontawesome-webfont.woff	152
10	delete fontawesome-webfont.ttf	152
11	delete fontawesome.otf	151
12	delete fontawesome-webfont.eot	150
13	update awesome-angular2.md	146
14	delete fontawesome-webfont.svg	145
15	update font awesome	139
16	good stuff	133
17	good night	133
18	added fontawesome	119
19	delete font-awesome.css	114
20	all good	107
21	delete fontawesome-webfont.woff2	106
22	looks good	106
23	fontawesome	96
24	latest and greatest	93
25	font-awesome	90
26	created my first goodie	89
27	update awesome.md	88
28	update font-awesome	88
29	greatest	82

'awesome' 이라는 단어가 제일 많이 사용되는 것으로 나타났다. font awesome 과 관련이 있는 'awesome'을 제외하면 두 번째 순위인 'looking good' 의 'good' 이 많이 쓰임을 알 수 있다.

	message	hits
0	looking good	156
1	good stuff	133
2	good night	133
3	all good	107
4	looks good	106
5	latest and greatest	93
6	created my first goodie	89
7	greatest	82
8	good to go	79
9	good enough	77

그렇다면 'good' 이라는 단어와 가장 많이 쓰인 단어를 분석해보자.



'the great renaming', 'good so far', 'goods', 'making test great again', 'no comment is good comment' 라는 단어들이 함께 쓰였음을 분석 결과를 통해 결론 내릴 수 있다.

Naive Bayes Classifier를 통한 커밋 메시지의 긍정(pos), 부정(neg) 분류

그럼 이제 Naive Bayes Classification(NBC)을 통해 커밋 메시지를 긍정, 부정으로 간단하게 분류해보도록 하자. NBC는 보통 스팸 필터, 문서 분류 등에 사용되는 분류기이다.

두 사건을 서로 독립이라고 가정하고 긍정(pos)과 부정(neg)라는 태그로 분류할 수 있다.

문장	태그
The great renaming	긍정
looking good	긍정
Try something crazy	부정
fix stupid bug	부정
no comment	중립
latest and	중립

예를 들어 The great renaming 은 긍정이지만, try something crazy 는 부정이다.

Naive Bayes classification은 P(A|B) 로 B가 나왔을 때 A일 확률을 구하는 것으로 일종의 조건부 확률로 각 단어가 독립(independent)임을 가정한다. 이에 따라 수식은 아래와 같이 수립할 수 있다.

$$P(w_1, w_2) = P(w_1) \cdot P(w_2)$$
$$P(w_1, w_2|c_i) = P(w_1|c_i) \cdot P(w_2|c_i)$$

위의 Naive Bayes classification를 이용해 위의 커밋 메시지 로그에서 분석해온 비속어와 감탄사에서 가장 많이 나온 문장을 상위 10개씩 추출하여 20개의 문장으로 진행해보자.

```
train = [('no fucking comments','neg'),
         ('piece of shit','neg'),
         ('stupid typo','neg'),
         ('stupid','neg'),
         ('fix stupid bug','neg'),
         ('im an idiot','neg'),
         ('stupid mistake','neg'),
         ('dammit','neg'),
         ('stupid fix','neg'),
         ('more shit','neg'),
         ('looking good','pos'),
         ('good stuff','pos'),
         ('good night','pos'),
         ('all good','pos'),
         ('looks good','pos'),
         ('latest and greatest','pos'),
         ('created my first goodie','pos'),
         ('greatest','pos'),
         ('good to go','pos'),
         ('good enough','pos')]
```

그리고 train 문장에서 사용된 전체 단어를 분석한다.

```
{'all',
 'an',
 'and',
 'bug',
 'comments',
 'created',
 'dammit',
 'enough',
 'first',
 'fix',
 'fucking',
 'go',
 'good',
 'goodie',
 'greatest',
 'idiot',
 'im',
 'latest',
 'looking',
 'looks',
 'mistake',
 'more',
 'my',
 'night',
 'no',
 'of',
 'piece',
 'shit',
 'stuff',
 'stupid',
 'to',
 'typo'}
```

이 단어들을 일종의 '말뭉치'라고 하자. 그리고 말뭉치를 기준으로 train 문장에 속한 단어인지 아닌지를 분석한다.

```
[({'all': False,
   'an': False,
   'and': False,
   'bug': False,
   'comments': True,
   'created': False,
   'dammit': False,
   'enough': False,
   'first': False,
   'fix': False,
   'fucking': True,
   'go': False,
   'good': False,
   'goodie': False,
```

train 의 첫 문장인 'no fucking comments'의 경우 우리가 만든 말뭉치 단어
[all, and, bug, comments, created, dammit, fucking, go, good, ...]를 기준으로 해당 단어의 존재 여부를 분석한 것이다.
결과 첫 부분을 분석하면 train의 첫 문장에는 all는 없고(False), and도 없고(False), comments은 있다(True)가 된다. 이를 이용해서 Navive Bayes 분류기를 이용하여 분석해보자.

```
classifier = nltk.NaiveBayesClassifier.train(t)
classifier.show_most_informative_features()
```

```
Most Informative Features
             good = False           neg : pos    =      3.0 : 1.0
           stupid = False           pos : neg    =      1.9 : 1.0
         greatest = False           neg : pos    =      1.2 : 1.0
              fix = False           pos : neg    =      1.2 : 1.0
             shit = False           pos : neg    =      1.2 : 1.0
               to = False           neg : pos    =      1.1 : 1.0
          looking = False           neg : pos    =      1.1 : 1.0
            idiot = False           pos : neg    =      1.1 : 1.0
               no = False           pos : neg    =      1.1 : 1.0
               in = False           pos : neg    =      1.1 : 1.0
```

GitHub Commit message 의 비속어/감탄사 분석을 통해 추출해낸 데이터에서 상위 10개씩을 뽑아 만든 Train 문장에 붙은 긍정/부정 태그를 이용해서 분류한 결과 'fix' 라는 단어가 없을 때 (False 일 때) 긍정일 비율이 1.2 : 1.0 로 나타났다.

이를 통해 'fix'라는 단어가 포함된 커밋 메시지가 많을 경우 프로젝트에 버그(Bug)가 대체로 많고 프로젝트 진행이 어려운 상황임을 결론내릴 수 있다.