

한글(한글 + 영어) 문장이라면 한글 train으로
영어 문장이라면 영어 train으로
훈련시킨다

2-3 훈련시키기 - 한글

```
t = Twitter()

def train_hangul(text, state):
    pair_word_tag_list = t.pos(text)

    for _word, tag in pair_word_tag_list:
        if _word not in word_ko_dict:
            word_ko_dict[_word].append(0)
            word_ko_dict[_word].append(0)

        if state == '0':
            word_ko_dict[_word][0] += 1
        else:
            word_ko_dict[_word][1] += 1
```

아직 tag값은 사용하지 않고 work token만 가지고
word_ko_dictionary에 긍정값인지 부정값인지 확률을 측정하기 위해 카운팅

2-4 훈련시키기 - 영어

```
def train_not_hangul(text, state):
    text = text.lower()
    words = nltk.word_tokenize(text)
    pair_word_tag_list = nltk.pos_tag(words)

    for _word, tag in pair_word_tag_list:
        if _word not in word_not_ko_dict:
            word_not_ko_dict[_word].append(0)
            word_not_ko_dict[_word].append(0)

        if state == '0':
            word_not_ko_dict[_word][0] += 1
        else:
            word_not_ko_dict[_word][1] += 1
```

아직 tag값은 사용하지 않고 work token만 가지고
word_not_ko_dictionary에 긍정값인지 부정값인지 확률을 측정하기 위해 카운팅

2-5 분류시키기

```
print("classify..")
result_cnt = 0
result_correct = 0
with open("ratings_valid.txt", encoding="utf-8") as f:
    for line in f:
        if line[-1] is "\n":
            line = line[:-1]

        split = line.split("\t")
        id_movie = split[0]
        document_movie = split[1]
        label_movie = split[2]

        result_cnt += 1
        if label_movie == classify(document_movie):
            result_correct += 1
            print("OK : ", document_movie, label_movie)
        else:
            print("NO : ", document_movie, label_movie)

print(result_cnt)
print(result_correct)
```

한줄씩 읽으면서 classify로 긍정인지 부정인지 판단한다
아래 소스는 valid를 check하는 것으로 정확도를 측정하기 위한 코드

2-6 분류시키기 2

```
def classify(text):
    if is_hangul(text):
        return classify_hangul(text)
    else:
        return classify_not_hangul(text)

def classify_hangul(text):
    pair_word_tag_list = t.pos(text)
    pos = prob_pos
    neg = prob_neg
    for _word, tag in pair_word_tag_list:
        if _word in word_ko_dict:
            pos += Decimal(word_ko_dict[_word][1] / (word_ko_dict[_word][0] + word_ko_dict[_word][1])).log10()
            neg += Decimal(word_ko_dict[_word][0] / (word_ko_dict[_word][0] + word_ko_dict[_word][1])).log10()

    if pos > neg:
        return '1'
    else:
        return '0'
```

한글 인지 아닌지는 dictionary를 check하는데만 사용

$p(\text{pos}|\text{words})$ 랑 $p(\text{neg}|\text{words})$ 를 위해
 $p(\text{pos}) * p(\text{word1}|\text{pos}) * p(\text{word2}|\text{neg}) \dots$
 $p(\text{neg}) * p(\text{word1}|\text{neg}) * p(\text{word2}|\text{neg}) \dots$

를 위해서 log를 씌운 값을 더해서
pos가 큰지 neg가 큰지 return한다

3 실행결과

3-1 형태소에 따로 기능을 추가하지 않고 token화만 한 후 분류한 결과

```
OK : 그레이스의 겔모습만 보던 사람들에게 그녀의 마음을 보여준 영화라고 생각함..
OK : 별루다 0
OK : 정말 재밌게 본 컷영화 1
NO : 같은시나리오에 다른 배우들이 연기했을 어땠을까? 1
OK : 마이क्र리의 영화는 항상 조용하면서 강력하다 1
OK : 나 인생의 영화 1
OK : 좀 생동감있는데 소재가 재미있었네요 다음작품 한번 기대해봅니다. 1
OK : 처음으로 영화관에서 잠 0
OK : 그들은 즐겁게 술을 주는데 난 왜 홀걸지 않을까? 0
OK : 보는 내내 울었다 너무 슬프고 감동적이라 ㅠㅠ 1
OK : 아~~ 담주까지 아케 기다려요!! 극본, 대사 대박~~ 감사, 청창, 주인공4명
OK : 하품이 나오는 활작 0
10001
8079
```

앞으로 할 것을 위한 참고자료

- 한글

```
8
9 package com.twitter.penguin.korean.util
10
11 /**
12  * Korean Part-of-Speech
13  *
14  * N Noun: 명사 (Nouns, Pronouns, Company Names, Proper Noun, Person Names, Numerals, Standalone, Dependent)
15  * V Verb: 동사 (하, 먹, 자, 차)
16  * J Adjective: 형용사 (깨끗다, 크다, 적다)
17  * A Adverb: 부사 (참, 매우, 빨리, 반드시, 과연)
18  * D Determiner: 관형사 (세, 많, 많, 첫, 이, 그, 저)
19  * E Exclamation: 감탄사 (헉, ㅋㅋㅋ, 아아아, 일싸구)
20  *
21  * C Conjunction: 접속사
22  *
23  * J SubstantiveJosa: 조사 (의, 에, 에서)
24  * I AdverbialJosa: 부사격 조사 (-은, -히, -로)
25  * e Eomi: 어미 (다, 오, 아, 하필ㅋㅋ)
26  * r PreEomi: 선어미 (없)
27  *
28  * p NounPrefix: 접두사 ('호'대박)
29  * v VerbPrefix: 동사 접두어 ('쳐'먹어)
30  * s Suffix: 접미사 (-적)
31  *
32  * f Foreign: 한글이 아닌 문자들
33  *
34  * 지시사는 Determinant로 대체하기로 함
35  * Determinant is used for demonstratives.
36  *
37  * Korean: Korean chunk (candidate for parsing)
38  * Foreign: Mixture of non-Korean strings
39  * Numbers: 숫자
40  * Emotion: Korean Single Character Emotions (ㅋㅋㅋ, ㅎㅎㅎ, ㅠㅠㅠ)
41  * Alpha: Alphabets 알파벳
42  * Punctuation: 문장부호
43  * Hashtag: Twitter Hashtag 해시태그 #Korean
44  * ScreenName: Twitter username (@nlpenguin)
45  *
46  * Unknown: Could not parse the string.
47  */
48 object KoreanPos extends Enumeration {
49   type KoreanPos = Value
50 }
```

twitter-korean-text는 normalization, tokenization, stemming, phrase extraction 이렇게 네가지 기능을 지원합니다.

정규화 normalization (입니딕ㅋㅋ -> 입니다 ㅋㅋ, 사롱해 -> 사랑해)

- 한국어를 처리하는 예시입니딕ㅋㅋㅋㅋㅋ -> 한국어를 처리하는 예시입니다 ㅋㅋ

토큰화 tokenization

- 한국어를 처리하는 예시입니다 ㅋㅋ -> 한국어Noun, **는**Josa, 처리Noun, 하는Verb, 예시Noun, 일Adjective, 니디Eomi ㅋㅋ KoreanParticle

어근화 stemming (입니딕 -> 이딕)

- 한국어를 처리하는 예시입니딕 ㅋㅋ -> 한국어Noun, **는**Josa, 처리Noun, 하디Verb, 예시Noun, 이디Adjective, ㅋㅋKoreanParticle

어구 추출 phrase extraction

- 한국어를 처리하는 예시입니딕 ㅋㅋ -> 한국어, 처리, 예시, 처리하는 예시

Introductory Presentation: [Google Slides](#)

- 영어

POS tag	description	example
CC	Coordinating conjunction	for, and, nor, but, or, yet, so
CD	cardinal number	기수 one, two, 35, 10, etc.
DT	Determiner	명사 앞에 나와서 명사를 수식하는 역할 a that any, all, this, etc.
EX	Existential there	존재의 there there is an apple
FW	Foreign word	외래어 dong, alam 등 토빗에서 자주 발생하는 misspelled word들이 많이 포함
IN	preposition or subordinating conjunction	전치사 because, ago, under, within, once, etc.
JJ	adjective	형용사 sick, cheap, great, etc.
JJR	adjective, comparative	형용사 비교급 better, more, happier, etc.
JJS	adjective, superlative	형용사 최상급 best, highest, most, etc.
LS	List Item Marker	I, II, III, IV, etc.
MD	Modal	조동사 can, could, would, etc.
NN	Noun, singular or mass	명사 show, food, girl, etc.
NNP	proper noun, singular	고유명사(단수) dad, mart, 요일, 등, etc.
NNPS	proper noun, plural	고유명사(복수)
NNS	noun, plural	명사(복수)
PDT	Predeterminer	한치한치 such
POS	Possessive Ending	Nouns ending in's you, he, I
PP	Personal Pronoun	인칭대명사 you, he, I
PPS	Possessive Pronoun	my, your, mine
RB	adverb	부사 대부분 -ly로 끝남. Even, totally, personally
RBR	adverb, comparative	hotter, sooner
RBS	adverb, superlative	most, hardest
R P	particle	활판화사 up, on turn it on
SYM	symbol	used for mathematical, scientific, or technical symbols
TO	to	
UH	interjection	감탄사 oh, uh, uh, well, yes
VB	verb base form	원반동사
VBD		과거동사 got
VBS		동명사 형식동사 being
VBN		과거분사 enjoyed
VBP	non-3rd person singular present	come, feel
VBZ	3rd person singular present	comes, feels
WDT	wh-determiner	which that(문맥대명사로 쓰임)
WP	wh-pronoun	관계대명사 what, who, whom
WPS	Possessive wh-pronoun	whoever, who, what
WRB	wh-adverb	how, where, why, whenever

4 추가 작업

1. Infinity 가 되는 것을 막기 위해 양측에 threshold 값 추가
8079 -> 8441

```
쉬운게 0 -Infinity  
아니 -0.385477645661
```

```
? -0.33924  
-3.4057760  
10001  
8441
```

2. 한글이 아닌것 training에 stemmer 추가
8441 -> 8442

```
-3.4057760  
10001  
8442
```

```
for w in words:  
    if w not in stop_words:  
        words_filtered.append(stemmer.stem(w))  
pair_word_tag_list = nltk.pos_tag(words_filtered)
```

3. 한글에 training에 stemmer 추가
8442 -> 7856

4. 한글 classify에도 stemmer추가
7856 -> 8337

```
train_hangul(text, state):  
pair_word_tag_list = t.pos(text, norm=True, stem=True)
```

5. 한글 training 에 classify에 Suffix, Josa, Punctuation 제거
8337 -> 8336

```
ignore_ko_tag = ['Suffix', 'Josa', 'Punctuation']
```

6. 한글 문장이 뒤쪽에 영향을 많이 받는 것 같아서 뒤쪽에 가중치를 둬
8336 -> 8315

```
for word, tag in pair_word_tag_list:  
    power += (1 / length) * inc_power  
    if tag in ignore_ko_tag:
```

7. 그 외에도 여러 변수에 대한 가중치를 추가해서 (1의 threshold값, 6의 power값 등)
8315 -> 8370대