전처리 클래스

<전체 부분>

```
import json
import re
from pykospacing import spacing # 설치 방법 : pip install git+https://github.com/haven-jeon/PyKoSpacing.git

class Preprocessing_Insta () :

    def __init__(self):
        self.escape_code = ['\n', '\xa0', '\"', '\'', '\t', '\r', '\$', '\\', '\u200d']
        return

    # hashtag 추출(#포함)
    def extract_hashtag(self, content) :
        hashtag_list = re.findall('\#[\w가-힣a-zA-Z0-9]*', str(content))
        if hashtag_list:
            return hashtag_list
        else :
            return ''

    # post 추출
    def extract_post(self, content) :
        post = re.sub('\#[\w가-힣a-zA-Z0-9]*',"",str(content))
        post = self.del_escape(post)
        post = re.sub("\@[\w가-힣a-zA-Z0-9]*","",post)
        return post #string type

    # 태그된 userID 추출
    def extract_tagged_userID(self, content) : #태그된 userID 추출의 경우 hashtag 추출과 달리 @를 제거해준 값 리턴
        re_content = re.findall('\@[\w가-힣a-zA-Z0-9]*', str(content))
        userID_list= []
        for userID in re_content:
            userID_list.append(re.sub("@","",userID))
        return userID_list

    # hashtag(#) 제거
    def remove_hash(self, hashtag_list) :
        for hashtag in hashtag_list:
            tmp = []
            for j in  hashtag:
                tmp.append(re.sub("#","",j))
        return tmp

    # pykospacing패키지를 사용한 띄어쓰기 처리
    def auto_spacing(self, content) :
        return spacing(content)

    # Escape Code 처리
    def del_escape(self, content):
        for e in self.escape_code:
            content = content.replace(e, ' ')
        return content

    # emoji 삭제
    def del_emoji(self, content) :
        only_BMP_pattern = re.compile("["
        u"\U00010000-\U0010FFFF"  #BMP characters 이외
                           "]+", flags=re.UNICODE)
        return only_BMP_pattern.sub(r'', content)

    # content list 전처리
    def preprocess_content(self, content_list, spacing = False, sub_hash = False) :
        post_list =[]
        hashtag_list= []
        for content in content_list :
            original_post = self.extract_post(content)
            if spacing :
                post_list.append(self.auto_spacing(original_post))
            else :
                post_list.append(original_post)
            if sub_hash :
                hashtag_list.append(self.remove_hash(self.extract_hashtag(content)))
            else :
                hashtag_list.append(self.extract_hashtag(content))
        return post_list, hashtag_list

if __name__ == "__main__":
    content_list = ['다이어트 해야되는데...😂😂\n.\n.\n.\n#멋짐휘트니스연산점 #연산동pt','럽스타 그자체❤❤ #럽스타그램 #운동하는커플 #연산동pt']
    test_class = Preprocessing_Insta()
    post_ls, hashtag_ls = test_class.preprocess_content(content_list, spacing= True)
    print(post_ls)
    print(hashtag_ls)
    print("-----------------------------------------------------------------------------------------------")
    print("*************이모지 삭제 활용 예시*************")
    for post in post_ls :
        print(test_class.del_emoji(post))
```

class Preprocessing_Insta () :

- **클래스 명에 해당한다. (수정할 필요가 있다. - 보고서의 내용과 다르다. 클래스 명을 정하고 보고서 내용을 수정하는 방향으로..)**

```
def __init__(self):
        self.escape_code = ['\n', '\xa0', '\"', '\'', '\t', '\r', '\$', '\\', '\u200d']
        return

 # Escape Code 처리
    def del_escape(self, content):
        for e in self.escape_code:
            content = content.replace(e, ' ')
        return content

 # post 추출
    def extract_post(self, content) :
        post = re.sub('\#[\w가-힣a-zA-Z0-9]*',"",str(content))
        post = re.sub("\@[\w가-힣a-zA-Z0-9]*","",post)
        post = self.del_escape(post)
        return post
```

def __init__(self):

- self.escape_code = ['\n', '\xa0', '\"', '\'', '\t', '\r', '\$', '\\', '\u200d']

  의 경우 띄어쓰기, tab 등 escape code를 처리하기 위해 미리 리스트로 할당해 주었다.

- def del_escape(self, content):

  del_escape 함수는 str형의 content를 매개변수로 받고, content에 self.escape_code = ['\n', '\xa0', '\"', '\'', '\t', '\r', '\$', '\\', '\u200d']에 해당하는 escape_code가 등장할 시 띄어쓰기로 replace해주는 함수이다. 포스트 추출 함수에 사용된다.

- def extract_post(self, content)

  위 함수는 str형의 content를 매개변수로 받고 3가지 기능을 수행한다.

  1. content에 포함된 #문자열을 모두 제거한다. (해시태그 제거)
  2. content에 포함된 @문자열을 모두 제거한다. (태그된 사용자 아이디 제거)
  3. content에 포함된 escape code가 있을 시 del_escape함수를 통해 띄어쓰기로 변환한다.

```
# hashtag 추출(#포함)
    def extract_hashtag(self, content) :
        hashtag_list = re.findall('\#[\w가-힣a-zA-Z0-9]*', str(content))
        if hashtag_list:
            return hashtag_list
        else :
            return ''
```

- def extract_hashtag(self, content)

  str형식의 content에서 해쉬태그를 #을 포함하여(ex: #과일, #운동) 추출, list 형식으로 반환하는 함수이다. #을 포함한 이유는 TC_tagger의 품사태깅을 위함이다.

  해시태그가 없을 경우 ''을 반환한다. (포스트 추출과 해시태그 추출 함수를 합치지 않은 이유는, 해시태그만 추출하기 원하는 사용자가 있을 것이라고 생각해서이다. - 2차 정규화 과정)

```
# hashtag(#) 제거
def remove_hash(self, hashtag_list) :
    for hashtag in hashtag_list:
        tmp = []
        for j in  hashtag:
            tmp.append(re.sub("#","",j))
    return tmp
```

- list 형식의 hashtag_list를 매개변수로 받는다. 단순히 list의 각 원소에서 #을 제거해주는 기능을 한다. 보통 해시태그 추출 함수(extract_hashtag(self, content))의 결과값이 [#해시1, #해시2, #해시3] 형식으로 반환되기 때문에, 그 이후에 사용된다.

```
 def extract_tagged_userID(self, content) : #태그된 userID 추출의 경우 hashtag 추출과 달리 @를 제거해준 값 리턴
        re_content = re.findall('\@[\w가-힣a-zA-Z0-9]*', str(content))
        userID_list= []
        for userID in re_content:
            userID_list.append(re.sub("@","",userID))
        return userID_list
```

- extract_tagged_userID(self, content)

  str형식의 content에 포함된 사용자ID를 추출하여 반환한다. 해시태그 추출 함수와 달리 @이를 포함하지 않고 바로 사용자ID 만을 반환한다.

```
# pykospacing패키지를 사용한 띄어쓰기 처리
def auto_spacing(self, content) :
    return spacing(content)
```

- def auto_spacing(self, content)

  pykospacing 패키지를 사용한 띄어쓰기 처리 기능을 수행한다.

```
# emoji 삭제
def del_emoji(self, content) :
    only_BMP_pattern = re.compile("["
    u"\U00010000-\U0010FFFF"  #BMP characters 이외
                      "]+", flags=re.UNICODE)
    return only_BMP_pattern.sub(r'', content)
```

- def del_emoji(self, content)

  유니코드를 사용하여 이모지를 제거해주는 함수

```
def preprocess_content(self, content_list, spacing = False, sub_hash = False) :
        post_list =[]
        hashtag_list= []
        for content in content_list :
            original_post = self.extract_post(content)
            if spacing :
                post_list.append(self.auto_spacing(original_post))
            else :
                post_list.append(original_post)
            if sub_hash :
                hashtag_list.append(self.remove_hash(self.extract_hashtag(content)))
            else :
                hashtag_list.append(self.extract_hashtag(content))
        return post_list, hashtag_list
```

- def preprocess_content(self, content_list, spacing = False, sub_hash = False) :

  대량의 게시글에서 전처리된 포스트와 해시태그를 반환하는 함수로 3개의 인자를 갖는다.

  1. 문자열 list 형식의 content_list를 매개변수로 받는다.
  2. bool 형식의 spacing 매개변수로 pykospacing 패키지를 사용한 띄어쓰기 처리 기능을 수행할 것인지 결정할 수 있다.
  3. bool 형식의 sub_has 매개변수로 #을 제거한 해시태그를 뽑을 것인지 결정할 수 있다.

  위 함수를 쓴 예시와 결과는 다음과 같다.

```
if __name__ == "__main__":
    content_list = ['다이어트 해야되는데...😂😂\n.\n.\n.\n#멋짐휘트니스연산점 #연산동pt','럽스타 그자체❤❤ #럽스타그램 #운동하는커플 #연산동pt']
    test_class = Preprocessing_Insta()
    post_ls, hashtag_ls = test_class.preprocess_content(content_list, spacing= True)
    print(post_ls)
    print(hashtag_ls)
>>>['다이어트 해야 되는데...😂😂 . . .', '럽스타 그자체 ❤❤']
>>>[['#멋짐휘트니스연산점', '#연산동pt'], ['#럽스타그램', '#운동하는커플', '#연산동pt']]
```

비쥬얼 스튜디오 코드의 터미널 창에서는 출력 부분이 조금 깨진는 현상이 발생한다.

```
for post in post_ls :
        print(test_class.del_emoji(post))
>>>다이어트 해야 되는데... . . .
>>>럽스타 그 자체 ❤❤
```

위 코드는 이모지를 제거 함수를 사용한 예시이다.

😂😂은 잘 제거된 반면, ❤❤는 특수 문자로 인식 제거되지 않았다.