2020. 7. 12. 05 KS Practice. Text Preprocessing - Colaboratory

텍스트 전처리 (Text Preprocessing)

텍스트를 자연어 처리를 위해 용도에 맞도록 사전에 표준화 하는 작업 텍스트 내 정보를 유지하고, 중복을 제거하여 분석 효율성을 높이기 위해 전처리를 수행

1) 토큰화 (Tokenizing)

텍스트를 자연어 처리를 위해 분리 하는 것을 토큰화는 단어별로 분리하는 "단어 토큰화(Word Tokenization)"와 문장별로 분리하는 "문장 토큰화(Sentence Tokenization)"로 구분

(이후 실습에서는 단어 토큰화를 "토큰화"로 통일하여 칭하도록 한다)

2) 품사 부착(PoS Tagging)

각 토큰에 품사 정보를 추가 분석시에 불필요한 품사를 제거하거나 (예. 조사, 접속사 등) 필요한 품사를 필터링 하기 위해 사용

3) 개체명 인식 (NER, Named Entity Recognition)

각 토큰의 개체 구분(기관, 인물, 지역, 날짜 등) 태그를 부착 텍스트가 무엇과 관련되어있는지 구분하기 위해 사용 예를 들어, 과일의 apple과 기업의 apple을 구분하는 방법이 개체명 인식임

4) 원형 복원 (Stemming & Lemmatization)

각 토큰의 원형 복원을 함으로써 토큰을 표준화하여 불필요한 데이터 중복을 방지 (=단어의 수를 줄일수 있어 연산을 효율성을 높임) 어간 추출(Stemming) : 품사를 무시하고 규칙에 기반하여 어간을 추출 표제어 추출 (Lemmatization) : 품사정보를 유지하여 표제어 추출

5) 불용어 처리 (Stopword)

자연어 처리를 위해 불필요한 요소를 제거하는 작업 불필요한 품사를 제거하는 작업과 불필요한 단어를 제거하는 작업으로 구성 불필요한 토큰을 제거함으로써 연산의 효율성을 높임

1 영문 전처리 실습

NLTK lib (https://www.nltk.org/) 사용

https://colab.research.google.com/drive/12MufGcL7f7OR7RTbdGUDR9u2cFW-tHXo#printMode=true 1/8

2020. 7. 12. 05 KS Practice. Text Preprocessing - Colaboratory

1.1 실습용 영문기사 수집

온라인 기사를 바로 수집하여 실습데이터로 사용

https://www.forbes.com/sites/adrianbridgwater/2019/04/15/what-drove-the-ai-renaissance/ import requests from bs4 import BeautifulSoup

url u= 'https://www.forbes.com/sites/adrianbridgwater/2019/04/15/what-drove-the-ai-renaissan response = requests.get(url) soup = BeautifulSoup(response.text,'html.parser')

eng\_news e= soup.select('p') #[class="speakable-paragraph"]') eng\_text = eng\_news[3].get\_text()

eng\_text e 1.2 영문 토큰화

https://www.nltk.org/api/nltk.tokenize.html

!pip install nltk

# #word\_tokenize() : 단어와 구두점(온점(.), 컴마(,), 물음표(?), 세미콜론(;), 느낌표(!) 등과 같은 기호)으로 구 import nltk nltk.download('punkt') from nltk.tokenize import word\_tokenize

text = 'Good muffins cost $3.88\nin New York. Please buy me\ntwo of them.\n\nThanks.' word\_tokens = word\_tokenize(text) print(word\_tokens)

# #word\_tokenize() : 단어와 구두점(온점(.), 컴마(,), 물음표(?), 세미콜론(;), 느낌표(!) 등과 같은 기호)으로 구 import nltk from nltk.tokenize import word\_tokenize token1 = word\_tokenize(eng\_text) print(token1)

WordPunctTokenizer? W# #WordPunctTokenizer() : 알파벳과 알파벳이 아닌문자를 구분하여 토큰화 import nltk from nltk.tokenize import WordPunctTokenizer

text = 'Good muffins cost $3.88\nin New York. Please buy me\ntwo of them.\n\nThanks.' wordpuncttoken = WordPunctTokenizer().tokenize(text) print(wordpuncttoken)

TreebankWordTokenizer? T

https://colab.research.google.com/drive/12MufGcL7f7OR7RTbdGUDR9u2cFW-tHXo#printMode=true 2/8

2020. 7. 12. 05 KS Practice. Text Preprocessing - Colaboratory

# #TreebankWordTokenizer() : 정규표현식에 기반한 토큰화 import nltk from nltk.tokenize import TreebankWordTokenizer

text = 'Good muffins cost $3.88\nin New York. Please buy me\ntwo of them.\n\nThanks.' treebankwordtoken = TreebankWordTokenizer().tokenize(text) print(treebankwordtoken)

1.3 영문 품사 부착 (PoS Tagging)

분리한 토큰마다 품사를 부착한다

https://www.nltk.org/api/nltk.tag.html

태크목록 : https://pythonprogramming.net/natural-language-toolkit-nltk-part-speech-tagging/

from nltk import pos\_tag nltk.download('averaged\_perceptron\_tagger')

taggedToken t= pos\_tag(word\_tokens) print(taggedToken)

1.4 개체명 인식 (NER, Named Entity Recognition)

http://www.nltk.org/api/nltk.chunk.html

nltk.download('words') nnltk.download('maxent\_ne\_chunker')

from nltk import ne\_chunk neToken = ne\_chunk(taggedToken) print(neToken)

1.5 원형 복원

각 토큰의 원형을 복원하여 표준화 한다.

1.5.1 어간추출 (Stemming)

규칙에 기반 하여 토큰을 표준화 ning제거, ful 제거 등

https://www.nltk.org/api/nltk.stem.html

규칙상세 : https://tartarus.org/martin/PorterStemmer/def.txt

from nltk.stem import PorterStemmer ps = PorterStemmer()

print("running -> " + ps.stem("running")) print("beautiful -> " + ps stem("beautiful")) https://colab.research.google.com/drive/12MufGcL7f7OR7RTbdGUDR9u2cFW-tHXo#printMode=true 3/8

2020. 7. print( 12. beautiful -> + ps.stem( beautiful 05 KS Practice. ))

Text Preprocessing - Colaboratory

print("believes -> " + ps.stem("believes")) print("using -> " + ps.stem("using")) print("conversation -> " + ps.stem("conversation")) print("organization -> " + ps.stem("organization")) print("studies -> " + ps.stem("studies"))

1.5.2 표제어 추출 (Lemmatization)

품사정보를 보존하여 토큰을 표준화

http://www.nltk.org/api/nltk.stem.html?highlight=lemmatizer

nltk.download('wordnet') nfrom nltk.stem import WordNetLemmatizer wl = WordNetLemmatizer()

print("running -> " + wl.lemmatize("running")) print("beautiful -> " + wl.lemmatize("beautiful")) print("believes -> " + wl.lemmatize("believes")) print("using -> " + wl.lemmatize("using")) print("conversation -> " + wl.lemmatize("conversation")) print("organization -> " + wl.lemmatize("organization")) print("studies -> " + wl.lemmatize("studies"))

1.6 불용어 처리 (Stopword)

stopPos = ['IN', 'CC', 'UH', 'TO', 'MD', 'DT', 'VBZ','VBP']

# #최빈어 조회. 최빈어를 조회하여 불용어 제거 대상을 선정 from collections import Counter Counter(taggedToken).most\_common()

stopWord = [',','be','able']

word = [] for tag in taggedToken:

if tag[1] not in stopPos:

if tag[0] not in stopWord:

word.append(tag[0])

print(word)

1.7 영문 텍스트 전처리 종합

import nltk nltk.download('averaged\_perceptron\_tagger') nltk.download('words') nltk.download('maxent\_ne\_chunker') nltk.download('wordnet')

https://colab.research.google.com/drive/12MufGcL7f7OR7RTbdGUDR9u2cFW-tHXo#printMode=true 4/8

2020. 7. 12. 05 KS Practice. Text Preprocessing - Colaboratory

from nltk.tokenize import TreebankWordTokenizer sumtoken = TreebankWordTokenizer().tokenize("Obama loves fried chicken of KFC") print(sumtoken)

from nltk import pos\_tag sumTaggedToken = pos\_tag(sumtoken) print(sumTaggedToken)

from nltk import ne\_chunk sumNeToken = ne\_chunk(sumTaggedToken) print(sumNeToken)

from nltk.stem import PorterStemmer ps = PorterStemmer() print("loves -> " + ps.stem("loves")) print("fried -> " + ps.stem("fried"))

from nltk.stem import WordNetLemmatizer wl = WordNetLemmatizer() print("loves -> " + wl.lemmatize("loves")) print("fried -> " + wl.lemmatize("fried"))

#불용어 처리 sumStopPos = ['IN'] sumStopWord = ['fried']

word = [] for tag in sumTaggedToken:

if tag[1] not in sumStopPos:

if tag[0] not in sumStopWord:

word.append(tag[0])

print(word)

2 한글 전처리 실습

영문은 공백으로 토큰화가 가능하지만, 한글의 경우 품사를 고려하여 토큰화 해야한다.

2.1 실습용 한글기사 수집

온라인 기사를 바로 수집하여 실습데이터로 사용

http://news.chosun.com/site/data/html\_dir/2018/07/10/2018071004121.html

import requests from bs4 import BeautifulSoup

url u

= 'http://news.chosun.com/site/data/html\_dir/2018/07/10/2018071004121.html' response = requests.get(url) response.encoding = 'utf-8' # 한글이므로 encoding을 utf-8로 지정 soup = BeautifulSoup(response.text,'html.parser')

https://colab.research.google.com/drive/12MufGcL7f7OR7RTbdGUDR9u2cFW-tHXo#printMode=true 5/8

2020. 7. 12. 05 KS Practice. Text Preprocessing - Colaboratory

kor\_news k= soup.select('div[class="par"]') kor\_text = kor\_news[0].get\_text()

kor\_text k2.2 한글 토큰화 및 형태소 분석

#konlpy #설치 !pip install konlpy

한글 자연어처리기 비교

https://konlpy.org/ko/latest/morph/

# #코모란(Komoran) 토큰화 from konlpy.tag import Komoran komoran= Komoran() kor\_text = "인간이 컴퓨터와 대화하고 있다는 것을 깨닫지 못하고 인간과 대화를 계속할 수 있다면 컴퓨터는 지능적인 것으 komoran\_tokens = komoran.morphs(kor\_text) print(komoran\_tokens)

# #한나눔(Hannanum) 토큰화 from konlpy.tag import Hannanum hannanum= Hannanum() kor\_text = "인간이 컴퓨터와 대화하고 있다는 것을 깨닫지 못하고 인간과 대화를 계속할 수 있다면 컴퓨터는 지능적인 것으 hannanum\_tokens = hannanum.morphs(kor\_text) print(hannanum\_tokens)

# #Okt 토큰화 from konlpy.tag import Okt okt= Okt() kor\_text = "인간이 컴퓨터와 대화하고 있다는 것을 깨닫지 못하고 인간과 대화를 계속할 수 있다면 컴퓨터는 지능적인 것으 okt\_tokens = okt.morphs(kor\_text) print(okt\_tokens)

# #Kkma 토큰화 from konlpy.tag import Kkma kkma= Kkma() kor\_text = "인간이 컴퓨터와 대화하고 있다는 것을 깨닫지 못하고 인간과 대화를 계속할 수 있다면 컴퓨터는 지능적인 것으 kkma\_tokens = kkma.morphs(kor\_text) print(kkma\_tokens)

2.3 한글 품사 부착 (PoS Tagging)

PoS Tag 목록

https://docs.google.com/spreadsheets/u/1/d/1OGAjUvalBuX- oZvZ\_-9tEfYD2gQe7hTGsgUpiiBSXI8/edit#gid=0

# #

코모란(Komoran) 품사 태깅 komoranTag = [] https://colab.research.google.com/drive/12MufGcL7f7OR7RTbdGUDR9u2cFW-tHXo#printMode=true 6/8

2020. 7. 12. g []

05 KS Practice. Text Preprocessing - Colaboratory

for token in komoran\_tokens:

komoranTag += komoran.pos(token) print(komoranTag)

# #한나눔(Hannanum) 품사 태깅 hannanumTag = [] for token in hannanum\_tokens:

hannanumTag += hannanum.pos(token) print(hannanumTag)

# #Okt 품사 태깅 oktTag = [] for token in okt\_tokens:

oktTag += okt.pos(token) print(oktTag)

# #Kkma 품사 태깅 kkmaTag = [] for token in kkma\_tokens:

kkmaTag += kkma.pos(token) print(kkmaTag)

2.4 불용어(Stopword) 처리

분석에 불필요한 품사를 제거하고, 불필요한 단어(불용어)를 제거한다

#불용어 #처리 stopPos = ['Suffix','Punctuation','Josa','Foreign','Alpha','Number']

# #최빈어 조회. 최빈어를 조회하여 불용어 제거 대상을 선정 from collections import Counter Counter(oktTag).most\_common()

더블클릭 또는 Enter 키를 눌러 수정

stopWord = ['의','이','로','두고','들','를','은','과','수','했다','것','있는','한다','하는','그','있다'

word w= [] for tag in oktTag:

if tag[1] not in stopPos:

if tag[0] not in stopWord:

word.append(tag[0])

print(word) p 2 N-gram

https://colab.research.google.com/drive/12MufGcL7f7OR7RTbdGUDR9u2cFW-tHXo#printMode=true 7/8

2020. 7. 12. 05 KS Practice. Text Preprocessing - Colaboratory

import nltk from nltk import bigrams, word\_tokenize from nltk.util import ngrams nltk.download('punkt')

sentence = "I am a boy." tokens = word\_tokenize(sentence)

bigram = bigrams(tokens) trigram = ngrams(tokens, 3)

for t in bigram: print(t)

for t in trigram:

print(t)

import nltk nltk.download('movie\_reviews') nltk.download('punkt') from nltk.corpus import movie\_reviews

sentences = [] for tokens in movie\_reviews.sents():

bigram = ngrams(tokens, 2, pad\_left=True, pad\_right=True, left\_pad\_symbol="SS", right\_p sentences += [t for t in bigram]

sentences[:20]

movie\_reviews.sents() msentences[-5:]

https://colab.research.google.com/drive/12MufGcL7f7OR7RTbdGUDR9u2cFW-tHXo#printMode=true 8/8