



한국어 텍스트 전처리

≡ 태그	
📅 날짜	@2023년 4월 4일
👤 사람	정민화

언어학 기초: 단어의 단위

전반적인 과정

1. 필요한 문자(한글, 알파벳, 숫자, 특수부호 등)만 선택
2. 문자열 분절(토큰화)
 - 1) 문서단위 분절
 - 2) 문장단위 분절
 - 3) 어절단위 분절
 - 4) 형태소단위 분절
 - 5) 용언의 기본형(사전형, 원형) 단위 분절
 - 6) 그 외
3. 불용어 제거
4. 그 외에 할 수 있는 것
 - 문장 분절
 - 띄어쓰기 수정
 - 신조어 수정(ㅋㅋㅋ, ㅎㅎㅎ)
 - 번역(한자, 외국어) 등..

임베딩

1. 임베딩이란?
2. BOW(Bag Of Words)
3. DTM(Document Term Matrix): 문서 단어 행렬
4. TF-IDF
5. n-gram

머신러닝에서 임베딩하기

딥러닝에서 임베딩하기

- 1) 문자열의 길이 분포 확인
- 2) 인덱스 부여 + 패딩하기

언어학 기초: 단어의 단위

▼ 언어학 기초: 단어의 단위(미리 읽어보시기를 권장)

단어를 구분할 때 어떤 단위로 분절할 수 있는가?

정의

단위	정의
token(음절)	single occurrence of word form. 공백 단위로 구분되는 단위들 (영어 텍스트 분석 기본단위)
type	unique form in corpus. 말뭉치 내의 고유한 형태를 하나의 단어로 봄
lemma (원형, 사전형)	형태론적인 변형(문법적)을 한 단어를 하나로 봄. 예) am, are, is → be
lexeme	다의어 등을 구분. 예) bank(강둑) 과 bank(은행)은 다름
morpheme (형태소)	·의미·의 최소단위(단어X). 이때 의미는 사전적 뿐만 아니라 문법적(기능적)의미를 포함

적용 - 영어

예문: I disliked the bank across the bank.

단위	분절
token	['I', 'disliked', 'the', 'bank', 'across', 'the', 'bank'] → 영어 텍스트 분석의 가장 기본단위
type	['I', 'disliked', 'the', 'bank', 'across'] → the, bank 가 하나의 단어로 합쳐짐
lemma	['I', 'dislike', 'the', 'bank', 'across'] → disliked 에서 과거활용을 제거함. 다의어 bank 고려X
lexeme	['I', 'dislike', 'the', 'bank', 'across', 'bank'] → 다의어를 구분
morpheme	['I', 'dis-', 'like', '-ed', 'the', 'bank', 'across', 'the', 'bank']

→ 형태소에서는 부정을 의미하는 접두사 'dis-' 와 과거형 굴절어미 '-ed'가 분절됨

적용-한국어

예문: 낮에는 밥을 먹고 밤엔 밥을 먹었다.

단위	분절
token	['낮에는', '밥을', '먹고', '밤엔', '밥을', '먹었다'] → 띄어쓰기 단위로 구분
type	['낮에는', '밥을', '먹고', '밤엔', '밥을', '먹었다'] → 모두 고유함
lemma	['낮', '에', '는', '밥', '을', '먹다', '밤'] → 문법적 기능을 하는 형태를 모두 제거
lexeme	['낮', '에', '는', '밥', '을', '먹다', '밤', '밤'] → 다의어를 구분
morpheme	['낮', '에', '는', '밥', '을', '먹-', '-고', '밤', '에', '-ㄴ', '밤', '을', '먹-', '-었-', '-다'] → 한국어 텍스트 분석의 가장 기본 단위

- 한국어의 lemma는 좀 따지기 어려움.
일반적으로 활용(품사, 수, 시제에 따라 형태를 변형하는 것)의 대상인 용언(동사, 형용사)만 원형(lemma)로 복원하는 경우가 일반적임.
조사가 단어인가? 아닌가?는 언어학 내에서도 논란이 큼. 다만, 국립국어원 설명은 조사와 그 앞의 체언(명사)를 분절하여도 체언에 독립성을 유지하기 때문에 조사 또한 단어로 봄.

(국립국어원 참고

https://korean.go.kr/front/onlineQna/onlineQnaView.do?mn_id=216&qna_seq=216649&pageIndex=1)

- 한국어 형태소 분석의 경우
-고: 연결어미
-ㄴ: 보조사 '는'의 축약형
-었-: 과거시제 선어말어미
-다: 어말어미
등이 분절됨
- 일반적으로는 한국어에서는 형태소분석을 통해 문장을 분절하고, 용언(동사, 형용사)을 원형(lemma)로 복원해줌

** 개념 하나 더: 어근(root)과 어간(stem)

전반적인 과정

1. 필요한 문자(한글, 알파벳, 숫자, 특수부호 등)만 선택

주로 정규표현식(regular expression)을 활용

- 패턴을 만들고 해당 패턴을 만족하는 문자열을 추출(전화번호, 이메일, 대사, 날짜 등)

```
# 날짜 추출(0000-00-00패턴인 경우)
import re
s = '안녕하세요. 새해 복 많이 받으세요. 글은 2021-01-06에 썼어요'
m = re.search('[0-9]{4}-[0-9]{2}-[0-9]{2}', s) # 숫자 4개-2개-2개인 패턴
print(m.group()) # 출력결과 : 2021-01-06

# email인 경우
import re
pattern = r"([\w\.-]+)@([\w\.-]+)(\.[\w\.-]+)"
pattern = r"[a-zA-Z0-9_-]+@[a-z]+\.[a-z]+" #위의 패턴과 같음

str = "Please contact info@sololearn.com for assistance"
match = re.search(pattern, str)
if match:
    print(match.group())
```

- 제거할 문자를 정하고 이를 제외 혹은 대체하기(replace쓰면 됨)
- **선택할 문자를 정하고 이를 선택하기(판다스 문법으로도 가능)**
 - 우리 데이터의 경우 문장부호와 특수문자 (. 등) 존재 → 제거: 뉴스의 토픽을 잘 설명한다고 생각되는 한글, 영어만 선택
 - kiwi 형태소 분석기의 경우 영어 알파벳을 따로 태깅할 수 있어 위와 같이 전처리해도 무방함


```
# 한글, 알파벳만을 선택하는 경우 - 숫자, 특수 문자 등은 제외
df.drop_duplicates(subset = ['new_title'], inplace=True) # 중복 제거

# 특수한 이모지 제거 위해 한국어, 영어, 공백이 아닌 것은 모두 제거(공백으로 대체)
df['new_title']=df['new_title'].str.replace(pat=r'[^ㄱ-힣|a-zA-Z| ]+', repl= ' ', regex=True)

df['new_title'].replace('', np.nan, inplace=True) # 공백은 Null 값으로 변경
df = df.dropna(how='any') # Null 값 제거
print('전처리 후 데이터 샘플의 개수 :', len(df))
```

2. 문자열 분절(토큰화)

사실 단어의 분류는 아주 많음! 단어라는 개념 자체가 모호하다는 것이 큰 이유임

 예문) 나는 어제 연세대학교 대우관 별관에 있었는데
오늘도 연세대학교 대우관 별관에 있다.

단순히 공백으로 구분한 것이 단어라면

‘연세대학교’, ‘대우관’, ‘별관’은 각각 다른 단어로 처리됨

그러나, 사람에 따라 ‘연세대학교 대우관 별관’/‘대우관 별관’이 하나라고 생각할 수도 있음!

따라서, “어떤 단위로 분절(토큰화)하여 임베딩할 것인지”는 텍스트분석 및 처리에 있어 중요한 문제임

1) 문서단위 분절

Doc2vec 등

2) 문장단위 분절

최근에 문자열의 길이가 길어지면서 선호되는 방법(sentencepiece, SBERT, spanBERT 등)

→ 문서, 문장단위 임베딩은 문서, 문장의 의미를 최대한으로 보존할 수 있어 최근 선호됨

3) 어절단위 분절

단순 띄어쓰기 단위(영어에서 자주 쓰이는 단위, 언어학에서는 token임)

→ 하지만 한국어는 단순 띄어쓰기 단위로 분절하면 조사, 어미 등을 분리하기 어려움

구현이 쉬워(.split())로 분리하면 되니까) 간단한 실습에서 자주 사용

일반적으로 영어 텍스트의 경우 이 방법을 사용(nltk가 다 해줌)

NLTK :: nltk.tokenize package
<https://www.nltk.org/api/nltk.tokenize.html>

4) 형태소단위 분절

의미의 최소단위(여기서의 의미는 문법적인 의미를 포함)로의 분절

박규병님의 w2v를 이용한 사전학습벡터는 konlpy내 꼬꼬마(kkma) 형태소분석기에서 분절한 형태소단위로 이루어짐

한국어 텍스트 분석의 기본단위. konlpy를 주로 사용함

- konlpy내 라이브러리 비교는 아래의 링크를 참조(알고리즘적 비교)

한국어 형태소 분석기(POS) 분석 - 3편. 형태소 분석기 비교

개발: Shin285 (github에 공개), shinware개발언어: java알고리즘: HMM여러 어절을 하나의 품사로 분석 가능함
으로써 형태소 분석기의 적용 분야에 따라 공백이 포함된 고유명사(영화 제목, 음식점명, 노래 제목, 전문 용어 등)를
더 정확하게 분석

<https://velog.io/@metterian/한국어-형태소-분석기POS-분석-3편.-형태소-분석기-비교>

velog

형태소 분석의 장점

[1] 원하는 품사만을 고를 수 있음

텍스트에서 주로 의미를 가지는 품사는 명사, 동사, 형용사, 부사 등임(목적에 따라 다름)

이처럼 품사를 선별하면 불필요한 형태소는 제하고, 필요한 형태소에 대해서는 더 정확한 분석을 할 수 있기 때문에 유용함
문자열의 길이가 긴 경우 해당 품사만을 선별하여 문자열의 길이를 줄이고, 연산량을 줄일 수 있음

[2] 단어집합(언어학에서는 코퍼스, 딥러닝에서는 vocab)의 크기를 줄일 수 있음

같은 형태소(위 문장에서는 있/VA)는 단어집합(사전)을 만들 때 하나로 통합되기 때문에 수를 줄일 수 있음

형태소 분석의 단점

[1] 분석기에 따라 다른 형태소 태그

아래는 가장 유명한 한국어 형태소분석기인 konlpy내 여러 라이브러리(한나눔, 꼬꼬마, 코모란, 메캅, 트위터(OKT))를 비교한 결과
한나눔과 twitter(OKT)가 아주 특수함

예문) 나는 밥을 먹는다

Hannanum	Kkma	Komorran	Mecab	Twitter
나 / N	나 / NP	나 / NP	나 / NP	나 / Noun
는 / J	는 / JX	는 / JX	는 / JX	는 / Josa
밥 / N	밥 / NNG	밥 / NNG	밥 / NNG	밥 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
먹 / P	먹 / VV	먹 / VV	먹 / VV	먹는 / Verb
는다 / E	는 / EPT	는다 / EC	는다 / EC	다 / Eomi
	다 / EFN			

*한나눔: 용언이 P, 동사 PV, 형용사 PA

N*은 명사계열,

V*는 용언 계열(동사 VV, 형용사 VA)

J*는 조사 계열(주어, 목적어 등을 표기하는 문법적인 의미의 최소단위)

EC/P/F는 어미(연결어미, 선어말어미, 어말어미)

** mecab의 품사 태그 참고

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/3814e868-6d52-47c6-88b7-84106272fd75/%ED%83%9C%EA%B7%B8_v2.0.html

[2] 띄어쓰기 없는 문장, 신조어, 비문 등에 대한 대처가 어려움

위의 문장에서 ‘연세대학교 대우관 별관’의 경우

‘연세대학교’, ‘대우’, ‘관’, ‘별관’

으로 분류함.

3. “아이폰 기다리다 지쳐 애플공홈에서 언락폰질러버렸다 6+ 128기가실버ㅋ”

각 분석기가 사전에 포함되지 않은 단어를 어떻게 해결하는지 확인해볼까요?

Hannanum	Kkma	Komoran	Mecab	Twitter
아이폰 / N	아이 / NNG	아이폰 / NNP	아이폰 / NNP	아이폰 / Noun
기다리 / P	폰 / NNG	기다리 / VV	기다리 / VV	기다리 / Verb
다 / E	기다리 / VV	다 / EC	다 / EC	다 / Eomi
지치 / P	다 / ECS	지치 / VV	지쳐 / VV+EC	지쳐 / Verb
어 / E	지치 / VV	어 / EC	애플 / NNP	애플 / Noun
애플공홈 / N	어 / ECS	애플 / NNP	공 / NNG	공홈 / Noun
에서 / J	애플 / NNP	공 / NNG	홈 / NNG	에서 / Josa
언락폰질러버 렸다 / N	공 / NNG	홈 / NNG	에서 / JKB	언락폰 / Noun
6+ / N	홈 / NNG	에서 / JKB	언락 / NNG	질 / Verb
128기가실버 / N	에서 / JKM	언 / NNG	폰 / NNG	러 / Eomi
	언락 / NNG	락 / NNG	질러버렸 / VV+EC+VX+EP	버렸 / Verb
	폰 / NNG	폰 / NNG	다 / EC	다 / Eomi

—> 이러한 경우 새로운 단어를 사전에 추가하여 해결 가능

ckonlpy

https://github.com/lovit/customized_konlpy

konlpy의 단점

- java, jdk 등 귀찮은 환경설정이 필요함...
- 더군다나 서버에서 설치하려면 sudo권한이 필요함(mecab)...
- 업데이트가 안되고 있다

*** kiwi(Korean Intelligent Word Identifier) 장점 ***

- 설치 간편
- 세종품사태그 기반 + 일부 태그 추가(한자, 알파벳, 특수문자, 해시태그 등)
- 불용어사전 내장

```
# kiwi 사용하기
kiwi = Kiwi()
kiwi.prepare() # 전처리 위한 준비
stopwords = Stopwords() # 불용어 사전 불러오기

# kiwi를 활용한 형태소 분절
tokenized_sen = []
def kiwi_preprocess(sen):
    tokens = kiwi.tokenize(sen, stopwords = stopwords)
    t_list = [token.form for token in tokens] # .form: 형태소만 반환
    tokenized_sen.append(t_list)
    # 만약 특정 품사만 선택한다면 if조건 추가
```

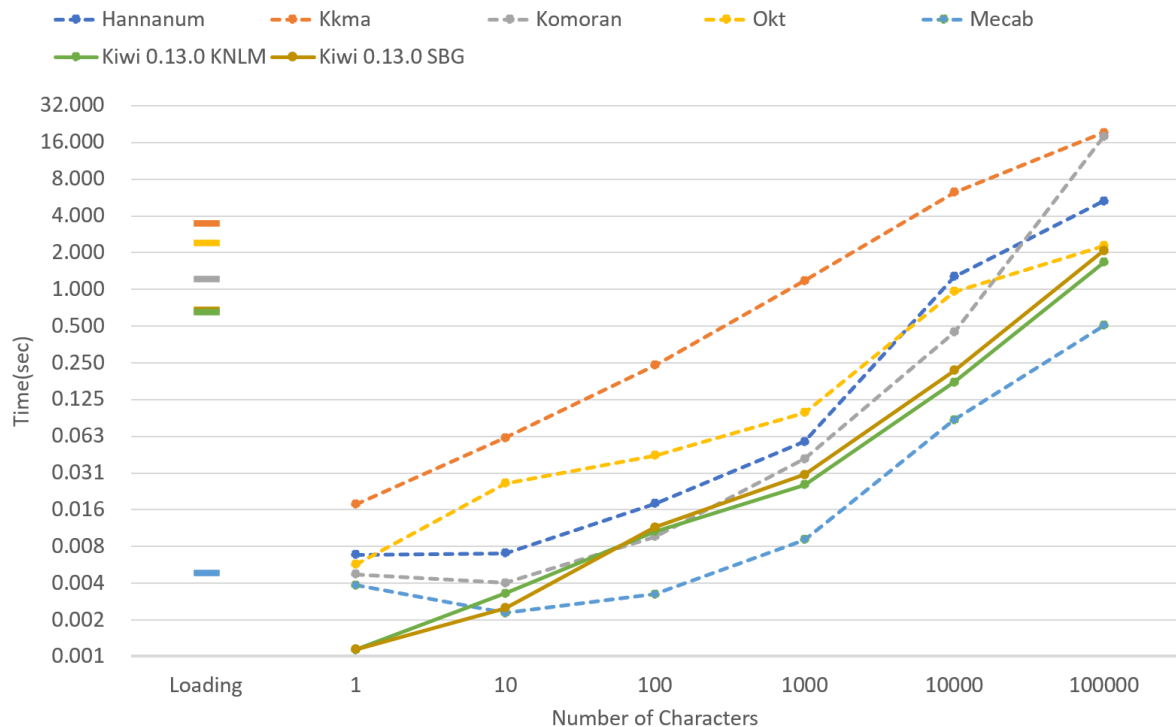
```
# 데이터 프레임에 해당 결과 추가
str_list = df['new_title'].tolist()
for s in str_list:
    kiwi_preprocess(s)
df['tokenized'] = tokenized_sen
```

결과

	new_title	topic_idx	tokenized
	김영남 우리 민족 위상 과시 뜨거운 분위기 이어가길	1	[김영남, 민족, 위상, 과시, 뜨겁, 분위기, 잇, 가, ㄴ]
	적극행정 추진전략 및 성과공유대회 참석한 이낙연 총리	1	[적극, 행정, 추진, 전략, 및, 성과, 공유, 대회, 참석, 이낙연, 총리]
	박대통령 한일 합의에 소녀상 언급없이 선동하면 안돼	1	[박, 대통령, 한일, 합의, 소녀상, 언급, 선동, 안]
	민주 국회의원 재보선 곳 후보 공모에 명 신청	1	[민주, 국회, 의원, 재보선, 곳, 후보, 공모, 명, 신청]
	박찬호 현진이 승 하니깐 생각이 나는데	0	[박찬호, 현진이, 승, 니깐, 생각, 나, 데]
	현행 헌법과 다른 점은 지방자치 경제민주화 개념 강화	1	[현행, 헌법, 다른, 점, 지방, 자치, 경제, 민주, 개념, 강화]
	고군분투 시즌 서재덕 MVP 덕규리로 활짝 웃다종합	0	[고군분투, 시즌, 서재덕, MVP, 덕규리, 활짝, 웃, 종합]
	트럼프 유조선 공격 사소한 일 이란과 충돌우려 속 수위조절	1	[트럼프, 유조선, 공격, 사소, 이란, 충돌, 우려, 속, 수위, 조절]
	이희호 여사 청경호 일로 만료 당분간 경호 유지될 듯	1	[이희호, 여사, 청, 경호, 만료, 당분간, 경호, 유지, 듯]
	통일농구 오늘은 친선 남북대결 김정은 관전 가능성	1	[통일, 농구, 오늘, 친선, 남북, 대결, 김정은, 관전, 가능]
	임도현 감독 신영석 중심으로 뽀돌 뽀돌 한일전 승리 따냈다	0	[임도현, 감독, 신영석, 중심, 뽀돌, 뽀돌, 한일전, 승리, 따, 내]

*** konlpy와의 비교 ***

<https://github.com/bab2min/Kiwi>



5) 용언의 기본형(사전형, 원형) 단위 분절

형태소 분석에서 체언(명사, 대명사, 수사), 관계언(조사) 등은 활용(형태를 변형)하여 사용하지 않기 때문에 분절해도 상관없음

이와 달리, 시제, 의미에 따라 활용하여 사용하는 용언(동사, 형용사)은 형태소분리를 하게되면 어근과 어미/접사가 분리되어 원래 동사/형용사의 모습으로 존재하지 못함

어근(root): 단어 의미의 중심이 되는 부분.

비슷한 개념인 어간(stem): 용언이 활용할 때 중심이 되는 부분. 활용에서 어미에 선행하는 부분

예) 깨끗하다 → 깨끗(어근), -하다(형용사 파생 접사)

깨끗하다 → 깨끗하-(어간), -다(어말어미)

우리가 분석 혹은 처리 결과로 얻고 싶은 결과는 ‘**깨끗하다**’이지 깨끗- 혹은 깨끗하- 의 형태가 아님

따라서, 각각의 용언에 대해 **기본형(사전형)**이라고도 함, 영어로는 **lemma**)으로 복원시켜주어야함

- **kiwi형태소분석기에서 코드적으로 lemmarization하는 방법**

```
# kiwi 사용하기
kiwi = Kiwi()
kiwi.prepare() # 전처리 위한 준비
stopwords = Stopwords() # 불용어 사전 불러오기

# 형태소 분석 + 기본형 복원
def kiwi_preprocess(sen):
    tokens = kiwi.tokenize(sen, stopwords = stopwords)
    morph_list = [word + ('다' if tag.startswith('V') else '') for word, tag, _, _ in tokens]
    return " ".join(morph_list)
    # 동사 'VV', 형용사 'VA'

# 데이터 프레임에 추가하기
str_list = df['new_title'].tolist()
new_list = []
for s in str_list:
    new_list.append(kiwi_preprocess(s))
df['tokenized'] = new_list
```

결과

	new_title	topic_idx	tokenized
	김영남 우리 민족 위상 과시 뜨거운 분위기 이어가길	1	김영남 민족 위상 과시 뜨겁다 분위기 있다 가다 ㄹ
	적극행정 추진전략 및 성과공유대회 참석한 이낙연 총리	1	적극 행정 추진 전략 및 성과 공유 대회 참석 이낙연 총리
	박대통령 한일 합의에 소녀상 언급없어 선동하면 안돼	1	박 대통령 한일 합의 소녀상 언급 선동 안
	민주 국회의원 재보선 곳 후보 공모에 명 신청	1	민주 국회 의원 재보선 곳 후보 공모 명 신청
	박찬호 현진이 승 하니깐 생각이 나는데	0	박찬호 현진이 승 니깐 생각 나다 데
	현행 헌법과 다른 점은 지방자치 경제민주화 개념 강화	1	현행 헌법 다르다 점 지방 자치 경제 민주 개념 강화
	고군분투 시즌 서재덕 MVP 덕규리로 활짝 웃다종합	0	고군분투 시즌 서재덕 MVP 덕규리 활짝 웃다 종합
	트럼프 유조선 공격 사소한 일 이란과 충돌우려 속 수위조절	1	트럼프 유조선 공격 사소 이란 충돌 우려 속 수위 조절
	이희호 여사 청경호 일로 만료 당분간 경호 유지될 듯	1	이희호 여사 청 경호 만료 당분간 경호 유지 듯
	통일농구 오늘 친선 남북대결 김정은 관전 가능성	1	통일 농구 오늘 친선 남북 대결 김정은 관전 가능
	임도현 감독 신영석 중심으로 돌돌 뭉쳐 한일전 승리 따냈다	0	임도현 감독 신영석 중심 돌돌 뭉치다 한일전 승리 따다 내다

***** konlpy.okt의 기본형으로 복원하는 기능을 제공하고 있음**

품사태그가 적은건 아쉽지만 기본형 복원 가능한 점은 좋다고 생각함

```
from konlpy.tag import Okt
okt = Okt()
text = '나는 어제 연세대학교 대우관 별관에 있었는데 오늘도 연세대학교 대우관 별관에 있다.'

print(okt.morphs(text))
print(okt.morphs(text, stem = True)) #stem_True 옵션을 통해 원형 복원 가능
```

```
['나', '는', '어제', '연세대학교', '대', '우관', '별관', '에', '있었는데', '오늘', '도', '연세대학교', '대', '우관', '별관', '에', '있다', '.']
['나', '는', '어제', '연세대학교', '대', '우관', '별관', '에', '있다', '오늘', '도', '연세대학교', '대', '우관', '별관', '에', '있다', '.']
```

기본형 복원의 장점

보다 깔끔한 분석이 가능해짐+단어집합의 수 줄일 수 있음

기본형을 복원함으로써 텍스트 분석 할 때 결과값이 더 깔끔해짐

활용된 동사, 형용사가 모두 하나의 단어가 되기 때문에 텍스트의 단어집합의 수가 줄어듦

기본형 복원의 단점

보다 복잡하고, 오류가 많아질 수 있는 작업임

특히 한국어의 경우 불규칙 활용이 존재하여 이를 복원하는 것은 더 어려움

okt의 경우에도 직접 분석할 때 돌다를 '도르다'로 변형한거 보고 이마짚.. 🤔

→ 그만큼 자동적인 변형이 어려움

이 과정이 어렵기 때문에 텍스트분석에서는 주로 "명사"만을 추출하여 분석에 활용하는 것 같음

- 간단한 형태소 분석기 결과 비교 - okt, komoran, mecab, kiwi(경험적 비교)

→ 분석목적에 맞게 적절한 분석기를 선택하는 것이 필요함

[morphs.ipynb](#)

6) 그 외

음절단위: BPE(Byte Pair Encoding) 등

자모, 알파벳단위: fasttext

3. 불용어 제거

원하는 단위로 분절하였다면 텍스트 분석/처리에 불필요한 용어(= 불용어)를 제거해주어야함

영어의 경우 이미 nltk에서 정의된 불용어가 있기 때문에 이를 활용하면 됨

한국어의 경우 형태소 분석기 내에 불용어가 정의되어있는 경우(kiwi 형태소 분석기)가 있지만

대부분의 경우(konlpy라이브러리 전체)는 별도의 불용어를 정의하지 않음

이러한 경우 불용어를 별도의 리스트를 활용하여 정의한 뒤 형태소 분석 후 불용어사전에 없는 형태소를 리턴하는 방식으로 코드를 설계

```
import re
tokenizer = Okt()
def text_preprocessing(text, tokenizer):
    stopwords = ['을', '를', '이', '가', '은', '는'] # 조사를 불용어로 정의
    txt = re.sub('[^가-힣a-z]', ' ', text) #한글 , 영어만 추출
    token = tokenizer.morphs(txt) # 형태소단위 분절
    token = [t for t in token if t not in stopwords] #불용어 제거
    return token

ex_text = "이번에 새롭게 개봉한 영화의 배우들은 모두 훌륭한 연기력과 아름다운 목소리를 갖고 있어!!"
example_pre = text_preprocessing(ex_text, tokenizer)
```

*** 다만, 불용어가 이미 정의된 경우/직접 정의하는 경우에도 결과값을 보고 불용어를 추가하는 경우가 필연적으로 생김(분석을 의뢰한 측에서 단어 하나하나씩 걸고 넘어지기 때문에... 😊)

👉아래는 보편적으로 정의된 한국어 불용어 리스트

Korean Stopwords

아 휴 아이구 아이구 아이고 어 나 우리 저희 따라 의해 을 를 예 의 가 으로 로 에게 뿐이다 의거하여 근거하여 기준으로 예하면 예를 들면 예를 들자면 저 소인 소생 저희 지말고 하지마 하지마라 다른 물론 또한 그리고 비길수 없다 해서는 안된다 뿐만 아니라 만이 아니다 만은 아니다

▶ <https://www.ranks.nl/stopwords/korean>

→ 맹신하지는 말고, 주어진 과제에 따라 변형해야함

4. 그 외에 할 수 있는 것

문장 분절

특정한 단위(\n, \t, 기본적으로는 문장부호(.,?!))가 있으면 제일 좋음

하지만 애매하다면?

kss(korean sentence splitter)라이브러리를 사용해보자

<https://github.com/likejazz/korean-sentence-splitter>

띄어쓰기 수정

PyKospacing

<https://github.com/haven-jeon/PyKoSpacing>

신조어 수정(ㅋㅋㅋ, ㅎㅎㅎ)

반복되는 문자(ㅋㅋㅋ, ㅎㅎㅎ 등)의 횟수를 축약, 한글만 추출 등의 기능을 제공

soynlp.normalizer

soynlp/normalizer_usage.ipynb at master · lovit/soynlp

한국어 자연어처리를 위한 파이썬 라이브러리입니다. 단어 추출/ 토큰나이저 / 품사판별/ 전처리의 기능을 제공합니다. - soynlp/normalizer_usage.ipynb at master · lovit/soynlp

https://github.com/lovit/soynlp/blob/master/tutorials/normalizer_usage.ipynb

lovit/soynlp

한국어 자연어처리를 위한 파이썬 라이브러리입니다. 단어 추출/ 토큰나이저 / 품사판별/ 전처리의 기능을 제공합니다.

11 Contributors 315 Used by 856 Stars 184 Forks

번역(한자, 외국어) 등..

한자의 경우 'hanja' 라이브러리 사용 가능

<https://github.com/suminb/hanja>

```
# 설치 먼저
!pip install hanja

# 임포트
import hanja
from hanja import hangul

# 실제 사용
hanja.translate('大韓民國은 民主共和國이다.', 'substitution')
>>> '대한민국은 민주공화국이다.'
```

임베딩

1. 임베딩이란?

텍스트를 컴퓨터가 이해하고, 효율적으로 처리하게 하기 위해 적절히 숫자로 변환해주는 것.

어떤 방식으로 텍스트를 표현하느냐에 따라 자연어처리의 성능이 크게 달라지기 때문에 많은 연구가 이루어짐

Q. 어떻게 단어를 숫자 표현으로 나타낼 수 있는가?

- 빈도(count) 기반의 방법들 📌

2. BOW(Bag Of Words)

단어의 등장순서를 고려하지 않는 빈도수 기반의 단어 표현 방법

- (1) 각 단어에 고유한 정수 인덱스를 부여
- (2) 각 인덱스의 위치에 단어 토큰의 등장 횟수를 기록한 벡터를 만들

예) 예문1: 낮에는 밥을 먹는다.

→ [낮, 예, 는, 밥, 을, 먹다, -는-, -다]

예문2: 밤에는 야식을 먹는다.

→ [밤, 예, 는, 야식, 을, 먹다, -는-, -다]

→ 형태소 분석 진행 후 원형으로 복원



vocab = {'낮':0, '예':1, '는':2, '밥':3, '을':4, '먹다':5, '-는-':6, '-다':7, '밤':8, '야식':9}
bag of words vector: {1, 2, 2, 1, 2, 2, 2, 1, 1}

3. DTM(Document Term Matrix): 문서 단어 행렬

서로 다른 문서들의 BOW를 결합한 표현 방법

각 단어들의 빈도를 문서별로 행렬로 표현한 것

예)

문서1 : 먹고 싶은 사과 → [먹다, -고, 싶다, 은, 사과]

문서2 : 먹고 싶은 바나나 → [먹다, -고, 싶다, 은, 바나나]

문서3 : 길고 노란 바나나 바나나 → [길다, -고, 노랗다, -ㄴ, 바나나, 바나나]

문서4 : 나는 과일이 좋다 → [나, 는, 과일, 이, 좋다]

형태소 분석 진행 후 원형으로 복원한 뒤 DTM을 만들면 아래와 같음

	-고	과일	길다	노랗다	먹다	-ㄴ	는
문서1	1	0	0	0	1	0	0
문서2	1	0	0	0	1	0	0
문서3	1	0	1	1	0	1	0
문서4	0	1	0	0	0	0	1

문서들의 특징을 서로 비교할 수 있는 수치로 표현가능

DTM의 한계

- 1) 대부분의 값이 0이 될 수 있음(희소행렬, 희소벡터)
- 2) 단순 빈도수 기반의 접근법 → 빈도가 높다/문서내 자주 등장한다고 모두 중요한 단어일까?

4. TF-IDF

TF-IDF(Term Frequency-Inverse Document Frequency)

DTM내에 있는 각 단어의 중요도를 계산할 수 있는 가중치

- 1) $tf(d, t)$: 특정 문서 d 에서 특정단어 t 의 등장 횟수
- 2) $df(t)$: 특정 단어 t 가 등장한 문서의 수
- 3) $idf(d, t)$: $df(t)$ 에 반비례 하는 수

$$idf(d, t) = \log\left(\frac{n}{1 + df(t)}\right)$$

n : 총 문서의 수

단, idf 는 단순히 df 의 역수는 아님.

\log 와 분모에 1을 더해주는 이유:

- 총 문서의 수(n)가 커질수록 idf 의 값이 기하급수적으로 커지기 때문에 \log 를 사용
- 불용어와 같이 빈도가 큰 단어들은 자주 쓰이지

- 특정 단어가 전체 문서에서 등장하지 않을 경우에 분모가 0이 되는 상황 방지 위해 +1

→ 단어의 빈도(tf) * idf를 곱해주면 가중치 값을 구할 수 있음

tf-idf의 장점

불용어 등 문장에서 자주 사용되지만 문서내 빈도수가 큰 단어들을 상쇄할 수 있음(한국어의 경우 조사, 어미 등) → 빈도와 중요도 모두 고려가능

tf-idf의 단점

여전히 희소표현임

여전히 빈도에 기반함

Q. 빈도만으로 텍스트의 의미를 충분히 표현할 수 있을까?

5. n-gram

맥락을 보고싶을 때 사용함

텍스트를 n개의 단어문자로 끊어서 이를 하나의 토큰으로 간주하여 빈도를 계산

예) An adorable little boy is spreading smiles

unigrams : an, adorable, little, boy, is, spreading, smiles

bigrams : an adorable, adorable little, little boy, boy is, is spreading, spreading smiles

trigrams : an adorable little, adorable little boy, little boy is, boy is spreading, is spreading smiles

4grams : an adorable little boy, adorable little boy is, little boy is spreading, boy is spreading smiles

n-gram의 장점

window를 설정하여 단어의 맥락을 함께 볼 수 있음

속담, 구문 등 여러 단어로 구성된 단어들을 하나의 토큰으로 볼 수 있음

n-gram의 단점

n의 개수를 늘릴수록 vocab의 수가 늘어남(불용어 정의 또한 더 복잡해짐)

여전히 빈도에 기반함

→ 따라서, 신경망에 기반한 단어 표현을 통해 임베딩의 차원을 줄이고, 맥락정보를 반영하는 방식을 선호함

머신러닝에서 임베딩하기

- 활용 데이터: KLUE TC(Topic Classification)

Korean Language Understanding Evaluation -자연어 이해 평가 데이터셋(KLUE)


한국어 고유의 특성을 반영할 수 있는 공개데이터셋 by Upstage, Naver, KAIST...

문장유사도, 개체명인식, 자연어 추론 등 다양한 과제를 할 수 있는 데이터셋을 제공

KLUE TC: 2016년 1월부터 2020년 12월까지 네이버 뉴스에 올라간 연합뉴스의 헤드라인을 수집한 데이터. 헤드라인이 어떤 토픽에 해당하는지 모델을 통해 분류

토픽종류: 7개 (정치, 경제, 사회, 생활문화, 세계, IT과학, 스포츠)

KLUE Benchmark

 <https://klue-benchmark.com/>

*** 사전 전처리 ***

- 7개의 토픽 중 정치, 스포츠 토픽의 문장들만 선별 → 이진분류과제로 변환
 - 스포츠 토픽: 0(6933개), 정치 토픽: 1(6751개), 경제 토픽(6222)
- 뉴데이터의 특성상, 한자가 포함되어있기 때문에 **hanja** 라이브러리를 이용하여 한글로 변환

	title	topic_idx		new_title	topic_idx
	김영남 우리 민족 위상 과시...뜨거운 분위기 이어가길	6	0	김영남 우리 민족 위상 과시...뜨거운 분위기 이어가길	1
	적극행정 추진전략 및 성과공유대회 참석한 이낙연 총리	6	1	적극행정 추진전략 및 성과공유대회 참석한 이낙연 총리	1
	朴대통령 한일 합의에 소녀상 언급없어...선동하면 안돼	6	2	박대통령 한일 합의에 소녀상 언급없어...선동하면 안돼	1
	민주 국회의원 재보선 4곳 후보 공모에 7명 신청	6	3	민주 국회의원 재보선 4곳 후보 공모에 7명 신청	1
	박찬호 현진이 10승 하나만 생각이 나는데...	5	4	박찬호 현진이 10승 하나만 생각이 나는데...	0
	현행 헌법과 다른 점은 ②지방자치·경제민주화 개념 강화	6	5	현행 헌법과 다른 점은 ②지방자치·경제민주화 개념 강화	1
	고군분투 시즌 서재덕 MVP 덕큐리로 활짝 웃다종합	5	6	고군분투 시즌 서재덕 MVP 덕큐리로 활짝 웃다종합	0
	트럼프 유조선 공격 사소한 일...이란과 충돌우려 속 수위조절	6	7	트럼프 유조선 공격 사소한 일...이란과 충돌우려 속 수위조절	1
	이희호 여사 靑경호 24일로 만료...당분간 경호 유지될 듯	6	8	이희호 여사 청경호 24일로 만료...당분간 경호 유지될 듯	1
	통일농구 오늘은 친선 남북대결...김정은 관전 가능성	6	9	통일농구 오늘은 친선 남북대결...김정은 관전 가능성	1

파일 📁

[230216_klue_tc.csv](#)

- tf-idf를 사용하여 임베딩 후 머신러닝 모델(random forest, naive bayes, svm에 넣기)

딥러닝에서 임베딩하기

임베딩하기 위한 본격적인 작업

pytorch에서는 텍스트를 처리하기 위한 torchtext라이브러리가 있으나, 버전에 따라 불안정해서(버전따라 어떤 함수는 되고 어떤 함수는 안되는게 너무 심해서;) 이번 실습에는 tensorflow를 사용하여 해당 과정을 진행할 것임

[230404_text_rnn_new.ipynb](#)

1) 문자열의 길이 분포 확인

각 신문기사의 길이분포를 확인함.

문장이 긴 경우 분포의 중위값(median)값, 길지 않은 경우 최대 길이로 맞추어주는 경향

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
text_len = train_data[train_data['topic_idx']==0]['tokenized'].map(lambda x: len(x))
ax1.hist(text_len, color='skyblue')
ax1.set_title('sports news title')
ax1.set_xlabel('length of samples')
ax1.set_ylabel('number of samples')
print('스포츠 기사제목의 중위값 길이 :', np.median(text_len))

text_len = train_data[train_data['topic_idx']==1]['tokenized'].map(lambda x: len(x))
ax2.hist(text_len, color='purple')
ax2.set_title('politics news title')
fig.suptitle('Words in texts')
ax2.set_xlabel('length of samples')
ax2.set_ylabel('number of samples')
print('정치 기사제목의 중위값 길이 :', np.median(text_len))
plt.show()
# 넉넉잡아 16으로 하기로 함
```

2) 인덱스 부여 + 패딩하기

각 형태소에 고유한 인덱스를 부여하여 사전형태로 만들고, 문장에 각 인덱스를 부여함

```
X_train = train_data['tokenized'].values
y_train = train_data['topic_idx'].values
X_test = test_data['tokenized'].values
y_test = test_data['topic_idx'].values
```

```
# 가장 빈도가 높은 11,800개의 단어만 선택하도록하는 Tokenizer 객체
tokenizer = Tokenizer(num_words = 11800, oov_token="<OOV>") #이후 test set에서 out of vocab에 대한 처리

# 그에 맞게 단어 인덱스를 구축
tokenizer.fit_on_texts(X_train)

# 고유한 사전으로 저장
word_dic = tokenizer.word_index

# 사전의 크기를 max_feature변수에 저장
max_features = len(word_dic)
# '<OOV>': 1 포함하여 총 11800

max_len = 16

# 문장에 각 인덱스를 부여
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)

# max_len보다 길이가 짧은 문장은 패딩처리
X_train = pad_sequences(X_train, maxlen=max_len)
X_test = pad_sequences(X_test, maxlen=max_len)
```

단어 사전 구축 결과

```
{'<OOV>': 1,
 '대통령': 2,
 '종합': 3,
 '북': 4,
 '박': 5,
 '문': 6,
 '감독': 7,
 '프로': 8,
 '농구': 9,
 '서': 10,
 '전': 11,
 '대표': 12,
 '경기': 13,
 '팀': 14,
 '배구': 15,
 '정상': 16,
 '첫': 17,
 '위': 18,
 '축구': 19,
 '여자': 20,
 '회담': 21,
 '월드컵': 22,
 '시즌': 23,
 '류현진': 24,
 '선수': 25,
```

패딩 후 인덱스 부여 결과

```
X_train[3]
✓ 0.0s
array([ 0,  0,  0,  0,  0, 72, 2954, 954, 432, 4075, 456,
       74, 1684, 362,  3,  45], dtype=int32)
```