

텍스트 전처리 실습

Natural Language Processing & AI Lab.,
Korea University



KOREA
UNIVERSITY



Natural Language
Processing
& Artificial Intelligence

Data Pre-processing



Why use preprocessing?

- 대표적인 예시
 - 아버지가방에들어가신다.



Why use preprocessing?

- 대표적인 예시
 - 아버지가 방에 들어가신다.

아버지가 방에 들어가신다. 아버지가 방에 들어가신다.



두 가지 뜻으로 읽힐 수 있는 것처럼 말이죠 _____



Why use preprocessing?

- 띄어쓰기나 맞춤법에 따라 다른 의미
 - 중의적 표현이나 반어법 등의 표현으로 의미를 찾기 어려움
- ⇒ 그래서, 텍스트 전처리 필요!
- ⇒ 이를 통해 의미를 찾음



Data Preprocessing

- Crawling을 통해 얻은 corpus에서 필요에 맞게 전처리가 필요
- 용도에 맞게 텍스트를 사전에 처리하는 작업
 - 주어진 원래 데이터를 그대로 사용하기보다는 원하는 형태로 변형해서 분석하는 경우가 많음
 - 유사한 말
 - 데이터 가공(Data Manipulation), 데이터 핸들링(Data Handling), 데이터 클리닝(Data Cleaning) 등...



Data Preprocessing

- 컴퓨터가 텍스트를 이해할 수 있도록 data preprocessing 방법
 - HTML 태그, 특수문자, 이모티콘
 - 정규표현식
 - 불용어 (Stopword)
 - 어간추출 (Stemming)
 - 음소표기법 (Lemmatizing)



Library - 1

- KoNLPy
 - <https://konlpy-ko.readthedocs.io/ko/v0.4.3/#>
 - 한국어 자연어처리를 위한 대표적 python Library
 - Twitter, Komoran, Mecab 등 다양한 형태소 분석기 내장하고 있음



Library - 2

- NLTK(Natural Language Toolkit)
 - <https://www.nltk.org>
 - 영어로 된 텍스트의 자연처리를 위한 대표적인 python Library
 - classification, tokenization, stemming tagging, parsing, and semantic reasoning 등 50개가 넘는 library를 제공하며 쉬운 interfaces 제공



Library - 3

- Gensim
 - <https://radimrehurek.com/gensim/>
 - 주로 Topic modeling, Corpus 및 Word Embedding model을 지원해줌
 - 한국어 및 다양한 언어를 지원해줌



Data Preprocessing task 1

- Tokenization(토큰화)
 - 주어진 코퍼스(corpus)에서 토큰(token)이라 불리는 단위로 나누는 작업을 토큰화(tokenization)라고 부름
 - 보통 의미있는 단위의 토큰을 정의함
 - 토큰의 기준을 단어(word)로 하는 경우, 단어 토큰화(word tokenization)라고 함
 - 단어(word)는 단어 단위 외에도 단어구, 의미를 갖는 문자열로도 간주되기도 함



Data Preprocessing task 1

- Tokenization(토큰화)
 1. 단어를 띄어쓰기를 기준으로 단어를 떼어 냄

Input text : I loved you. Data-mining



Output text : "I", "loved", "you.", "data-mining"



Data Preprocessing task 1

- Tokenization(토큰화)
 2. 문장 부호를 기준으로 떼어 냄

Input text : I loved you. Data-mining



Output text : "I", "loved", "you", ".", "data", "-", "mining"



Data Preprocessing task 1

- Tokenization(토큰화)
 - 3. 문장 부호를 떼어 내는데 예외를 둠

Input text : I loved you. Data-mining



Output text : "I", "loved", "you", ".", "data-mining"



Data Preprocessing task 1

- Tokenization(토큰화)

Input text : 한국어를 처리하는 예시입니다 ㅋㅋㅋ



Output text : " 한국어 ", " 를 ", " 처리", " 하는", " 예시", " 입", "
니다“, “ㅋㅋㅋ”



Data Preprocessing task 1

- Tokenization(토큰화) 고려할 사항
 - 구두점이나 특수문자를 단순 제외해서는 안됨
 - 줄임말과 단어 내에 띄어쓰기가 있는 경우
 - She's -> "she", "'", "s"
 - Don't -> "Do", "n't"
 - Data-mining -> "Data-mining"



Data Preprocessing task 2

- Normalization(정규화)
 - 표현 방법이 다른 단어들을 통합시켜서 같은 단어로 만듦
 - HTML 문서로부터 가져온 corpus라면 문서 내에 있는 HTML 태그 제거
 - 뉴스 기사의 경우 게재 시간 제거, 기자 이름 제거



Tokenization 실습



Data Preprocessing task 2

- Normalization(정규화)
 - 규칙기반을 통해 단어들을 통합시켜서 같은 단어로 만듦
 - HTML 문서로부터 가져온 corpus라면 문서 내에 있는 HTML 태그 제거
 - 뉴스 기사의 경우 게재 시간 제거, 기자 이름 제거



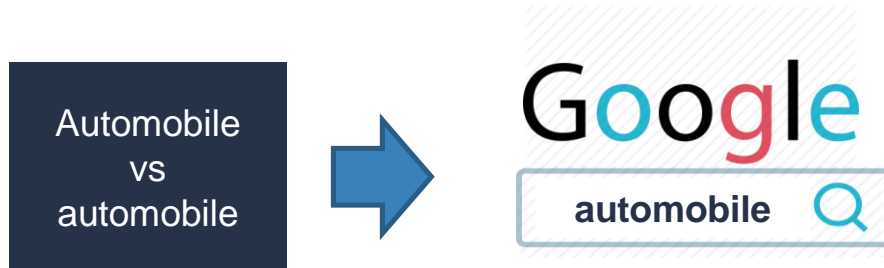
Data Preprocessing task 3

- Cleaning (정제)
 - 갖고있는 corpus로부터 noisy를 제거
 - 대, 소문자 통합
 - 등장 빈도 적은 단어 제거
 - 길이가 짧은 단어 제거



Data Preprocessing task 3

- Cleaning (정제)
 - 갖고있는 corpus로부터 noisy를 제거
 - 대, 소문자 통합





Data Preprocessing task 3

- Cleaning (정제)
 - 갖고있는 corpus로부터 noisy를 제거
 - 대, 소문자 통합

회사 이름
사람 이름
US vs us





Data Preprocessing task 3

- Cleaning (정제)
 - 갖고있는 corpus로부터 noisy를 제거
 - 등장 빈도 적은 단어 제거



Data Preprocessing task 3

- Cleaning (정제)
 - 갖고있는 corpus로부터 noisy를 제거
 - 길이가 짧은 단어 제거
 - a, it, at, to, on, in, by 등

학교
School

용
dragon

이름
name

강아지
dog



Cleaning & Normalization실습



Data Preprocessing task 4,5

- 눈으로 봤을 때는 서로 다른 단어들이지만, 하나의 단어로 일반화 시켜 문서 내의 단어 수를 줄임
 - Lemmatization (표제어 추출), Stemming(어간 추출)



Data Preprocessing task 4

- Lemmatization (표제어 추출)
 - 품사 정보가 보존된 형태의 기본형으로 변환
 - 표제어 추출에 가장 섬세한 방법은 => 형태학적 파싱
 - 형태소란?
 - 의미를 가진 가장 작은 단위
 - 어간(stem) : 단어의 의미를 담고 있는 단어의 핵심 부분
 - 접사(affix) : 단어에 추가적인 의미를 주는 부분

Cats → cat(어간) + s(접사)

Dies → die

Watched → watch

Has → have



Lemmatization 실습



Data Preprocessing task 5

- Stemming (어간 추출)
 - 대표적 포터 알고리즘의 어간 추출은 이러한 규칙들을 가짐
 - ALIZE → AL
 - ANCE → 제거
 - ICAL → IC

formalize → formal
allowance → allow
electrical → electric
먹었다(ate), 먹을(will eat) → 먹다(eat)



Stemming 실습



Data Preprocessing task 4,5

- Lemmatization(표제어 추출)
- Stemming (어간 추출)

Lemmatization

am → be
the going → the going
having → have

Stemming

am → am
the going → the go
having → hav
입니다 → 이다



Data Preprocessing task 6

- Stopword (불용어)
 - 갖고 있는 데이터에서 유의미한 단어 토큰만을 선별하기 위해서는 큰 의미가 없는 단어 토큰을 제거하는 작업이 필요
 - I, my, me, over, 조사, 접미사 같은 단어들은 문장에서는 자주 등장하지만 실제 의미 분석을 하는데는 거의 기여하는 바가 없음
 - [한국어] 조사, 접미사 : 나, 너, 은, 는, 이, 가, 하다, 합니다 등

Family is not an important thing. It's everything.



['Family', 'important', 'thing', '.', 'It', "'s", 'everything', '.']



Stopwords 실습



Data Preprocessing task 7

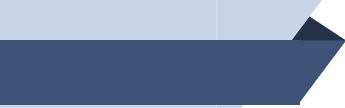
- Regular Expression (정규 표현식)
 - 특정 규칙을 정하여 텍스트 데이터를 정리함

모듈 함수	설명
<code>re.compile()</code>	정규표현식을 컴파일하는 함수입니다. 다시 말해, 파이썬에게 전해주는 역할을 합니다. 찾고자 하는 패턴이 빈번한 경우에는 미리 컴파일해놓고 사용하면 속도와 편의성면에서 유리합니다.
<code>re.search()</code>	문자열 전체에 대해서 정규표현식과 매치되는지를 검색합니다.
<code>re.match()</code>	문자열의 처음이 정규표현식과 매치되는지를 검색합니다.
<code>re.split()</code>	정규 표현식을 기준으로 문자열을 분리하여 리스트로 리턴합니다.
<code>re.findall()</code>	문자열에서 정규 표현식과 매치되는 모든 경우의 문자열을 찾아서 리스트로 리턴합니다. 만약, 매치되는 문자열이 없다면 빈 리스트가 리턴됩니다.
<code>re.finditer()</code>	문자열에서 정규 표현식과 매치되는 모든 경우의 문자열에 대한 이터레이터 객체를 리턴합니다.
<code>re.sub()</code>	문자열에서 정규 표현식과 일치하는 부분에 대해서 다른 문자열로 대체합니다.



정규식 표현식 실습

응용 실습



HTML 크롤링 후 단어 빈도수 구하기



명사만 추출해보기



THANKS!

허윤아

yj72722@korea.ac.kr

박찬준

bcj1210@naver.com