02 텍스트 전처리
(Text Processing)

2019160025 수학과 원윤정

데이터 전처리 과정

토큰화

주어진 코퍼스(corpus)에서
토큰(token)이라 불리는 단위로
나누는 작업

정제

갖고 있는 코퍼스로부터
노이즈 데이터를 제거하는 작업

정규화

표현 방법이 다른 언어들을
통합시켜서 같은 단어로 만드는
과정

단어 토큰화 : 토큰의 단위가 단어

Don't be fooled by the dark sounding name, Mr. **Jone's** Orphanage is as cheery as cheery goes for a pastry shop.

- NLTK에서 word-tokenize 사용

  Don’t 를 Don + ’t 로
  Jone’s 를 Jone + ‘s 로 분리함

- NLTK에서 wordPunctTokenizer 사용

  Don’t 를 Don’t 로
  Jone’s 를 Jone’s 로 분리함

◆고려사항

- 구두점이나 특수 문자를 단순 제외하면 안됨
- 줄임말이나 단어 내에 띄어쓰기가 있는 경우

문장 토큰화 : 토큰의 단위가 문장

- “?”, “!” 는 꽤 명확한 구분자이지만 “.” (온점)은 아님
- NLTK의 sent_tokenize는 영어 문장의 토큰화를 수행하는데, 단순히 온점을 구분자로 설정하지 않아서 “Ph.D.”를 문장 내의 단어로 인식함

이진분류기(Binary Classifier)

: 예외사항을 발생시키는 온점 처리를 위해 입력에 따라 두 개의 클래스로 분류

- 온점이 약어로 쓰이는 경우 (←약어 사전이 사용됨)
- 온점이 문장의 구분자일 경우

한국어 토큰화

: 한국어는 조사, 어미를 붙여서 말을 만드는 교착어이기 때문에 토큰화가 어려움

품사 태깅 (Part-of-speech tagging)

- 단어의 의미를 제대로 파악하기 위해선 어떤 품사로 쓰였는지 알아야 함
  - NLTK 는 영어 지원
  - KoNLPy(코엔엘파이) 는 한국어 지원

Penn Treebank POG Tag

PRP	인칭대명사	NNP	고유명사
VBP	동사	NNS	복수형명사
RB	부사	CC	접속사
VBG	현재부사	DT	관사
IN	전치사		

한국어 nlp에서 형태소 분석기 사용은 단어 토큰화가 아니라 형태소 단위로 형태소 토큰화를 수행함

정제(Cleaning)와 정규화(Normalization)

: 함께 진행되는 경우가 많음

- 표기가 다른 같은 의미의 단어를 하나의 단어로 정규화
  → **어간 추출과 표제어 추출**

- 대소문자 통합 (주로 소문자로)

- 노이즈 데이터(불필요한 단어) 제거
  →빈도가 적은 단어, **불용어**

- 노이즈데이터의 특징을 잡아낼 수 있다면 **정규 표현식**을 통해서 한번에 제거 가능

어간 추출과 표제어 추출

: 정규화 기법 중 단어의 개수를 줄이는 방법

➢ 표제어 추출 : 뿌리 단어 찾기

- 먼저 단어의 형태학적 파싱(어간과 접사 분리)를 진행
- NLTK의 WordNetLemmatizer로 표제어 추출

✓ 문맥을 고려하고 해당 단어의 품사를 보존함

➢어간 추출(Stemming)

- NLTK의 PorterStemmer로 어간 추출
- 단순 규칙에 기반하여 이루어지기 때문에 섬세한 작업이 아니고, 사전에 없는 단어일 수도 있음

✓ 품사 정보를 보존하지 않고 사전에 존재하지 않는 단어일 수도 있음

불용어(stopword)

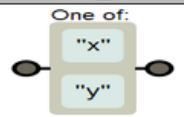
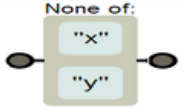
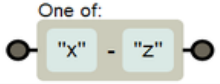
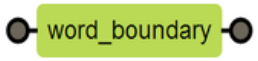
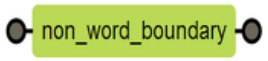
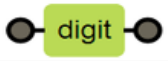
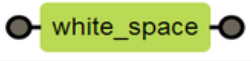
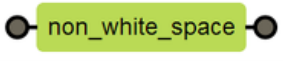
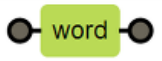
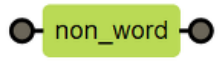
: 유의미한 단어 토큰만 선별하기 위해서 큰 의미없는 단어 토큰은 제거함

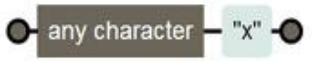
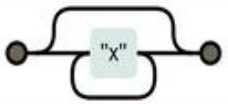
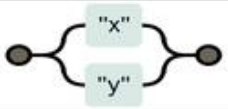
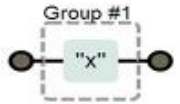
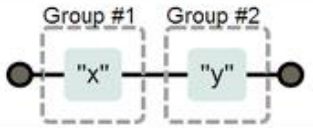
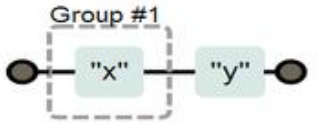
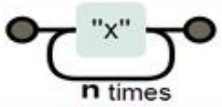
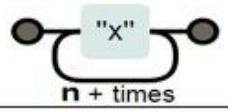
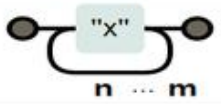
1. NLKT에서 불용어 확인하기 (NLTK의 stopwords 사용)
2. NLKT로 불용어 제거하기

◆ 한국어 불용어 제거하기

- 간단하게는 토큰화 후에 조사, 접속사 등을 제거하는 방법이 있음
- 조사, 접속사 뿐만 아니라 명사, 형용사 등 중에서 불용어로 제거하고 싶을 때는 사용자가 직접 불용어 사전을 만들어서 제거할 수 있음

정규 표현식(Regular Expression)

정규표현식	표현	설명
[xy]		x,y중 하나를 찾습니다.
[^xy]		x,y를 제외하고 문자 하나를 찾습니다. (문자 클래스 내의 ^는 not을 의미합니다.)
[x-z]		x~z 사이의 문자중 하나를 찾습니다.
₩^		^(특수문자)를 식에 문자 자체로 포함합니다. (escape)
₩b		문자와 공백사이의 문자를 찾습니다.
₩B		문자와 공백사이가 아닌 값을 찾습니다.
₩d		숫자를 찾습니다.
₩D		숫자가 아닌 값을 찾습니다.
₩s		공백문자를 찾습니다.
₩S		공백이 아닌 문자를 찾습니다.
₩t		Tab 문자를 찾습니다.
₩v		Vertical Tab 문자를 찾습니다.
₩w		알파벳 + 숫자 + _ 를 찾습니다.
₩W		알파벳 + 숫자 + _ 을 제외한 모든 문자를 찾습니다.

정규표현식	표현	설명
^x		문자열이 x로 시작합니다.
x$		문자열이 x로 끝납니다.
.x		임의의 한 문자를 표현합니다. (x가 마지막으로 끝납니다.)
x+		x가 1번이상 반복합니다.
x?		x가 존재하거나 존재하지 않습니다.
x*		x가 0번이상 반복합니다.
x|y		x 또는 y를 찾습니다. (or연산자를 의미합니다.)
(x)		( )안의 내용을 캡쳐하며, 그룹화 합니다.
(x)(y)		그룹화 할 때, 자동으로 앞에서부터 1번부터 그룹 번호를 부여해서 캡쳐합니다. 결과값에 그룹화한 Data가 배열 형식으로 그룹번호 순서대로 들어갑니다.
(x)(?:y)		캡쳐하지 않는 그룹을 생성할 경우 ?:를 사용합니다. 결과값 배열에 캡쳐하지 않는 그룹은 들어가지 않습니다.
x{n}		x를 n번 반복한 문자를 찾습니다.
x{n,}		x를 n번이상 반복한 문자를 찾습니다.
x{n,m}		x를 n번이상 m번이하 반복한 문자를 찾습니다.

정수 인코딩(Integer Encoding)

: 각 단어를 고유한 정수에 맵핑시키는 전처리 작업

➢ Dictionary 사용

- 모든 단어를 소문자로 바꾸고 불용어와 길이가 짧은 단어를 제거한 뒤 저장
- 빈도가 높은 순서대로 정렬
- 높은 빈도수를 가진 단어일수록 낮은 정수 인덱스 부여

➢ Counter 사용

- 하나의 리스트로 묶어서 저장한 뒤, counter()의 입력으로 한번에 중복된 단어는 제거하고 단어의 빈도수를 도출함
- 높은 빈도수를 가진 단어일수록 낮은 정수 인덱스 부여

➢NLKT의 FreqDist 사용

정수 인코딩(Integer Encoding)

: 각 단어를 고유한 정수에 맵핑시키는 전처리 작업

➢ 케라스의 텍스트 전처리

- fit_on_texts : 입력한 텍스트로부터 단어 빈도수가 높은 순으로 낮은 정수 인덱스가 부여됨
- word.counts : 각 단어가 카운트를 수행하였을 때 몇 개였는지 확인
- num_words = (지정한 상위 단어 개수) **+1** : 0부터 카운팅하기 때문 (**패딩**)

패딩(padding)

: 컴퓨터가 병렬 연산을 위해서 여러 문장의 길이를 임의로 동일하게 맞춰주는 작업

- 'PAD'라는 가상의 단어를 "0" 이라고 하고, 모든 문장의 길이를 가장 긴 문장의 길이로 맞추기 위해 "0"을 채워 넣어 길이를 동일하게 맞춤 (제로패딩)
- 기준 길이를 짧은 길이의 문장으로 하면 데이터는 손실됨
- 케라스에서는 pad_sequences() 제공

원-핫 인코딩(One-Hot Encoding)

: 문자를 숫자로 바꾸는 기법들 중 하나

◆ 단어집합

- 텍스트의 **서로 다른** 모든 단어를 중복을 허용하지 않고 모아 놓은 것

  →book과 books와 같이 단어의 변형 형태도 다른 단어로 간주함

원-핫 인코딩(One-Hot Encoding)

: 문자를 숫자로 바꾸는 기법들 중 하나

◆원-핫 인코딩

1. 단어 집합 만들기
2. 각 단어에 고유한 인덱스 부여 (정수 인코딩)
3. 표현하고 싶은 단어의 인덱스 위치에 1을 부여하고, 다른 단어의 인덱스 위치에는 0을 부여

- 케라스를 이용한 원-핫 인코딩: to_categorical() 이용

✓ **한계**

- 단어 집합의 크기를 벡터 차원으로 하기 때문에, 저장 공간 측면에서 비효율적임
- 단어간 유사성 측정이 안되기 때문에, 이를 해결하기 위해 단어의 잠재 의미를 반영하여 다차원 공간에 벡터화하는 기법들을 사용함

한국어 전처리 패키지

: 형태소와 문장 토크나이징 도구들과 함께 유용하게 사용할 수 있음

➢ **PyKoSpacing**

- 한국어 띄어쓰기 패키지로 띄어쓰기가 되어 있는 않은 문장을 띄어쓰기를 한 문장으로 변환해주는 패키지

➢ **Py-Hanspell**

- 네이버 한글 맞춤법 검사기를 바탕으로 만들어진 패키지
- 맞춤법을 고쳐주고 띄어쓰기도 보정해줌

➢ **Soynlp**

- 품사 태깅, 단어 토큰화 등을 지원하는 단어 토크나이저