과학기술정보통신부
Ministry of Science and ICT

정보통신기획평가원

medici

AiDA Lab.
Artificial Intelligence & Data Analysis Laboratory

2021년 혁신성장 청년인재 집중양성 추경 사업

빅데이터 분야

자연어 처리

Tokenizing (형태소 분석)

Seokhwan Yang

Tokenizing(형태소 분석) 개요

- 자연어: 우리가 일상 생활에서 사용하는 언어
- 기본적으로 컴퓨터는 자연어를 이해하지 못한다
- 컴퓨터에게 자연어를 이해하게 하려면?
  - 여러 방법이 연구되어 왔으나 가장 일반적인 방법은?
    - 토크나이징과 임베딩 기반으로 컴퓨터가 이해할 수 있도록 데이터화
    - 텍스트 유사도를 이용하여 문맥 분류

Tokenizing(형태소 분석) 개요

- **Tokenizing이란?**
  - **주어진 문장에서 토큰 단위로 정보를 나누는 작업**
  - **문장 형태의 데이터를 처리하기 위해 제일 처음 수행해야 하는 기본적인 작업**
  - **주로 텍스트 전처리 과정에서 사용됨**
  - **토큰: 일정한 의미가 있는 가장 작은 정보 단위**
- **Tokenizing 과정**
  - 어떤 문장을 일정한 의미가 있는 가장 작은 단어로 나눈다
  - 나눠진 단어들을 이용해 의미를 분석한다 (이때 가장 기본이 되는 단어가 토큰이다)

토큰화(Tokenizing)의 방식

- 토큰화(Tokenizing)의 기본적인 방식
  - 단어 단위 토큰화 : 단어(어절) 단위로 토큰화
  - 문자 단위 토큰화 : 문자 단위로 토큰화
  - 서브 워드 단위 토큰화 : 서브 워드 단위로 토큰화

토큰화(Tokenizing)의 방식

- **단어 단위 토큰화**
  - **가장 쉬운 방법은 공백으로 분리(별도의 토크나이저가 없어도 무방)**
    - 단점: 어휘 집합의 크기가 매우 커질 수 있음
      - "갔었어", "갔었는데요"는 서로 다른 토큰이 됨
      - 표현이 살짝만 바뀌어도 관련된 모든 경우의 수가 어휘 집합에 포함되어야 함

토큰화(Tokenizing)의 방식

- **사전 학습된 토크나이저를 사용한다면**
  - 어휘 집합의 비대화를 다소 완화 가능 (완전히 해결하기는 어려움)

  - 하나의 언어로 모델을 구축할 때, 어휘 집합의 크기는 10만개를 가뿐히 넘어감
  - **어휘 집합의 크기가 커질 수록 모델의 학습은 어려워짐**

토큰화(Tokenizing)의 방식

- **문자 단위 토큰화**
  - **한글의 경우 표현 가능한 글자는 11,172개**
    - 알파벳, 숫자, 기호를 모두 고려해도 어휘 집합의 크기는 15,000개 정도
  - **해당 언어의 모든 문자를 어휘 집합에 포함 → 미등록 토큰 문제 없음**
  - **단점**
    - 각 토큰은 의미 있는 단위가 될 수 없음
    - 어미에 따른 변화, 조사의 사용 등 한글의 특징이 모두 사라짐
    - 분석 결과인 토큰 시퀀스의 길이가 단어 단위 토큰화의 결과보다 상대적으로 길어짐
    - 언어 모델에 입력할 토큰 시퀀스가 길면 모델의 학습이 어려워지고 결과적으로 성능하락

토큰화(Tokenizing)의 방식

- **서브 워드 단위 토큰화**
  - **단어 단위 토큰화와 문자 단위 토큰화의 중간 형태**
  - **두 토큰화 방식의 장점만 적용**
    - 어휘 집합의 크기가 지나치게 커지지 않음
    - 미등록 토큰 문제 회피 가능
    - 분석 결과의 토큰 시퀀스가 너무 길어지지 않음
  - **대표적인 서브워드 단위 토큰화 기법: 바이트 페어 인코딩(BPE)**

BPE (Byte Pair Encoding)

- **BPE (Byte Pair Encoding)**
  - **1994년 제안된 정보 압축 알고리즘**
  - **데이터에서 가장 많이 등장한 문자열을 병합하여 데이터를 압축하는 기법**
  - **데이터에 등장한 글자를 초기 사전으로 구성하여 연속된 두 글자를 한 글자로 병합하는 방식 적용**
  - **최근에는 자연어 처리 모델에 널리 쓰이는 토큰화 기법**
    - 대표적인 활용 모델: GPT 모델

BPE (Byte Pair Encoding)

- **BPE 알고리즘의 적용 예**
  - **초기 사전: (a, b, c, d) → 4개, 문자열: aaabdaaabac → 11자**
    - aaabdaaabac → aa를 Z로 병합 → ZabdZabac
    - ZabdZabac → ab를 Y로 병합 → ZYdZYac
    - ZYdZYac → ZY를 X로 병합 → XdXac
  - **BPE 수행 이후**
    - 사전: (a, b, c, d, Z, Y, X) → 7개
    - 결과 문자열: XdXac → 5자

BPE기반 토큰화 기법은 사전 크기의 증가를 억제하면서도 정보를 효율적으로 압축할 수 있는 알고리즘이다.

BPE 어휘 집합은 고빈도 바이그램 쌍을 병합하는 방식으로 구축한다.

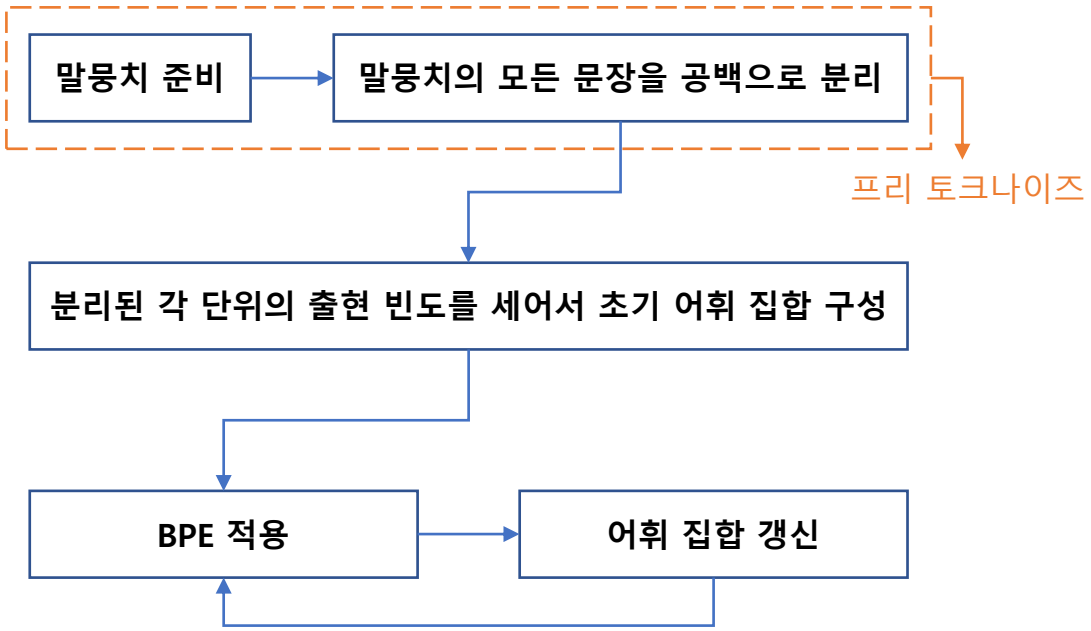
BPE (Byte Pair Encoding)

- BPE 알고리즘의 특징
  - 분석 대상 언어에 대한 지식이 필요하지 않음
  - 말뭉치(코퍼스)에서 자주 나타나는 문자열(서브 워드)을 토큰으로 분석
  - 자연어 처리에서 BPE가 처음 적용된 분야는 기계번역 분야

- BPE 활용 토큰화 절차
  1. 어휘 집합 구축: 자주 등장하는 문자열 병합 후 어휘집합 추가 반복
  2. 토큰화: 문장의 각 어절에서 어휘 집합에 있는 서브 워드를 어절에서 분리

BPE (Byte Pair Encoding)

- **BPE 어휘 집합 구축하기**



어휘 집합이 사용자가 정한 크기가 될 때까지 반복

초기 어휘 집합

b, g, h, n, p, s, u

프리 토크나이즈 결과

토큰	빈도
hug	10
pug	5
pun	12
bun	4
hugs	5

초기 어휘 집합으로 다시 작성한 빈도표

토큰	빈도
h, u, g	10
p, u, g	5
p, u, n	12
b, u, n	4
h, u, g, s	5

바이그램 쌍으로 나열

토큰	빈도
h, u	10
u, g	10
p, u	5
u, g	5
p, u	12
u, n	12
b, u	4
u, n	4
h, u	5
u, g	5
g, s	5

같은 바이그램 쌍 합치기

토큰	빈도
b, u	4
g, s	5
h, u	15
p, u	17
u, g	20
u, n	16

u, g 병합

토큰	빈도
h, ug	10
p, ug	5
p, u, n	12
b, u, n	4
h, ug, s	5

바이그램 쌍 빈도로 나열

토큰	빈도
b, u	4
h, ug	15
p, u	12
p, ug	5
u, n	16
ug, s	5

u, n 병합

토큰	빈도
h, ug	10
p, ug	5
p, un	12
b, un	4
h, ug, s	5

바이그램 쌍 빈도로 나열

토큰	빈도
b, un	4
h, ug	15
p, ug	5
p, un	12
ug, s	5

BPE 어휘 집합 구축 결과

b, g, h, n, p, s, u, ug, un, hug

BPE (Byte Pair Encoding)

- **BPE 토큰화**
  - **어휘 집합과 병합 우선순위를 기준으로 토큰화 수행**
    - 병합 우선순위

토큰	빈도
b, u	4
g, s	5
h, u	15
p, u	17
u, g	20
u, n	16

토큰	빈도
b, u	4
h, ug	15
p, u	12
p, ug	5
u, n	16
ug, s	5

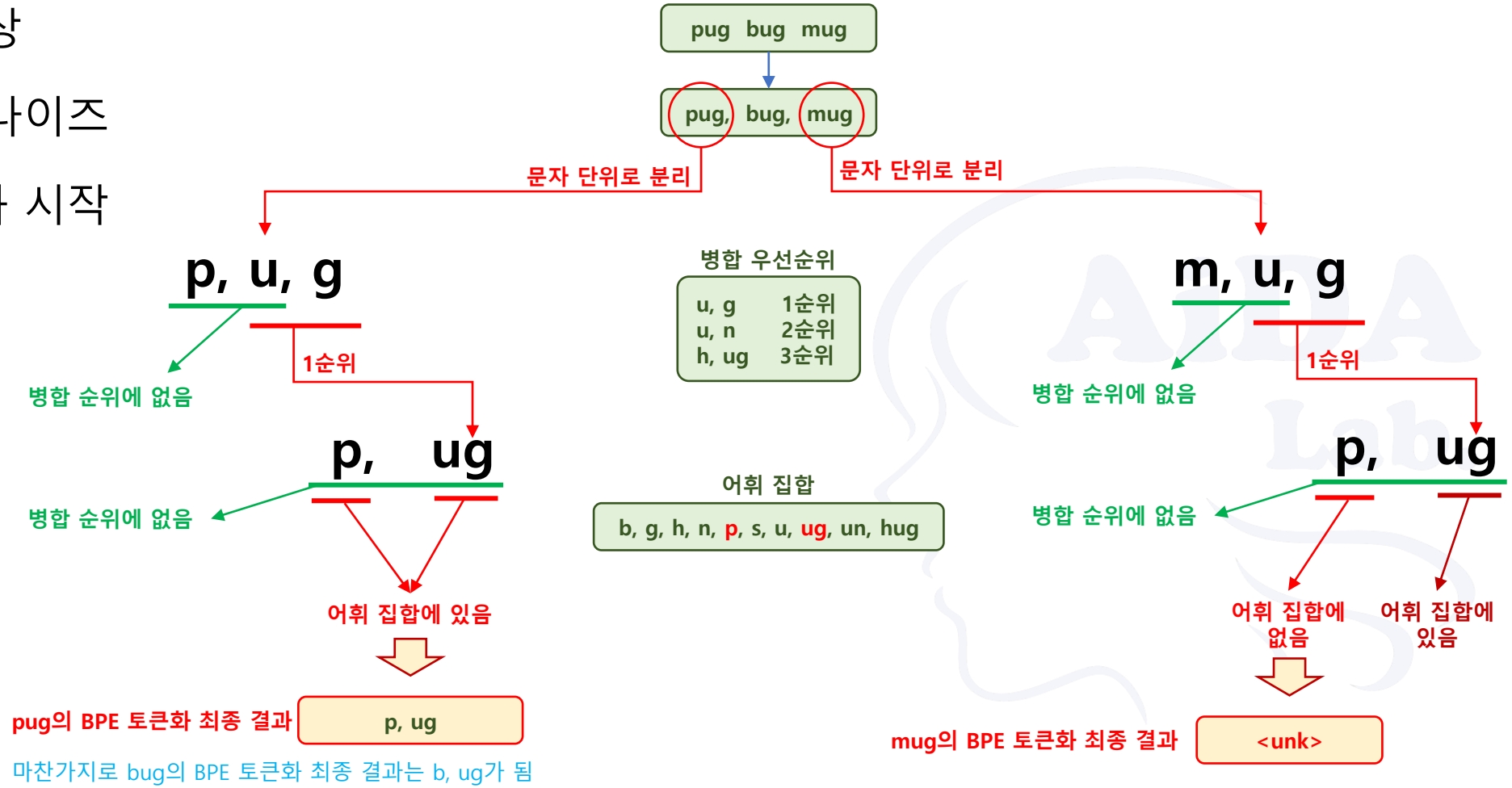
토큰	빈도
b, un	4
h, ug	15
p, ug	5
p, un	12
ug, s	5

병합 우선순위

u, g	1순위
u, n	2순위
h, ug	3순위

BPE (Byte Pair Encoding)

- **BPE 토큰화 예시**
  - 토큰화 대상
  - 프리 토크나이즈
  - BPE 토큰화 시작



BPE (Byte Pair Encoding)

실습

한국어 토크나이징(형태소 분석)

- **한국어 Tokenizing을 구현하려면?**
  - **한국어 문법에 대한 깊은 이해가 필수!!!**
  - **비전공자는 어떻게 해야 하나?**
    - 한국어 Tokenizing을 지원하는 파이썬 라이브러리를 사용한다.
    - 대표적인 한국어 자연어 처리 지원 라이브러리: KoNLPy (코엔엘파이)

한국어 토크나이징(형태소 분석)

- **"토큰 단위를 어떻게 정의하느냐"가 자연어 처리 성능에 큰 영향**
- **한국어 분석에서 사용하는 토큰의 단위는 "형태소"**
  - **형태소란?**
    - 언어학에서 사용되는 용어
    - 일정한 의미가 있는 가장 작은 말의 단위 (의미상 더 이상 쪼개지지 않는 단어)
  - **형태소를 토큰 단위로 사용한다면?**
    - 단어와 품사 정보를 같이 활용할 수 있다 → 효과적 처리가 가능함

한국어 토크나이징(형태소 분석)

- **문장을 어떻게 형태소 단위로 분리(Tokenizing)할 수 있는가?**
  - **영어의 경우는 Tokenizing이 쉽다**
    - 단어의 변화가 크지 않다
    - 띄어쓰기로 단어를 구분한다
    - 따라서, 공백을 기준으로 토크나이징을 수행해도 문제가 없다
  - **한글의 경우는 Tokenizing이 어렵다**
    - 한국어는 명사와 조사를 띄어 쓰지 않는다 → 공백을 기준으로 토크나이징 수행 불가능
    - 용언에 따라 여러 가지 어미가 붙는다 → 일정한 기준을 찾기 어렵다
    - 따라서 복잡한 특성을 고려하여 문장에서 형태소를 분석할 수 있는 "형태소 분석기" 가 필수
    - 다양한 문법적 특징을 반영하고 언어적 속성의 구조를 파악할 수 있어야 함

한국어 토크나이징(형태소 분석)

- **한국어 Tokenizing의 어려움의 예**
  - 아버지가 방에 들어가신다
  - 아버지 가방에 들어가신다

  } 컴퓨터는 이런 것을 구분하지 못함

- **형태소 분석기를 사용해서 분석한 결과**
  - 아버지가 방에 들어가신다.

  → [(‘아버지’, ‘명사’), (‘가’, ‘조사’), (‘방’, ‘명사’), (‘에’, ‘조사’), (‘들어가신다’, ‘동사(서술어)’), (‘.’, ‘마침표(기능적인 기호)’)]

한국어 토크나이징(형태소 분석)

- **한국어의 9품사**

품사	설명
명사	주로 물건이나 사람, 동식물을 가리킬 때 쓰는 품사 [강아지, 철수, 챗봇, ...]
대명사	사람이나 사물의 이름을 대신해서 쓰는 품사 [너, 우리, 무엇, 그것, ...]
수사	숫자나 순서를 나타내는 품사 [하나, 둘, 1, 2, 첫째, 둘째, ...]
동사	동작이나 작용을 나타내는 품사 [먹었다, 보았다, 간다, ...]
형용사	사물의 성질이나 상태를 나타내는 품사 [아름답다, 맵다, 희다, ...]
관형사	체언(명사, 대명사, 수사) 앞에서 체언을 수식하는 품사 [이, 그, 저, 새, 헌, 옛, ...]
부사	동사, 형용사, 동사구, 문장 전체를 수식하는 역할을 맡은 품사 [정말, 벌써, 매우, ...]
조사	명사, 부사 따위에 붙어 문법 관계를 맺어주는 품사 [~이, ~가, ~에서, ...]
감탄사	감탄이나 놀람, 느낌, 응답 등을 나타내는 품사 [꺄, 와, 아하, 헉, ...]

한국어 토크나이징(형태소 분석)

- **형태소 분석기**
  - **복잡한 한국어 문법때문에 형태소 분석기의 개발은 매우 어렵다**
  - **KoNLPy 등의 라이브러리 사용은 필수**
    - KoNLPy 내부에서 다양한 형태소 분석기를 통합하여 라이브러리 형태로 제공
    - 다소 튜닝이 필요하지만 기본적인 성능이 뛰어난 편 → 실무에서도 많이 사용 중
  - **KoNLPy가 제공하는 대표적인 형태소 분석기**
    - Kkma (꼬꼬마, 서울대에서 개발, GPL2 라이센스), Komoran (코모란, Shineware에서 자바로 개발, Apache 2.0 라이센스), Okt (트위터에서 개발, Apache 2.0 라이센스)
    - 통합제공하기 때문에 세 종류 모두 사용법이 거의 동일함

한국어 토크나이징(형태소 분석)

- **KoNLPy의 형태소 분석기 비교**

형태소 분석기	장점	단점
Kkma	• 분석 품질이 좋음 • 지원하는 품사 태그가 가장 많음	• 분석 속도가 느림 • 사용자 사전으로 추가한 복합 명사에 대하여 불완전하게 동작함
Komoran	• 자소가 분리된 문장이나 오탈자에 강함 • 사용자 사전 관리 용이	• 적당한 분석 품질과 분석 속도
Okt	• 매우 빠른 분석 속도 • 정규화 기능 지원	• 사용자 사전 관리가 어려움 • 용언 분석에 일관성이 부족함

한국어 토크나이징(형태소 분석)

- **사용자 사전 구축**
  - **챗봇의 데이터 입력단은 인터넷 구어체와 관련이 많음**

    → 일반적으로 딱딱한 구어체, 문어체 등을 사용하지 않음
  - **새롭게 생겨나는 단어나 문장은 형태소 분석기가 인식하지 못하는 경우 많음**
    - 기존의 많은 문장을 이용하여 형태소 분석기 모델이 개발되었으므로 새로운 형태의 단어, 문장은 학습 데이터에 포함되어 있지 않음 → 인식률 저하의 원인
  - **문제의 해결을 위해 대부분의 형태소 분석기는 사용자 사전을 추가할 수 있도록 구성됨**