자연어처리 Flow

리미애오

강사소개

백혜림
hreeee@yonsei.ac.kr

연세대학교 일반대학원 석사 졸업
신호처리전공 주 연구분야 잡음제거, 음원분리, 음성강화

강의이력

모두의연구소 aiffel 강남 1기 & 싹 1기
영우글로벌러닝 2, 3, 4기 자연어처리 강사
서울시교육청 직업계고 ai 기초 & 심화 강사
기상청 프로젝트 강사
엘리스 알고리즘 강사
그 외 기업강의 다수

㈜비바이노베이션 기술 자문

목차

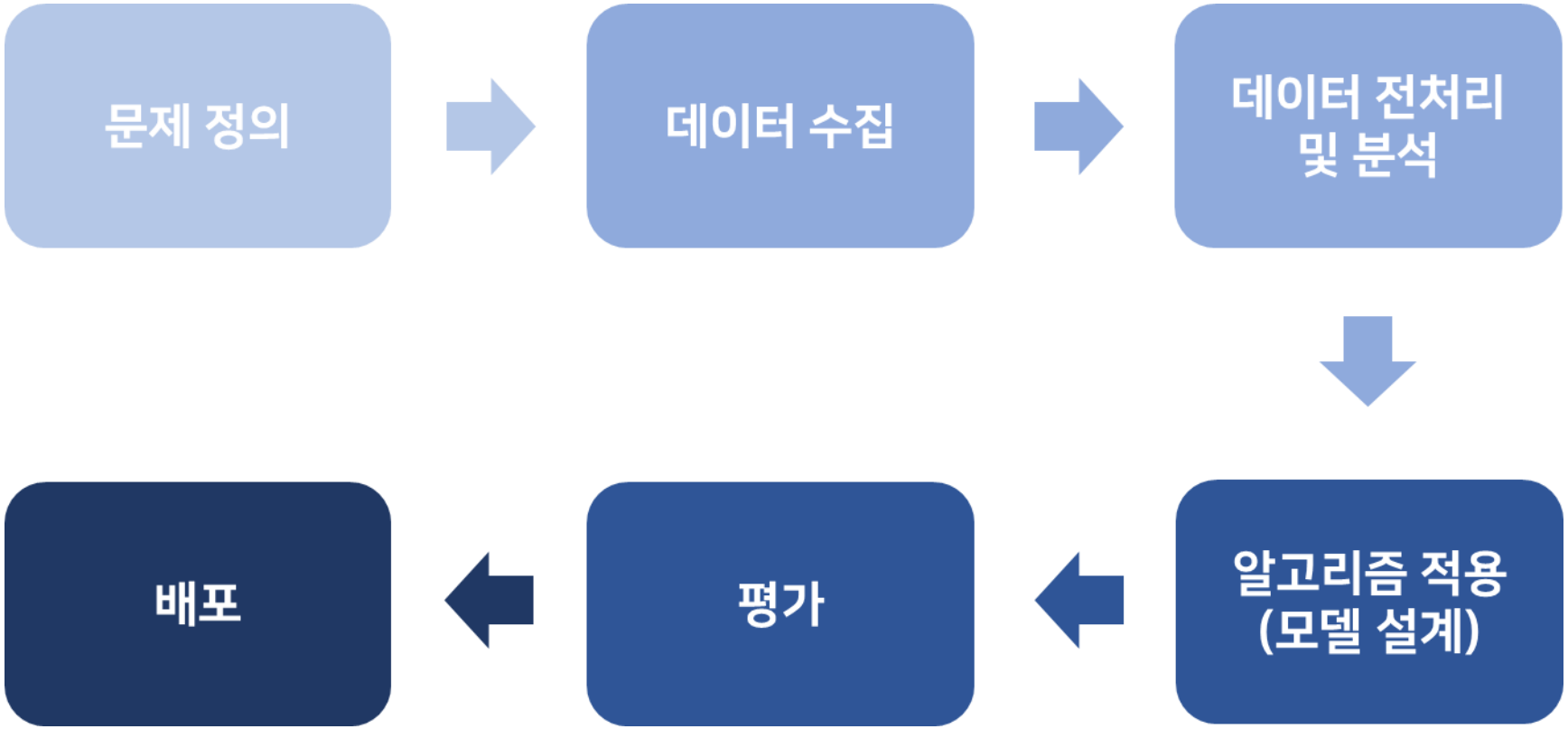
NLP Project Flow

백혜림 hreeee@yonsei.ac.kr

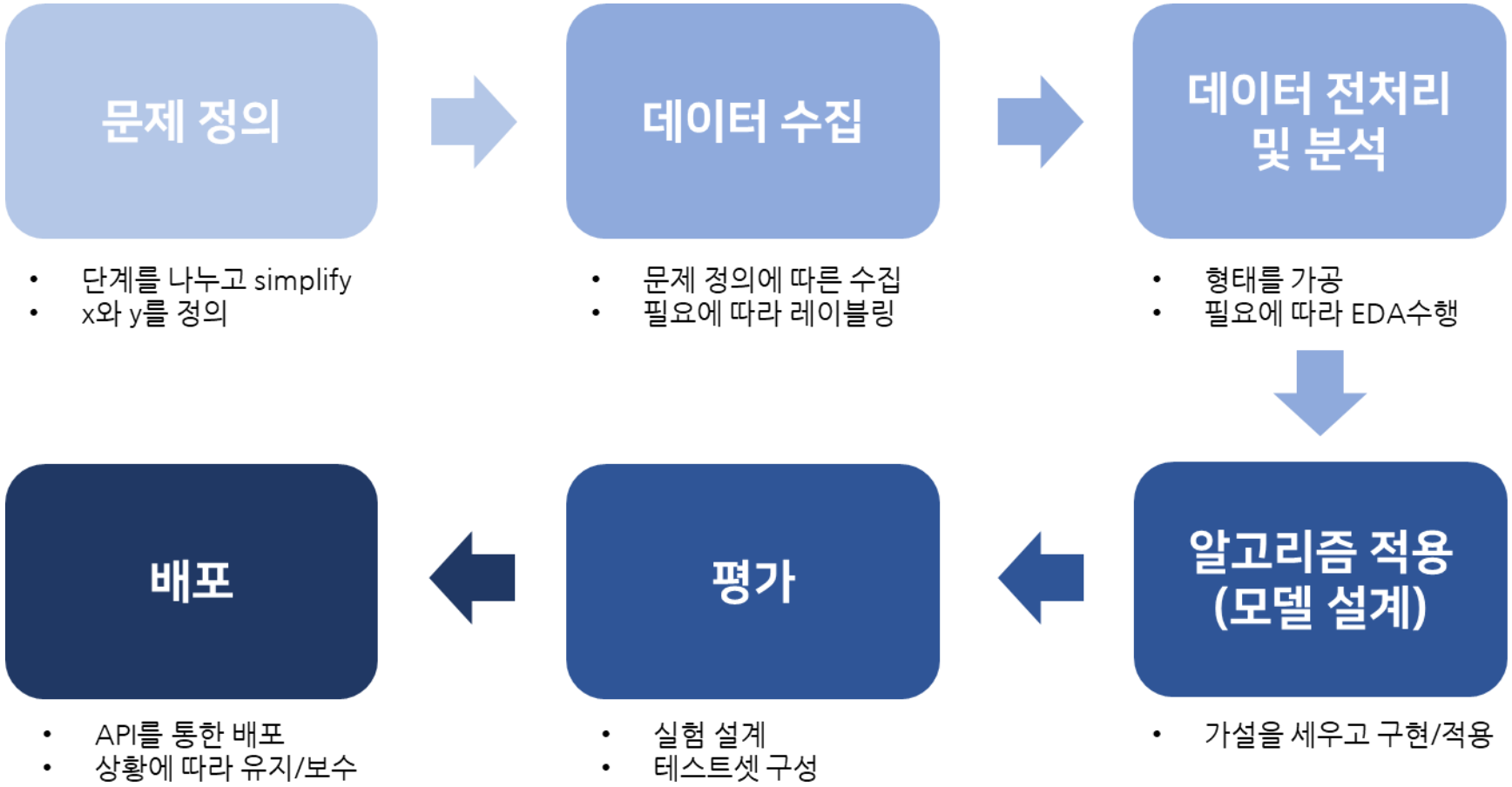
NLP 전처리의 늪

- **전처리의 늪에 오신 것을 환영합니다!**
  - 가장 재미없고, 반복적인 끝이 없는 작업
  - 가장 중요 어쩌면 모델링만큼

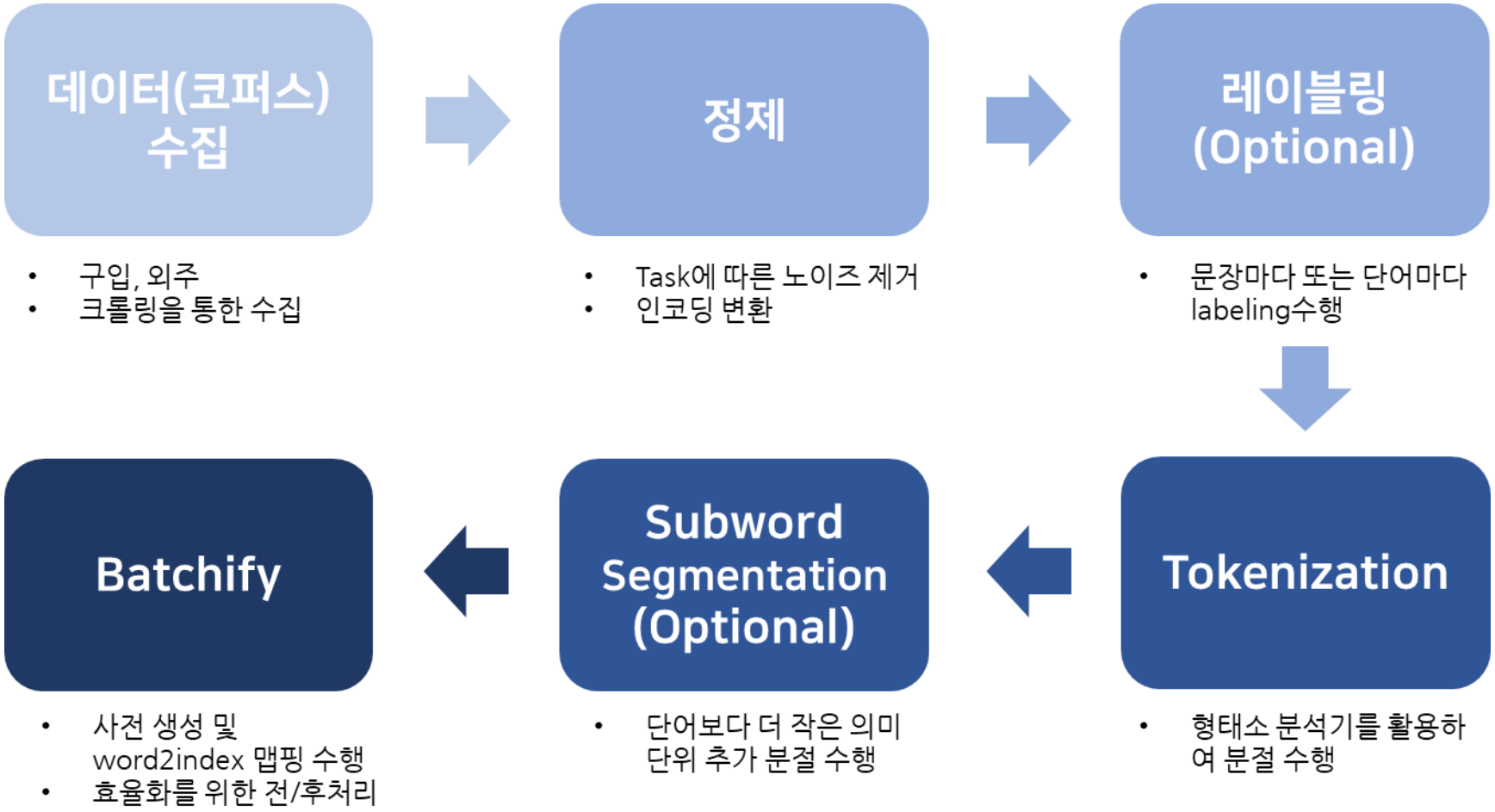
NLP Project Workflow



NLP Project Workflow



전 처리 Workflow



말뭉치 (Corpus)란?

- 자연어처리를 위한 방대한 양의 데이터 모음을 코퍼스라고 한다.
- 언어 분석에 사용되는 실제 언어의 체계적 디지털 모음!

- 둘 이상의 코퍼스가 있으면 Corpora (복수표현)

- 포함된 언어 숫자에 따라
  - Monolingual Corpus (단일 언어 코퍼스)
  - Bi-lingual Corpus (이중 언어 코퍼스)
  - Multilingual Corpus (다국어 코퍼스)

- Parallel Corpus : 대응되는 문장 쌍이 labeling되어 있는 형태

English	Korean
I love to go to school.	나는 학교에 가는 것을 좋아한다.
I am a doctor.	나는 의사 입니다.

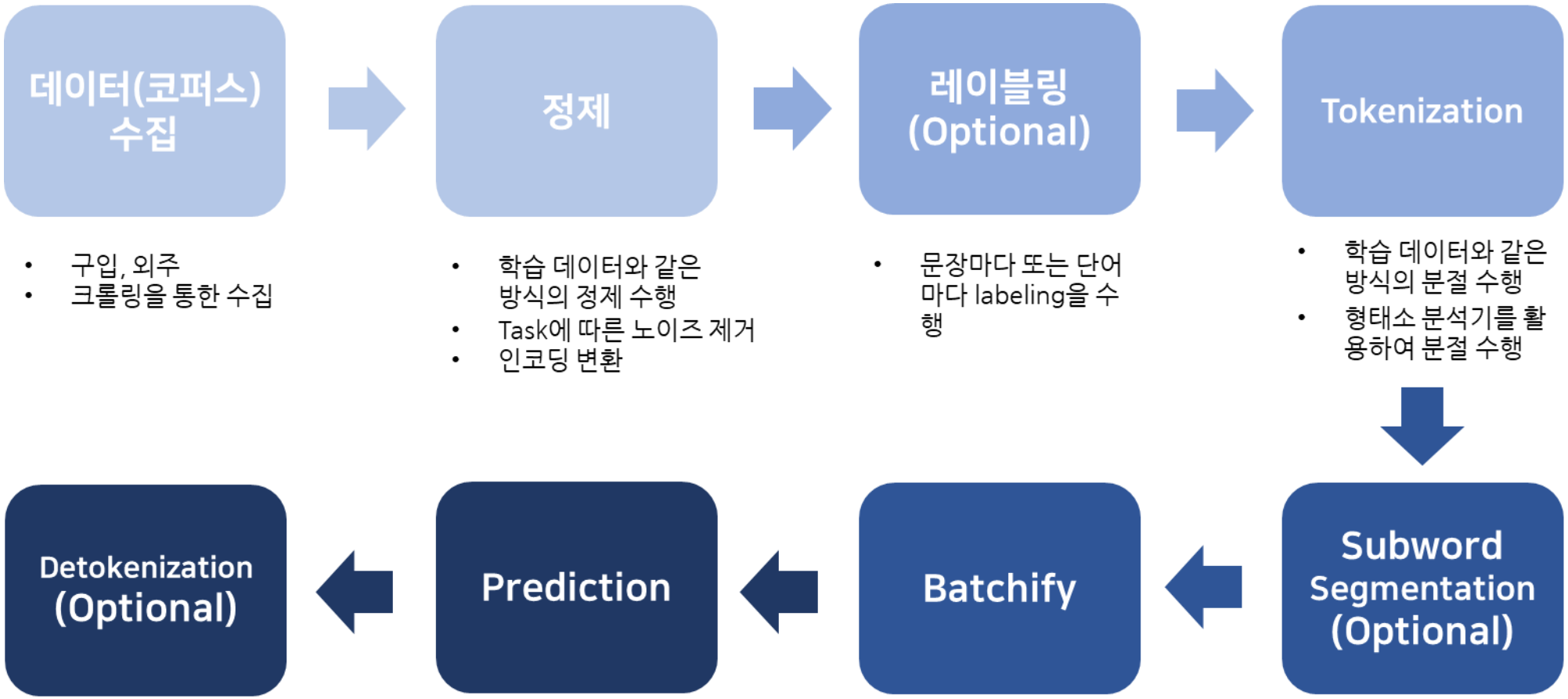
Why we need

- Parallel Corpus?

English	Korean
I love to go to school.	나는 학교에 가는 것을 좋아한다.
I am a doctor.	나는 의사 입니다.

- 주요 수집 대상
  - 뉴스 기사, 드라마/영화 자막
- 대부분의 경우, 문서 단위의 matching은 되어 있지만, 문장 단위는 되어 있지 않음

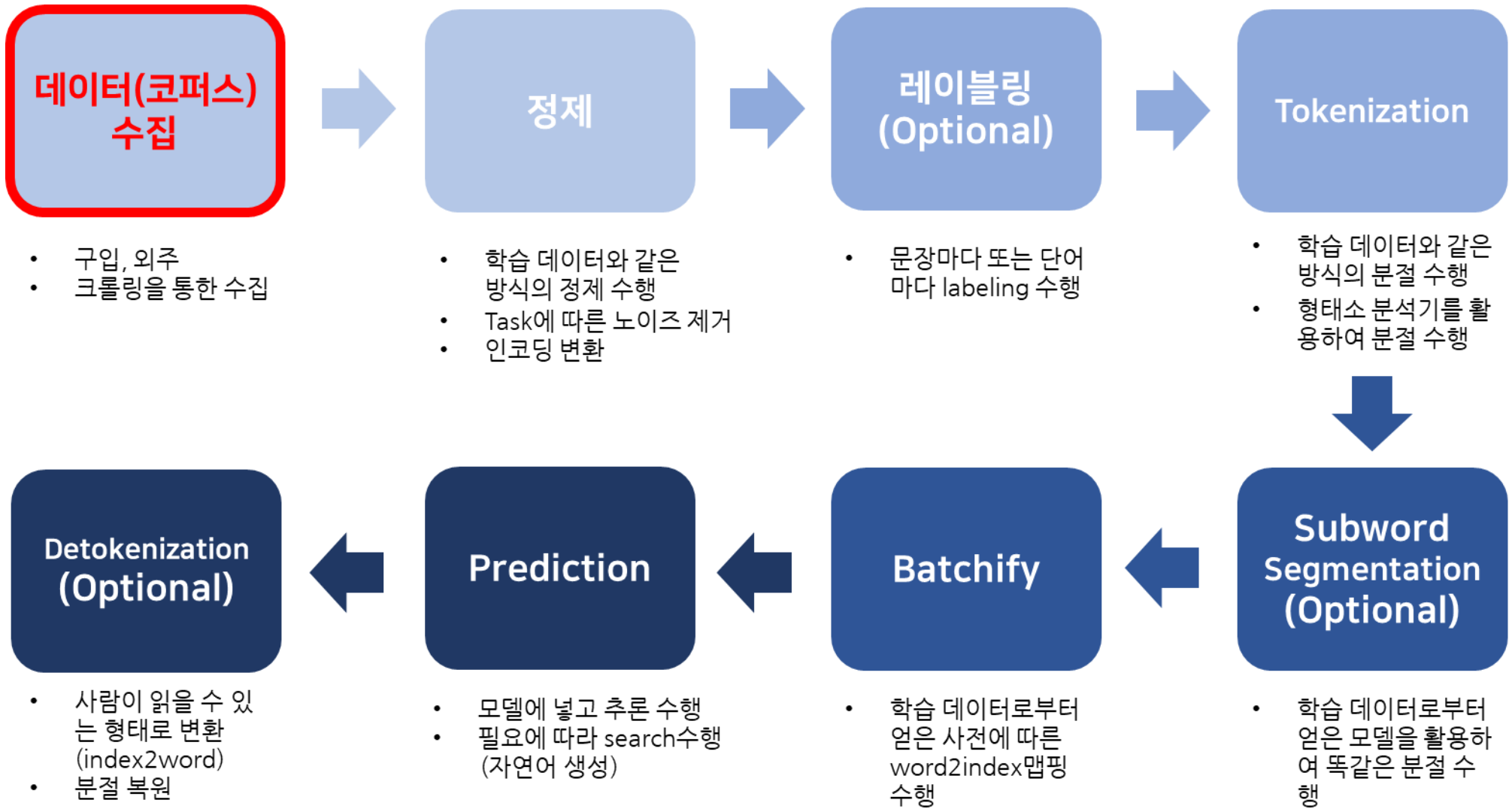
서비스 전체 Pipeline



Data Crawling

백혜림 hreeee@yonsei.ac.kr

서비스 전체 Pipeline



데이터 구입 및 외주의 한계

- 구입
  - 정제 및 레이블링이 완료된 양질의 데이터를 얻을 수 있음
  - 양이 매우 제한적
  - **구입처 :** 대학교, 한국전자통신연구원(ETRI), 데이터제공처

- 외주
  - 수집, 정체 및 레이블링을 외주 줄 수 있음
  - 가장 높은 비용 -> 양이 매우 제한적
  - 품질 관리를 위한 인력이 추가로 필요

최대 10만 단위

무료 공개 데이터

- 공개 사이트
  - AI-HUB
  - WMT competetion
  - Kaggle
  - 데이콘
  - OPUS (https://opus.nlpl.eu/)
- 양이 매우 제한적
- 한국어 코퍼스는 흔치 않다!

최대 10만 단위

Crawling

- 무한한 양의 코퍼스 수집 가능
  - 원하는 도메인 별로 수집 가능

- 하지만 품질이 천차만별이고, 정제 과정에서 많은 노력이 필요
  - 특수문자, 이모티콘, 노이즈, 띄어쓰기

Crawling

- 아직은 회색지대
  하지만 적법한 절차에 따른 크롤링이 필수

저작권이 존재하는 코퍼스로부터 학습한 모델과 그 생성물의 저작권은 누가 갖는가?

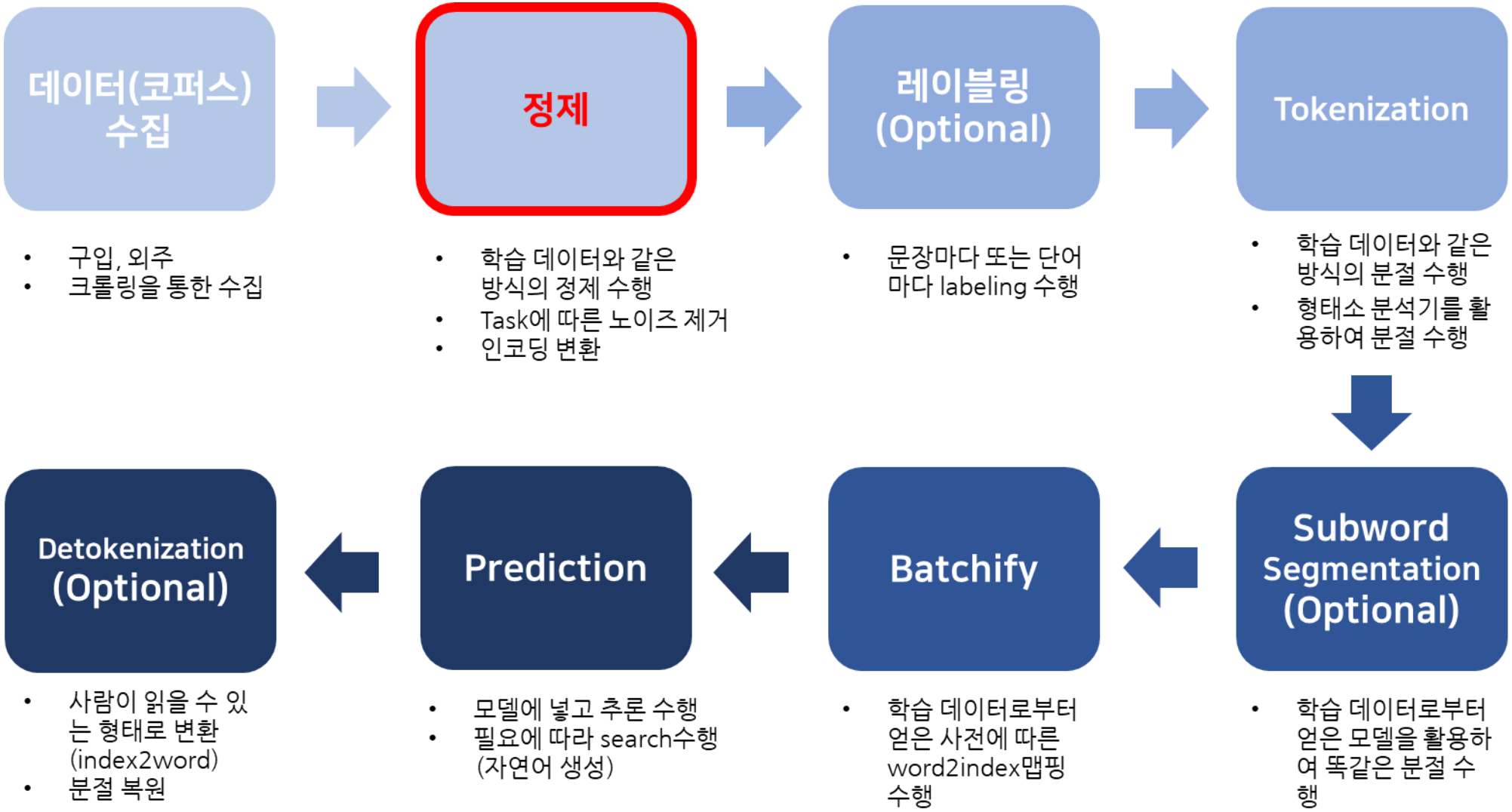
데이터 수집처

수집처	도메인	문체	수집 난이도	양방향	정제 난이도	비고
블로그	일반	대화체	낮음	X	최상	
N* 지식인	다양함	대화체	낮음	X	중간	
뉴스 기사	시사	문어체	낮음	O	낮음	문법 준수
Wikipedia	다양함	문어체	덤프 제공	O	낮음	
나무위키	다양함	문어체	낮음	X	낮음	
커뮤니티	다양함	대화체	중간	X	높음	클리앙 등
TED	다양함	대화체	낮음	O	낮음	
자막	일반	대화체	낮음	O	높음	

Data Cleaning

백혜림 hreeee@yonsei.ac.kr

데이터 수집처


데이터(코퍼스) 수집
구입, 외주
크롤링을 통한 수집
정제
학습 데이터와 같은 방식의 정제 수행
Task에 따른 노이즈 제거
인코딩 변환
레이블링 (Optional)
문장마다 또는 단어마다 labeling 수행
Tokenization
학습 데이터와 같은 방식의 분절 수행
형태소 분석기를 활용하여 분절 수행
Subword Segmentation (Optional)
학습 데이터로부터 얻은 모델을 활용하여 똑같은 분절 수행
Batchify
학습 데이터로부터 얻은 사전에 따른 word2index맵핑 수행
Prediction
모델에 넣고 추론 수행
필요에 따라 search수행 (자연어 생성)
Detokenization (Optional)
사람이 읽을 수 있는 형태로 변환 (index2word)
분절 복원

주의할 점

- Task에 따른 특성
  - 풀고자 하는 문제의 특성에 따라 전처리 전략이 다르다.
  - 신중한 접근이 필요
    - 이모티콘은 필요 없는 정보일까?

- 언어, 도메인, 코퍼스에 따른 특성
  - 각 언어, 도메인, 코퍼스 별 특성이 다르므로 다른 형태의 전처리 전략이 필요

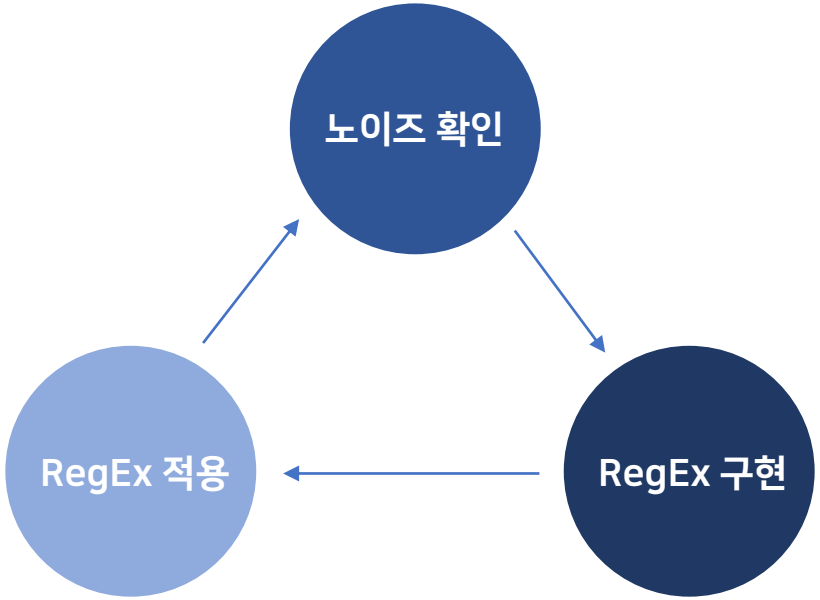
정규식을 활용한 정제

- 정규식(Regular expression)을 활용하면 복잡한 규칙의 노이즈도 제거/치환 가능
- 코딩 없이 단순히 텍스트 에디터(Sublime Text, VSCode등)도 가능

```
>>> # 주민등록번호 형식을 변경
>>> re.sub("-", "@", "901225-1234567")
'901225@1234567'
>>> # 필드 구분자를 통일
>>> re.sub(r"[:,|\s]", ", ", "Apple:Orange Banana|Tomato")
'Apple, Orange, Banana, Tomato'
>>> # 문자열의 변경 횟수를 제한
>>> re.sub(r"[:,|\s]", ", ", "Apple:Orange Banana|Tomato", 2)
'Apple, Orange, Banana|Tomato'
```

Interactive 노이즈 제거 과정

- 규칙에 의해 노이즈를 제거하기 때문에, 노이즈 전부를 제거하는 것은 어려움
- 따라서, 반복적인 규칙 생성 및 적용 과정이 필요
- 끝이 없는 과정
  - 노력과 품질사이의 trade-off
  - Sweet spot을 찾아야함.

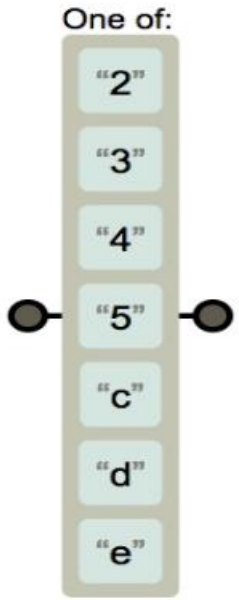


Regular Expression

백혜림 hreeee@yonsei.ac.kr

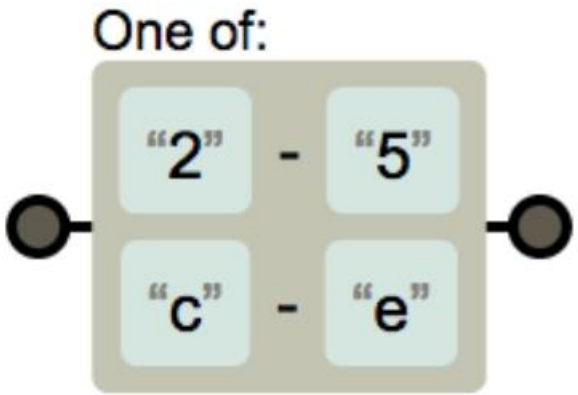
[]

- 2, 3, 4, 5, c, d, e 중의 character
- [2345cde]
- (2|3|4|5|c|d|e)



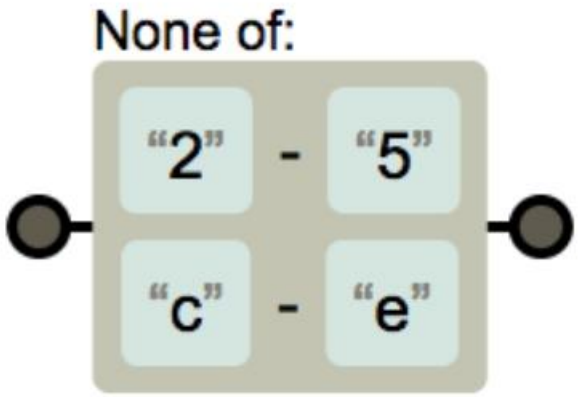
[-]

- 2, 3, 4, 5와 c, d, e 중의 character
- [2-5c-e]



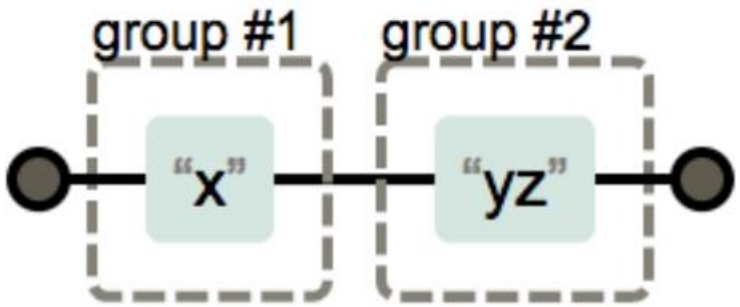
[^]

- 2, 3, 4, 5와 c, d, e를 제외한 모든 character
- [^2-5c-e]



()

- x를 ₩1에 지정, yz를 ₩2에 지정
- (x)(yz)



RegEx의 꿀기능

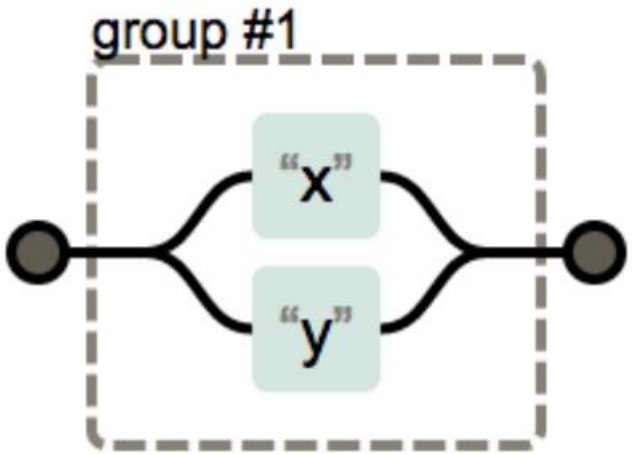
- 양 끝에 알파벳(소문자)으로 둘러싸인 'bc'를 제거하기
  - abcd
  - 0bc1

- 적용 예제

([a-z])bc([a-z]) → ₩1₩2

  - abcd →ad
  - 0bc1 →0bc1

- x 또는 y가 나타남. 그리고 ₩1에 지정
- (x|y)

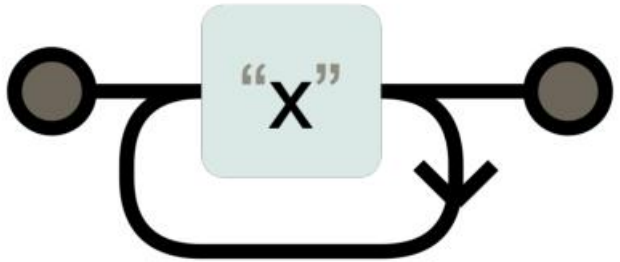


?

- x가 0번 또는 1번 나타남
- x?

+

- x가 한 번 이상 나타남
- x+

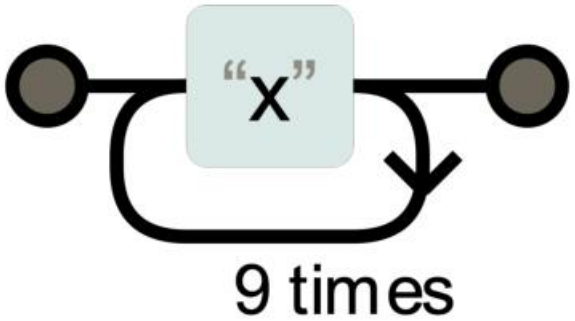


*

- x가 나타나지 않을 수도, 반복될 수도 있음
- 강력한 표현. 유의해서 사용해야 함
- x*

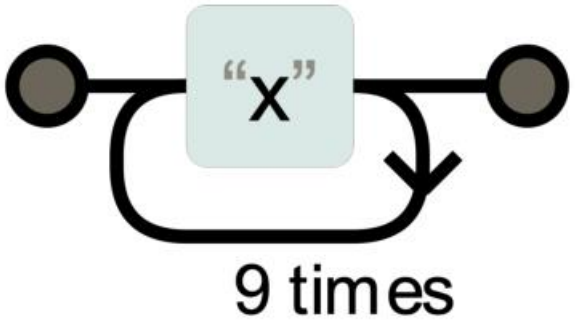
{n}, {n,}, {n,m}

- n번 반복
- x{n}



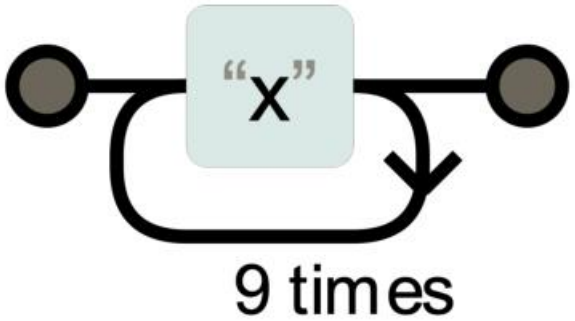
{n}, {n,}, {n,m}

- n번 이상 반복
- x{n,}



{n}, {n,}, {n,m}

- N번부터 m번까지 반복
- x{n,m}

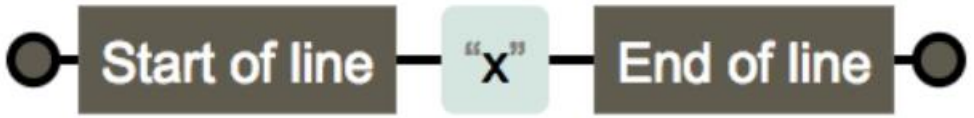


•

- Any character
- 매우 강력한 표현. 유의해서 사용해야 함

^ $

- 문장의 시작과 끝을 표시
- ^x$



그 밖의 지정 문자

Meta Characters	Description
₩s	공백 문자white space
₩S	공백 문자를 제외한 모든 문자
₩w	alphanumeric(알파벳 + 숫자) + '_' ([A-Za-z0-9_]와 같음)
₩W	non-alphanumeric 문자 및 '_' 제외 ([^A-Za-z0-9_]와 같음)
₩d	숫자 ([0-9]와 같음)
₩D	숫자를 제외한 모든 문자 ([^0-9]와 같음)

자연어 노이즈 제거

불완전한 문장으로 구성된 대화의 경우

- 한 문장씩 주고 받는 대화와 달리 메신저는 한 문장을 여러 번에 나눠 전송하거나 여러 문장을 한 번에 전송하는 경우가 있습니다.

예1) A: "아니아니" "오늘이 아니고" "내일이지" / B: "그럼 오늘 사야겠네. 내일 필요하니까?"

문장의 길이가 너무 길거나 짧은 경우

- 아주 짧은 문장은 의미가 없을 수 있고, 대체로 사용빈도가 높은 리액션에 해당하는 경우가 많아서 언어 모델을 왜곡시킬 우려가 있기 때문에 제외해주는 게 좋습니다.

예1) A: "ㅋㅋㅋ", "ㅠㅠㅠ"

아주 긴 문장은 대화와는 관계가 없는 문장일 수 있습니다.

예2) A: "이 편지는 영국에서부터 시작되어…"

채팅 데이터에서 문장 시간 간격이 너무 긴 경우

- 메신저는 누가 자판을 치는지 모르기 때문에 서로의 말이 얽히게 됩니다. 따라서 서로의 말의 텀이 짧으면 그것은 대화가 아니라 서로 할말만 하는 상태일 수 있습니다.

예1) A: "겨울왕국2" / B: "보러가자" / A: "엊그제 봤는데 잼씀" / B: "오늘 저ㄴ 아니 ㅡㅡ"

혹은 말의 텀이 너무 길다면 그것은 연속된 대화로 보기 어렵습니다.

예2) A: "나 10만원만 빌려줄 수 있어?" / …… / B: "아 미안 지금 봤다 아직 필요해?"

바람직 하지 않은 문장의 사용

- 욕설의 비율이나, 오타의 비율이 높은 문장은 자연어 모델 학습에 사용하지 않는 것이 좋습니다.

노이즈 유형 (1) 문장부호 : Hi, my name is john

- 문장 부호 양쪽에 공백을 추가하는 방법을 취하자!

```
1  def pad_punctuation(sentence, punc):
2      for p in punc:
3          sentence = sentence.replace(p, " " + p + " ")
4
5      return sentence
6
7  sentence = "Hi, my name is john."
8
9  print(pad_punctuation(sentence, [".", "?", "!", ","]))
```

노이즈 유형 (2) 대소문자 : First, open the first chapter

- 모든 단어를 소문자로 바꾸는 방법

```
1  sentence = "First, open the first chapter."
2
3  print(sentence.lower())
```

```
1  sentence = "First, open the first chapter."
2
3  print(sentence.upper())
```

노이즈 유형 (3) 특수문자 : He is a ten-year-old boy

- 사용할 알파벳과 기호들을 정의해 이를 제외하고 모두 제거하자!

```
1  import re
2
3  sentence = "He is a ten-year-old boy."
4  sentence = re.sub("([^a-zA-Z.,?!])", " ", sentence)
5
6  print(sentence)
```

배운 것들을 종합하기

- https://www.gutenberg.org/files/2397/2397-h/2397-h.htm

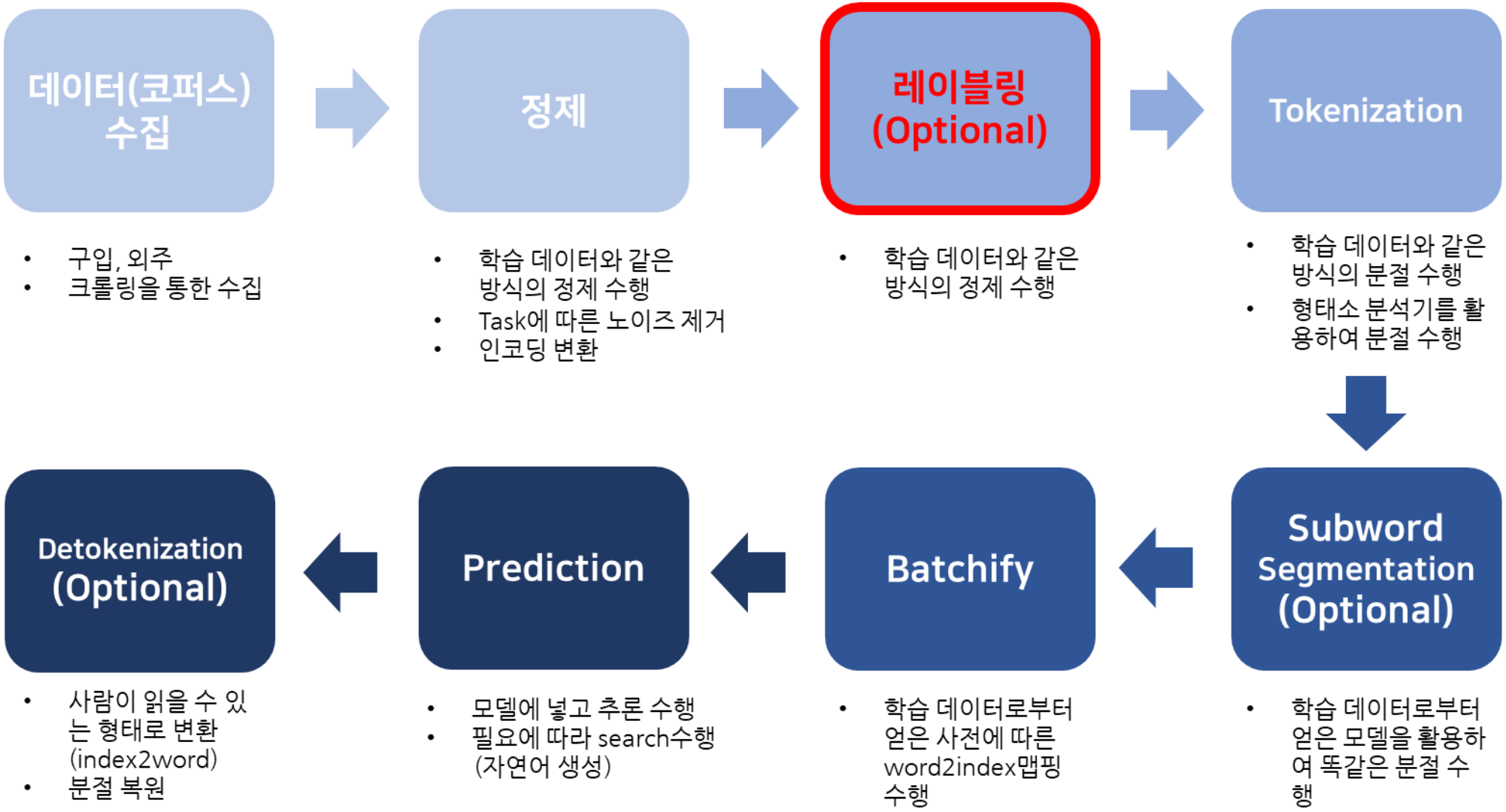
```
def cleaning_text(text, punc, regex):

# 노이즈 유형 (1), (2), (3)을 입력하세요.
        return

print(cleaning_text(corpus, [".", ",", "!", "?"], "([^a-zA-Z0-9.,?!₩n])"))
```

- 노이즈 유형 1,2,3을 다 사용해서 정제하기!

Labeling

백혜림 hreeee@yonsei.ac.kr

서비스 전체 Pipeline



Label

- Text Classification
  - Input : sentence
  - Output : class

- Token Classification
  - Input : sentence
  - Output : tag for each token -> sequence

- Sequence to Sequence
  - Input : sentence
  - Output : sentence

Label Example

- **Sentence > Class**
  - TSV형태의 하나의 파일
    - 각 row가 문장과 대응되는 레이블
    - 문장 column과 레이블 colunmn구성

- **Sentence > Sentence (Sequence)**
  - TSV형태의 하나의 파일
    - 각 row가 대응되는 문장 쌍
    - 각 문장 별로 column을 구성

  - 두 개 이상의 파일로 구성
    - 같은 순서의 row가 대응되는 문장 쌍
    - 한 문장당 여러 레이블이 존재 할 경우
      - Ex. 한국어 ↔ 영어 ↔ 중국어

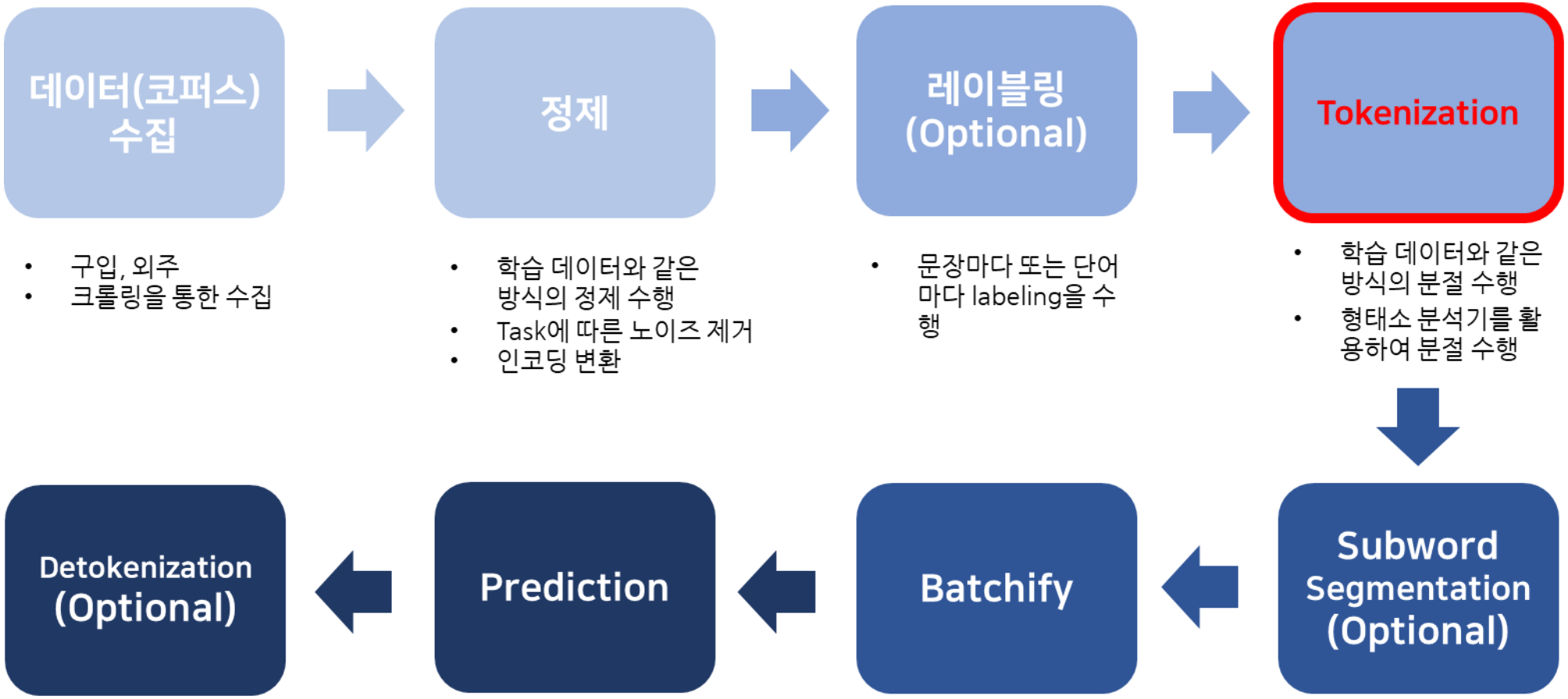
Tip 레이블링 직접 진행하기

- Human labeling은 prototyping시, 굉장히 강력한 도구!
- 효율적인 레이블링 도구를 구성하자 (ex. 엑셀)

Tokenization

백혜림 hreeee@yonsei.ac.kr

서비스 전체 Pipeline



- 구입, 외주
- 크롤링을 통한 수집

- 학습 데이터와 같은 방식의 정제 수행
- Task에 따른 노이즈 제거
- 인코딩 변환

- 문장마다 또는 단어마다 labeling을 수행

- 학습 데이터와 같은 방식의 분절 수행
- 형태소 분석기를 활용하여 분절 수행

- 학습 데이터로부터 얻은 모델을 활용하여 똑같은 분절 수행

- 학습 데이터로부터 얻은 사전에 따른 word2index맵핑 수행

- 모델에 넣고 추론 수행
- 필요에 따라 search수행 (자연어 생성)

- 사람이 읽을 수 있는 형태로 변환 (index2word)
- 분절 복원

2 Steps!

1. Sentence Segmentation
2. Tokenization

Sentence Segmentation

- 보통 훈련 시 우리가 원하는 데이터는
  - 1 sentence/line

- 우리가 수집한 corpus는
  - 한 라인에 여러 문장이 들어있거나,
  - 한 문장이 여러 라인에 들어있음

- Sentence Segmentation을 통해 원하는 형태로 변환
  - 마침표등을 단순히 문장의 끝으로 처리하면 안됨!
  - ex. 3.141592, U.S

- NLTK를 활용하여 변환 가능
  - `from nltk.tokenize import sent_tokenize`

Multiple sentence/line

현재 TED 웹사이트에는 1,000개가 넘는 TED강연들이 있습니다. 여기 계신 여러분의 대다수는 정말 대단한 일이라고 생각하시겠죠 -- 전 다릅 니다. 전 그렇게 생각하지 않아요. 저는 여기 한 가지 문제점이 있다고 생각합니다. 왜냐하면 강연이 1,000개라는 것은, 공유할 만한 아이디어들 이 1,000개 이상이라는 뜻이 되기 때문이죠. 도대체 무슨 수로 1,000개나 되는 아이디어를 널리 알릴 건가요?

```
현재 TED 웹사이트에는 1,000개가 넘는 TED강연들이 있습니다.
여기 계신 여러분의 대다수는 정말 대단한 일이라고 생각하시겠죠 -- 전 다릅니다.
전 그렇게 생각하지 않아요.
저는 여기 한 가지 문제점이 있다고 생각합니다.
왜냐하면 강연이 1,000개라는 것은, 공유할 만한 아이디어들이 1,000개 이상이라는 뜻이 되기 때문이죠.
도대체 무슨 수로 1,000개나 되는 아이디어를 널리 알릴 건가요?
```

Multiple sentence/sentence

현재 TED 웹사이트에는 1,000개가 넘는 TED강연들이 있습니다.
여기 계신 여러분의 대다수는
정말 대단한 일이라고 생각하시겠죠 --
전 다릅니다. 전 그렇게 생각하지 않아요.
저는 여기 한 가지 문제점이 있다고 생각합니다.
왜냐하면 강연이 1,000개라는 것은,
공유할 만한 아이디어들이 1,000개 이상이라는 뜻이 되기 때문이죠.
도대체 무슨 수로
1,000개나 되는 아이디어를 널리 알릴 건가요?
1,000개의 TED 영상 전부를 보면서

현재 TED 웹사이트에는 1,000개가 넘는 TED강연들이 있습니다.
여기 계신 여러분의 대다수는 정말 대단한 일이라고 생각하시겠죠 -- 전 다릅니다.
전 그렇게 생각하지 않아요.
저는 여기 한 가지 문제점이 있다고 생각합니다.
왜냐하면 강연이 1,000개라는 것은, 공유할 만한 아이디어들이 1,000개 이상이라는 뜻이 되기 때문이죠.
도대체 무슨 수로 1,000개나 되는 아이디어를 널리 알릴 건가요?
1,000개의 TED 영상 전부를 보면서

Tokenization

- Why?
  - 두 개이상의 다른 token들의 결합으로 이루어진 단어를 쪼개어,
  - Vocabulary 숫자를 줄이고, 희소성(sparseness)를 낮추기 위함

- Example

before:

> North Korea's state mouthpiece, the Rodong Sinmun, is also keeping mum on Kim's summit with Trump while denouncing ever-tougher U.S. sanctions on the rogue state.

after:

> North Korea 's state mouthpiece , the Rodong Sinmun , is also keeping mum on Kim 's summit with Trump while denouncing ever-tougher U.S. sanctions on the rogue state .

Korean Tokenization

- Why?
  - 교착어 : 어근에 접사가 붙어 다양한 단어가 파생됨
  - 띄어쓰기 통일의 필요성

Tokenization for Other Languages

- 영어 : 띄어쓰기가 이미 잘되어 있음. NLTK를 사용하여 comma등 후처리
- 중국어 : 기본적인 띄어쓰기가 없음. Character단위로 사용해도 무방
- 일본어 : 기본적인 띄어쓰기가 없음.

형태소 분석 및 품사 태깅 (Part of Speech Tagging)

- 형태소 분석 : 형태소를 비롯하여, 어근, 접두사/접미사, 품사(POS, part-of-speech) 등 다양한 언어적 속성의 구조를 파악하는 것

- 품사 태깅 : 형태소의 뜻과 문맥을 고려하여 그것에 마크업을 하는 일

출처: https://konlpy-ko.readthedocs.io/ko/v0.4.3/morph/

POS Tagger for Other Languages

언어	프로그램명	제작언어	특징
한국어	Mecab	C++	일본어 Mecab을 wrapping. 속도가 가장 빠름
한국어	KoNLPy	복합	설치와 사용이 편리하나, 일부 tagger의 경우 속도가 느림
일본어	Mecab	C++	속도가 가장 빠름
중국어	Stanford Parser	Java	미국 스탠포드에서 개발
중국어	PKU Parser	Java	북경대학교에서 개발
중국어	Jieba	Python	가장 최근에 개발. Python으로 제작되어 시스템 구성에 용이

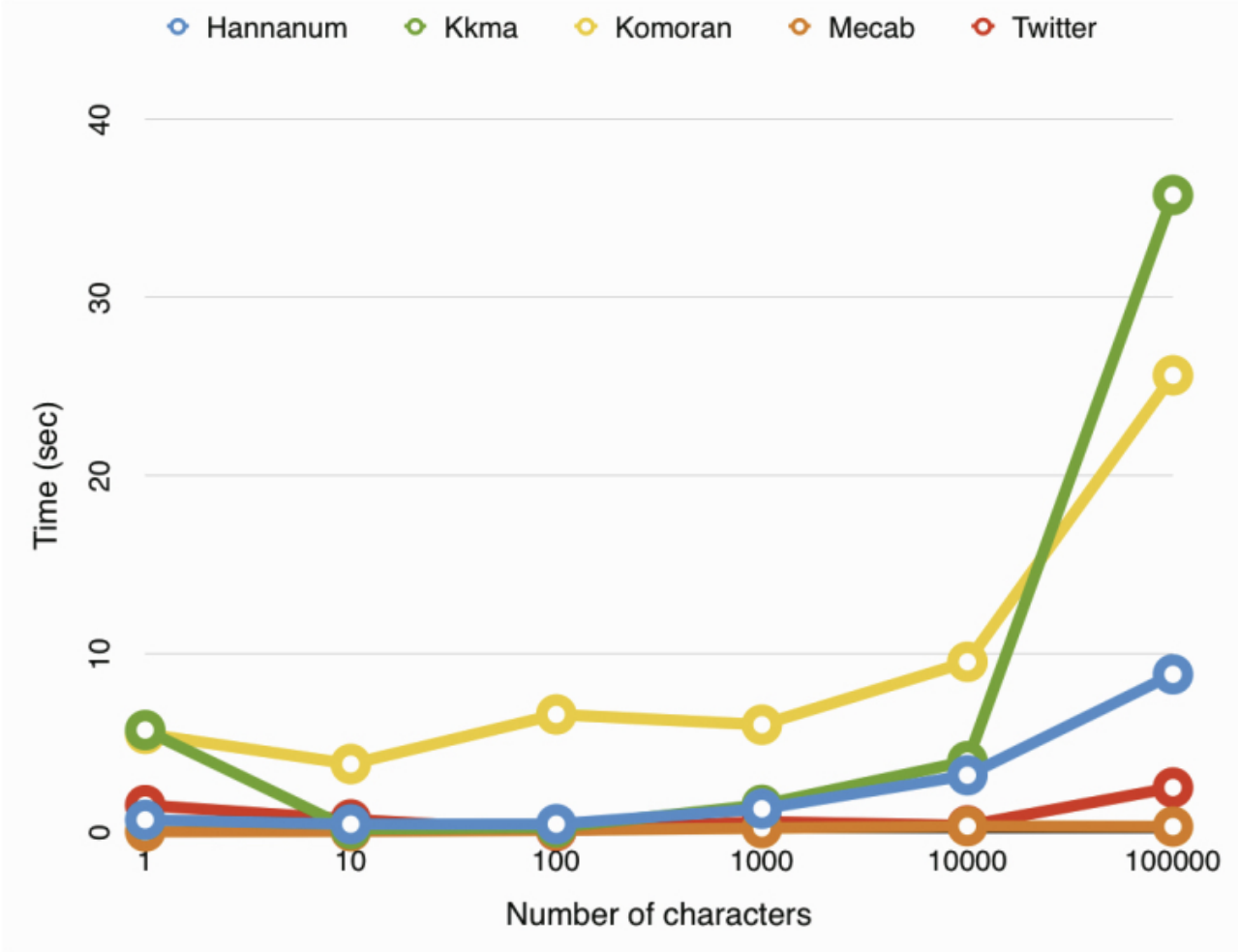
KoNLPy 내 분석기별 성능 차이 분석

- 로딩시간 : 분석기가 사용하는 사전(dictionary) 로딩을 포함해 형태소 분석기 클래스를 읽어 들이는 시간
- 실행시간 : 10만 문자의 문서를 분석하는 데 소요되는 시간

분석기명	로딩 시간	실행 시간
Kkma	5.6988	35.7163
Komoran	5.4866	25.6008
Hannanum	0.6591	8.8251
Okt(Twitter)	1.4870	2.4714
Mecab	0.0007	0.2838

문자 개수 대비 실행 시간

- 5개 형태소 분석기 모두 문자 개수가 많아질 때 실행 시간이 기하급수적으로 증가!



형태소 분석 품질 비교

- '아버지가방에들어가신다' 문장을 분석한 결과

Hannanum	Kkma	Komoran	Mecab	Okt
아버지가방에들어가/N	아버지/NNG	아버지가방에들어가신다/NNP	아버지/NNG	아버지/Noun
이/J	가방/NNG		가/JKS	가방/Noun
시ㄴ다/E	에/JKM		방/NNG	에/Josa
	들어가/VV		에/JKB	들어가신/Verb
	시/EPH		들어가/VV	다/Eomi
	ㄴ다/EFN		신다/EP+EC	

품사 태깅 예제 (feat. Mecab)

- 아버지가방에 들어가신다.
  - o아버지 NNG
  - o가 JKS
  - o방 NNG
  - o에 JKB
  - o들어가 VV
  - o신다 EP+EF
  - o. SF
  - o<EOS>
- 아버지가방에 들어가신다.
  - o아버지 NNG
  - o가방 NNG
  - o에 JKB
  - o들어가 VV
  - o신다 EP+EF
  - o. SF
  - o<EOS>

태그	설명	태그	설명
NNG	일반 명사	EP	선어말 어미
NNP	고유 명사	EF	종결 어미
NNB	의존 명사	EC	연결 어미
NNBC	단위를 나타내는 명사	ETN	명사형 전성 어미
NR	수사	ETM	관형형 전성 어미
NP	대명사	XPN	체언 접두사
VV	동사	XSN	명사 파생 접미사
VA	형용사	XSV	동사 파생 접미사
VX	보조 용언	XSA	형용사 파생 접미사
VCP	긍정 지정사	XR	어근
VCN	부정 지정사	SF	마침표, 물음표, 느낌표
MM	관형사	SE	줄임표 …
MAG	일반 부사	SSO	여는 괄호 (, [
MAJ	접속 부사	SSC	닫는 괄호),]
IC	감탄사	SC	구분자 , · / :
JKS	주격 조사	SY	
JKC	보격 조사	SL	외국어
JKG	관형격 조사	SH	한자
JKO	목적격 조사	SN	숫자
JKB	부사격 조사		
JKV	호격 조사		
JKQ	인용격 조사		
JX	보조사		
JC	접속 조사		

Summary

- 한국어의 경우
  1) 접사를 분리하여 희소성을 낮추고
  2) 띄어쓰기를 통일하기 위해 tokenization을 수행

- 굉장히 많은 POS Tagger가 존재하는데,
  - 전형적인 쉬운 문장(표준 문법을 따르며, 구조가 명확한 문장)의 경우, 성능이 비슷함
  - 하지만 신조어나 고유명사를 처리하는 능력이 다름
  - 따라서, 주어진 문제에 맞는 정책을 가진 tagger를 선택하여 사용해야 함.

Tokenization Style의 특성

백혜림 hreeee@yonsei.ac.kr

토큰 평균 길이에 따른 성격과 특징

토큰 길이가 짧을 수록

- Vocabulary 크기 감소
  - 희소성 문제 감소
- OOV가 줄어듬
- Sequence의 길이가 길어짐
  - 모델의 부담 증가
- 극단적 형태
  - Character단위

토큰 길이가 길 수록

- Vocabulary 크기 증가
  - 희소성 문제 증대
- OOV가 늘어남
- Sequence의 길이가 짧아짐
  - 모델의 부담 감소

토큰 길이에 따른 Trade off가 존재

정보량에 따른 이상적인 형태

- 빈도가 높을 경우, 하나의 token으로 나타내고,
- 빈도가 낮을 경우 더 잘게 쪼개어, 각각 빈도가 높은 token으로 구성한다.

압축 알고리즘?

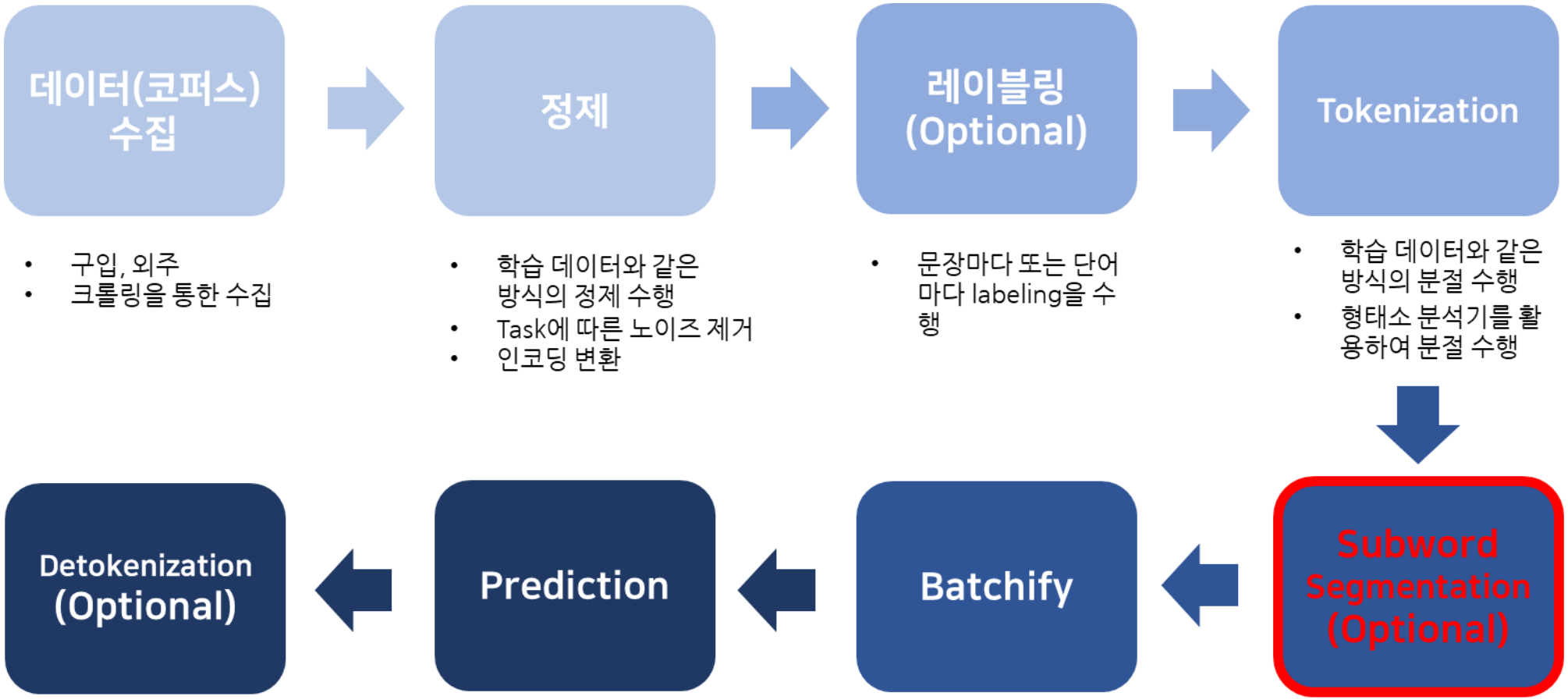
토큰화 실습을 해보자!

- Mecab을 이용하여 Colab에서 한국어 토큰화 실습을 해봅시다.

Subword Segmentation

백혜림 hreeee@yonsei.ac.kr

서비스 전체 Pipeline



단어보다 더 작은 의미 단위 : Subword

- 많은 언어들에서, 단어는 더 작은 의미 단위들이 모여 구성됨.

언어	단어	조합
영어	Concentrate	con(=together) + centr(=center) + ate(=make)
한국어	집중(集中)	集(모을 집) + 中(가운데 중)

- 따라서, 이러한 작은 의미 단위로 분절할 수 있다면 좋을 것
- 하지만, 이를 위해선 언어별 subword사전이 존재해야 할 것

단어보다 더 작은 의미 단위 : Subword

- 기계에 많은 단어를 학습하면 세상의 모든 단어를 알 수 있을까?
- 만약 기계가 모르는 단어가 등장하면 그 단어를 집합에 없는 단어란 의미에서 OOV(Out-of-Vacabulary) 또는 UNK(Unknown Token)이라 표현한다.
- 모르는 단어로 인해 문제를 푸는 것이 까다로워지는 상황을 OOV라 한다.

단어보다 더 작은 의미 단위 : Subword

- 서브워드 분리(Subword segmenation)잡업
  - 하나의 단어는 더 작은 단위의 의미있는 여러 서브워드들 (ex) birthplace = birth + place의 조합으로 구성된 경우가 많기 때문에 하나의 단어를 여러 Subword 로 분리해서 단어를 인코딩 및 임베딩하겠다는 의도를 가진 전처리 작업
- Preview와 predict를 보면 접두어인 pre가 공통이다.
- 컴퓨터도 이 두 단어를 따로 볼게 아니라, pre+view와 pre+dict로 본다면 학습을 더 잘 할 수 있겠죠?

단어보다 더 작은 의미 단위 : Subword

- 이를 통해 OOV나 희귀단어, 신조어 같은 문제들을 완화
- 실제로 언어의 특성에 따라 영어권 언어나 한국어는 서브워드 분리를 시도했을 때, 어느정도 의미있는 단위로 나누는 것이 가능
- 이런 작업을 하는 토크나이저를 **서브워드 토크나이저(Subword tokenizer)로** 명명

단어보다 더 작은 의미 단위 : Subword

- OOV문제를 완화하는 대표적인 서브워드 분리 알고리즘인 BPE(Byte Pair Encoding)알고리즘을 소개
- 실무에 사용할 수 있도록 구현한 센텐스피스(Sentencepiece)를 소개

Byte Pair Encoding (BPE) 알고리즘

- 압축 알고리즘을 활용하여 subword segmentation을 적용 [Sennrich at el., 2015]

- 학습 코퍼스를 활용하여 BPE 모델을 학습 후, 학습/테스트 코퍼스에 적용

- 장점:
  - 희소성을 통계에 기반하여 효과적으로 낮출 수 있다.
  - 언어별 특성에 대한 정보 없이, 더 작은 의미 단위로 분절 할 수 있다.
  - OoV를 없앨 수 있다. (seen character로만 구성될 경우)

- 단점:
  - 학습 데이터 별로 BPE 모델도 생성됨

BPE Training & Applying

- Training
  - ① 단어 사전 생성 (빈도 포함)
  - ② Character 단위로 분절 후, pair 별 빈도 카운트
  - ③ 최빈도 pair를 골라, merge 수행
  - ④ Pair 별 빈도 카운트 업데이트
  - ⑤ 3 번 과정 반복

- Applying
  - ① 각 단어를 character 단위로 분절
  - ② 단어 내에서 '학습 과정에서 merge에 활용된 pair의 순서대로' merge 수행

BPE Training Example

vocab {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}
pairs (l, o): 7, (o, w): 7, (w, </w>): 5, (w, e): 8, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (e, s): 9, (s, t): 9, (t, </w>): 9, (w, i): 3, (i, d): 3, (d, e): 3
best pair ('e', 's') 9

vocab {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w **es** t </w>': 6, 'w i d **es** t </w>': 3}
pairs (l, o): 7, (o, w): 7, (w, </w>): 5, (w, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, es): 6, (es, t): 9, (t, </w>): 9, (w, i): 3, (i, d): 3, (d, es): 3
best pair ('es', 't') 9

vocab {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w **est** </w>': 6, 'w i d **est** </w>': 3}
pairs (l, o): 7, (o, w): 7, (w, </w>): 5, (w, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, est): 6, (est, </w>): 9, (w, i): 3, (i, d): 3, (d, est): 3
best pair ('est', '</w>') 9

vocab {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w **est</w>**': 6, 'w i d **est</w>**': 3}
pairs (l, o): 7, (o, w): 7, (w, </w>): 5, (w, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('l', 'o') 7

vocab {'**lo** w </w>': 5, '**lo** w e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3}
pairs (lo, w): 7, (w, </w>): 5, (w, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, est</w>): 6, (w, i): 3, (i, d): 3, (d, est< /w>): 3
best pair ('lo', 'w') 7

BPE Training Example

vocab {'**low** </w>': 5, '**low** e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3}
pairs (low, </w>): 5, (low, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('n', 'e') 6

vocab {'low </w>': 5, 'low e r </w>': 2, '**ne** w est</w>': 6, 'w i d est</w>': 3}
pairs (low, </w>): 5, (low, e): 2, (e, r): 2, (r, </w>): 2, (ne, w): 6, (w, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('ne', 'w') 6

vocab {'low </w>': 5, 'low e r </w>': 2, '**new** est</w>': 6, 'w i d est</w>': 3}
pairs (low, </w>): 5, (low, e): 2, (e, r): 2, (r, </w>): 2, (new, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('new', 'est</w>') 6

vocab {'low </w>': 5, 'low e r </w>': 2, '**newest</w>**': 6, 'w i d est</w>': 3}
pairs (low, </w>): 5, (low, e): 2, (e, r): 2, (r, </w>): 2, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('low', '</w>') 5

vocab {'**low</w>**': 5, 'low e r </w>': 2, 'newest</w>': 6, 'w i d est</w>': 3}
pairs (low, e): 2, (e, r): 2, (r, </w>): 2, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('w', 'i') 3

vocab {'low</w>': 5, 'low e r </w>': 2, 'newest</w>': 6, '**wi** d est</w>': 3}

Segmentation Example

lastest news

1) l a t e s t </w> n e w s </w>
2) l a t es t </w> n e w s </w>
3) l a t est </w> n e w s </w>
4) l a t est</w> n e w s </w>
5) l a t est</w> ne w s </w>
6) l a t est</w> new s </w>

applicable pairs in order

1) ('e', 's')
2) ('es', 't')
3) ('est', '</w>')
4) ('l', 'o')
5) ('lo', 'w')
6) ('n', 'e')
7) ('ne', 'w')
8) ('new', 'est</w>')
9) ('low', '</w>')
10) ('w', 'i')

Segmentation Example

- Colab으로 실습해봅시다.

Subword Segmentation Modules

- Subword-nmt
  - https://github.com/rsennrich/subword-nmt
- WordPiece
  - Upgrade BPE version. Currently unavailable…?
- SentencePiece
  - https://github.com/google/sentencepiece

Wordpiece Model

- WordPiece Model은 BPE의 변형 알고리즘. 이하 WPM
- WPM은 BPE가 빈도수에 기반하여 가장 많이 등장한 쌍을 병합하는 것과 달리, 병합되었을 때, 코퍼스의 우도를 가장 높이는 쌍을 병합함.

WPM을 수행하기 이전의 문장: Jet makers feud over seat width with big orders at stake
WPM을 수행한 결과(wordpieces): _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

이 알고리즘은 최신 딥 러닝 모델 BERT를 훈련하기 위해서 사용되기도 하였습니다.

OOV가 미치는 영향

- 입력 데이터에 OoV가 발생할 경우, <UNK> 토큰으로 치환하여 모델에 입력
  - e.g. 나는 학교에 가서 밥을 먹었다. → 나는 <UNK>에 가서 <UNK>을 먹었다.
- 특히, 이전 단어들을 기반으로 다음 단어를 예측하는 task에서 치명적
  - e.g. Natural Language Generation
- 어쨌든 모르는 단어지만, 알고있는 subword들을 통해 의미를 유추해볼 수 있음
  - e.g. 버카충

Summary

- BPE 압축 알고리즘을 통해 통계적으로 더 작은 의미 단위(subword)로 분절 수행
- BPE를 통해 OoV를 없앨 수 있으며, 이는 성능상 매우 큰 이점으로 작용
- 한국어의 경우
  - 띄어쓰기가 제멋대로인 경우가 많으므로, normalization 없이 바로 subword segmentation을 적용하는 것은 위험
  - 따라서 형태소 분석기를 통한 tokenization을 진행한 이후, subword segmentaion을 적용하는 것을 권장

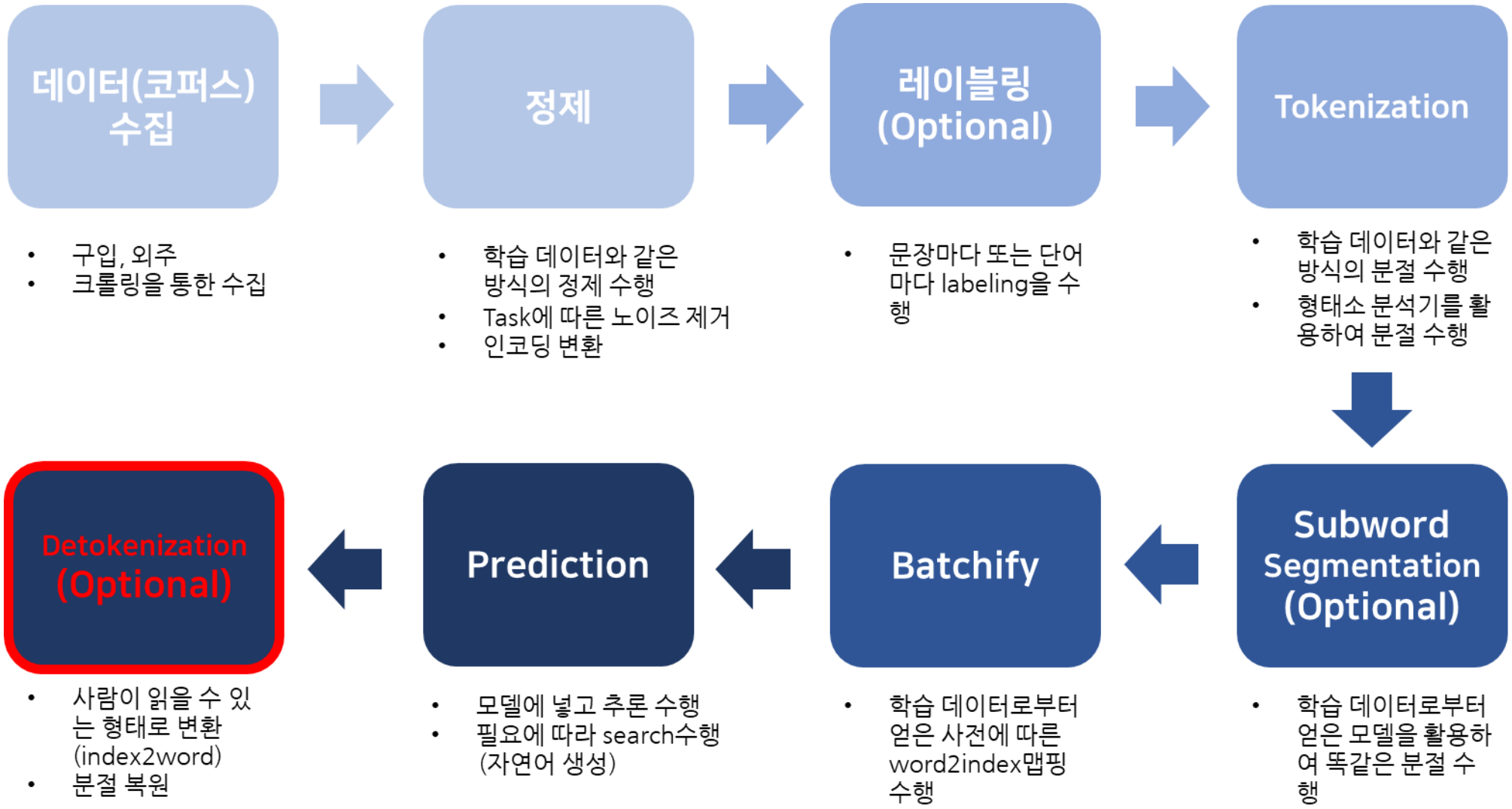
SubwordTextEncoder

- IMDB리뷰 토큰화를 실습해봅시다.

Detokenization

백혜림 hreeee@yonsei.ac.kr

서비스 전체 Pipeline



Tokenization

1. 영어 원문

   There's currently over a thousand TED Talks on the TED website.

2. tokenization을 수행하고, 기존 띄어쓰기와 구분을 위해 ▁ (U+2581) 삽입

   ▁There 's ▁currently ▁over ▁a ▁thousand ▁TED ▁Talks ▁on ▁the ▁TED ▁website .

3. subword segmentation을 수행, 공백 구분 위한 ▁ 삽입

   ▁▁There ▁'s ▁▁currently ▁▁over ▁▁a ▁▁thous and ▁▁TED ▁▁T al ks ▁▁on ▁▁the ▁▁TED ▁▁we b site ▁.

DeTokenization

1. whitespace를 제거
   ▁▁There▁'s▁▁currently▁▁over▁▁a▁▁thousand▁▁TED▁▁Talks▁▁on▁▁the▁
   ▁TED▁▁website▁.

2. ▁▁을 white space로 치환
   There▁'s currently over a thousand TED Talks on the TED website▁.

3. ▁를 제거
   There's currently over a thousand TED Talks on the TED website.

Summary

- 정제
  - Task와 언어 및 도메인에 따른 특성
    - 풀고자 하는 문제의 특성에 따라 전처리 전략이 다름
  - 끝이 없는 과정
    - 노력과 품질 사이의 trade-off
    - Sweet spot을 찾아야함

- 분절
  - 한국어의 경우 띄어쓰기 normalization을 위해 형태소 분석기 활용이 필요
  - Subword segmentation을 통해 좀 더 잘게 분절 할 수 있음

- 모두 비슷한 알고리즘을 사용하고 있으므로, 결국 데이터의 양과 품질이 좌우함
  - 따라서 전처리 과정을 경시해서는 안됨

수고하셨습니다.