자연어 처리 딥러닝 캠프

4장 전처리

최우진

2020/01/09

목차

4.1 전처리

- 코퍼스: '말뭉치'라고도 불리며, 보통 여러 단어들로 이뤄진 문장.
- 코퍼스의 종류

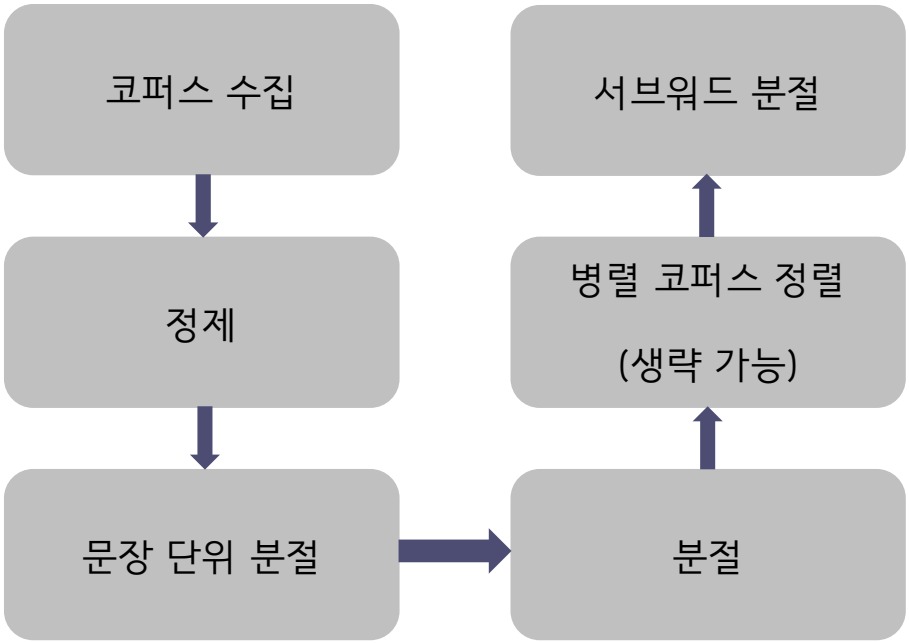
  - 단일 언어 코퍼스: 한 가지 언어로 구성
  - 이중 언어 코퍼스: 두 개의 언어로 구성
  - 다중 언어 코퍼스: 세 개이상의 언어로 구성
  - 병렬 코퍼스: 언어 간에 쌍으로 구성

영문	한글
I love to go to school.	나는 학교에 가는 것을 좋아한다.
I am a doctor.	나는 의사입니다.

→ 병렬 코퍼스 사례

4.1 전처리

- **전처리 과정 개요도**



4.2 코퍼스 수집

- 공개된 데이터 수집
- 크롤링을 통한 데이터 수집
  - 해당 웹사이트의 크롤링 허용 여부 : robots.txt를 통해 확인
  - ex) https://www.ted.com/robots.txt
- 크롤링을 사용을 위한 패키지
  - selenium : phantomJS, 크롬(헤드리스 브라우저)
  - beautiful-soup : HTML 파싱

```
User-agent: *
Disallow: /latest
Disallow: /latest-talk
Disallow: /latest-playlist
Disallow: /people
Disallow: /profiles
Disallow: /conversations
Disallow: /themes/rss

User-agent: Baiduspider
Disallow: /search
Disallow: /latest
Disallow: /latest-talk
Disallow: /latest-playlist
Disallow: /people
Disallow: /profiles
```

4.2 코퍼스 수집

- 4.2.1 단일 언어 코퍼스 수집

  - 캐글, 위키피디아, 위키 등

- 4.2.2 다중 언어 코퍼스 수집

  - 예시 사이트

  1. https://sites.google.com/site/koreanparalleldata/
  2. http://www.donga.com/en
  3. http://koreajoongangdaily.joins.com/news/list/List.aspx?gCat=060201
  4. http://opus.nlpl.eu/

4.3 정제

- 텍스트를 사용하기에 앞서 필수적인 과정
- Ex1) 음성 인식인 경우: 사람의 음성을 그대로 받아 적어야 하므로, 괄호 또는 별표와 같은 기호나 특수 문자는 제외
- Ex2) 전화번호나 이메일 주소와 같은 개인 정보는 제거하거나 암호화하여 저장

4.3 정제

- 4.3.1 전각 문자 제거

  -전각 문자를 일반적으로 사용되는 반각 문자로 제거

- 반각 문자: 123abc?! / 전각 문자: １２３ａｂｃ？！

- 반각/전각 문자로 혼용되는 대표적인 문자

！”＃＄％＆’（）＊＋，－．／0123456789
：；＜＝＞？＠ABCDEFGHIJKLMNOPQR
STUVWXYZ［＼］＾＿｀abcdefghijklmnopq
rstuvwxyz｛｜｝～

4.3 정제

- 4.3.2 대소문자 통일

번호	New York City
1	NYC
2	nyc
3	N.Y.C.
4	N.Y.C

- 하나의 의미를 지니는 여러 단어를 하나의 형태로 통일하여 '희소성'을 줄이는 효과기대

4.3 정제

- 4.3.3 정규 표현식을 사용한 정제

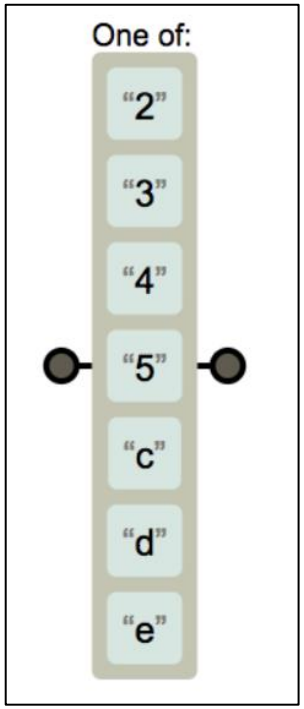
  - 크롤링을 통해 얻은 코퍼스의 노이즈를 제거

  - 사이트: https://regexper.com/

- [] 사용

  - [2345cde]

  - (2|3|4|5|c|d|e)

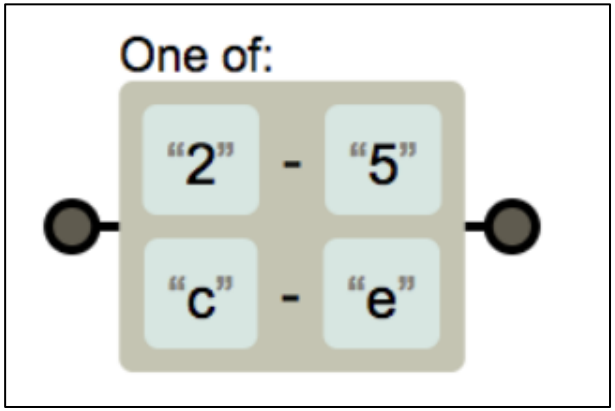


4.3 정제

- - 사용

  - '-'을 사용하여 연속된 숫자 또는 알파벳 등을 표현

  예) [2-5c-e]

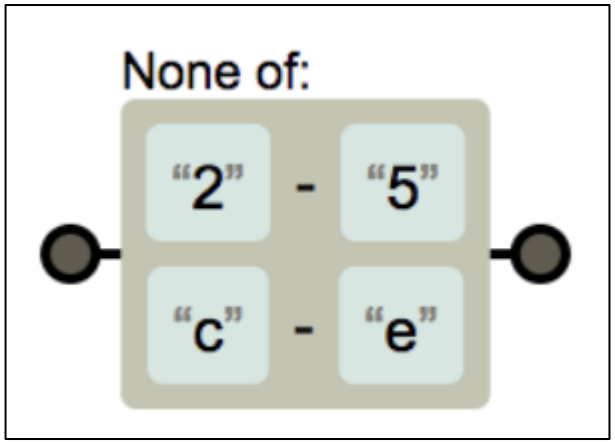


4.3 정제

- [^] 사용

  - Not을 기호 '^'를 써서 표현

  예) [^2-5c-e]

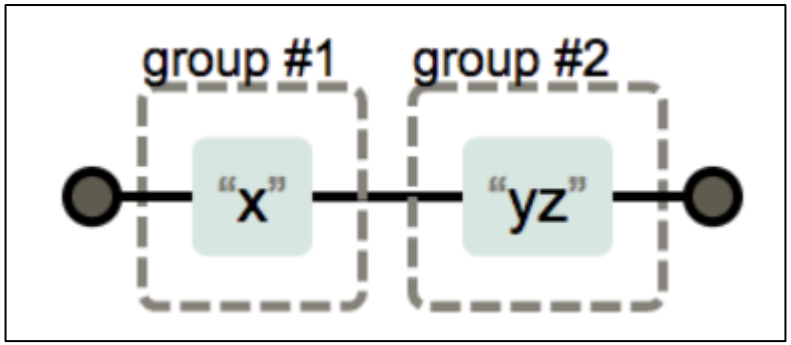


4.3 정제

- () 사용

  - 괄호를 이용해 그룹 생성

예) (x)(yz)

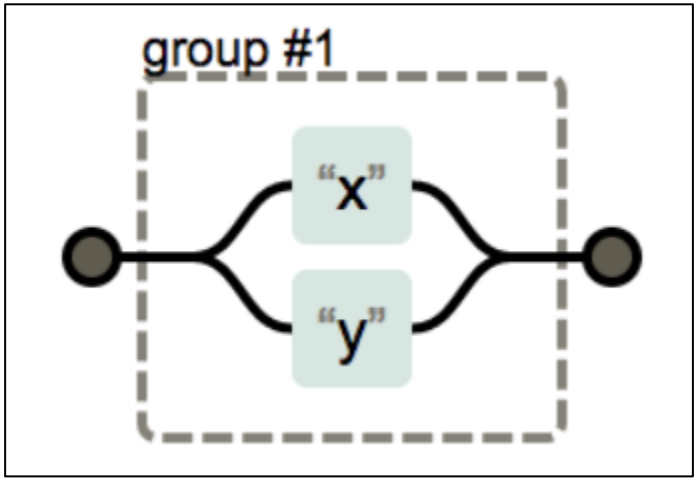


4.3 정제

- | 사용

  - '|'은 '또는'에 해당하는 or를 의미

  예) (x|y)

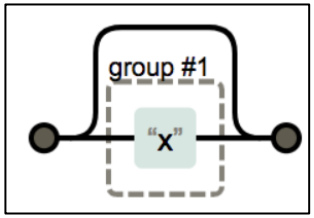


4.3 정제

- ?, *, + 사용

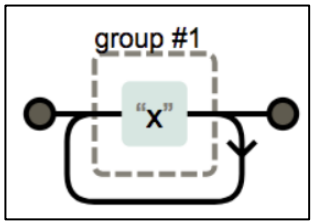
  - '?'는 앞의 수식하는 부분이 나타나지 않거나 한 번만 나타날 때 사용

  예) x?



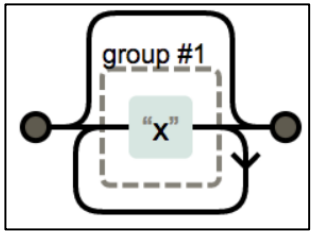
  - '+'는 앞의 수식하는 부분이 한 번 이상 나타날 때 사용

  예) x+



  - '*'는 앞의 수식하는 부분이 나타나지 않거나 여러 번 나타날 때 사용

  예) x*

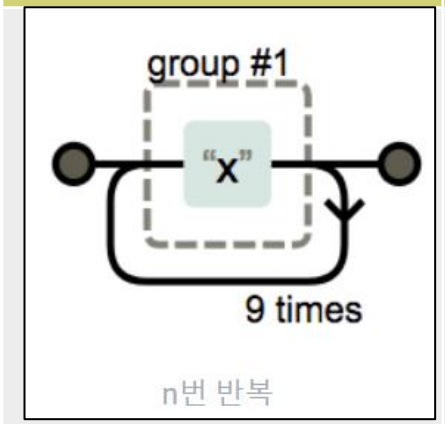
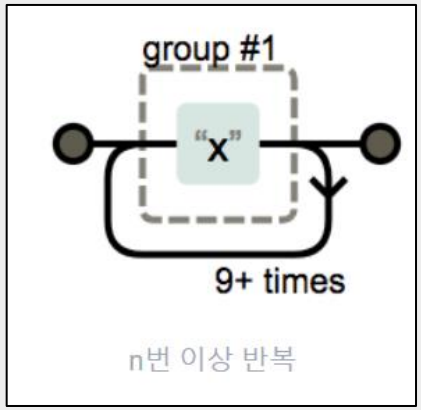
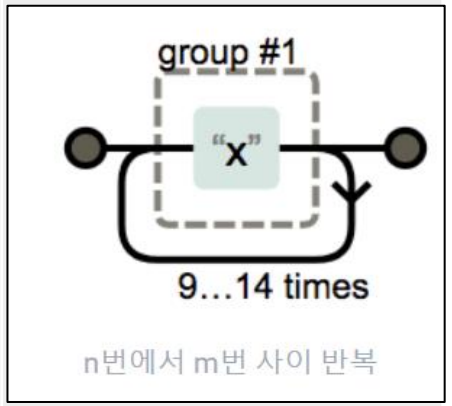


4.3 정제

- {n}, {n,}, {n,m} 사용

  - 정확하게 반복 횟수의 범위를 알 경우 사용

예) n = 9, m = 14

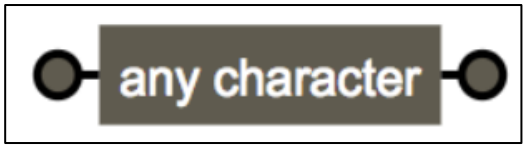
x{n}	x{n,}	x{n,m}
		

4.3 정제

- . 사용

  - 어떤 글자도 '.'에 포함

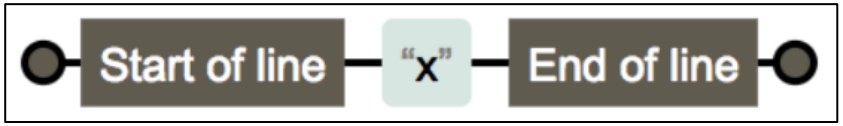
  예) .



- ^와 $ 사용

  - '[' 와 ']' 내에 포함되지 않은 '^'은 라인의 시작, '$'은 라인의 종료를 의미

  예) ^x$



4.3 정제

- 지정 문자 사용

  - 지정 문자를 사용하여 비슷한 유형의 글자들을 표현 가능

Meta Characters	Description
\s	공백문자(white space)
\S	공백문자를 제외한 모든 문자
\w	alphanumeric(알파벳 + 숫자) + '' *([A-Za-z0-9]*와 같음)
\W	non-alphanumeric 문자 *''도 제외 ([^A-Za-z0-9]*와 같음)
\d	숫자 ([0-9]와 같음)
\D	숫자를 제외한 모든 문자 (와 같음)

4.3 정제

- 예제

- 규칙

- 이름이 전화번호 앞에 나올 수도 있다.
- 이름 뒤에는 콜론(:)이 나올 수도 있다.
- 콜론 앞/뒤로는 공백(탭 포함)이 다수가 존재할 수도 있다.
- 전화번호는 국가번호를 포함할 수도 있다.
- 국가번호는 최대 3자리이다.
- 국가번호의 앞에는 '+'가 붙을 수도 있다.
- 전화번호 사이에 '-'가 들어갈 수도 있다.
- 전화번호는 빈칸이 없이 표현 된다.
- 전화번호의 맨 앞과 지역번호(또는 010)의 다음에는 괄호가 들어갈 수도 있다.
- 괄호는 한쪽만 나올 수도 있다.
- 지역번호 자리의 맨 처음 나오는 0은 빠질 수도 있다. 즉, 2자리가 될 수도 있다.
- 지역번호 다음 번호 그룹은 3에서 4자리 숫자이다.
- 마지막은 항상 4자리 숫자이다.

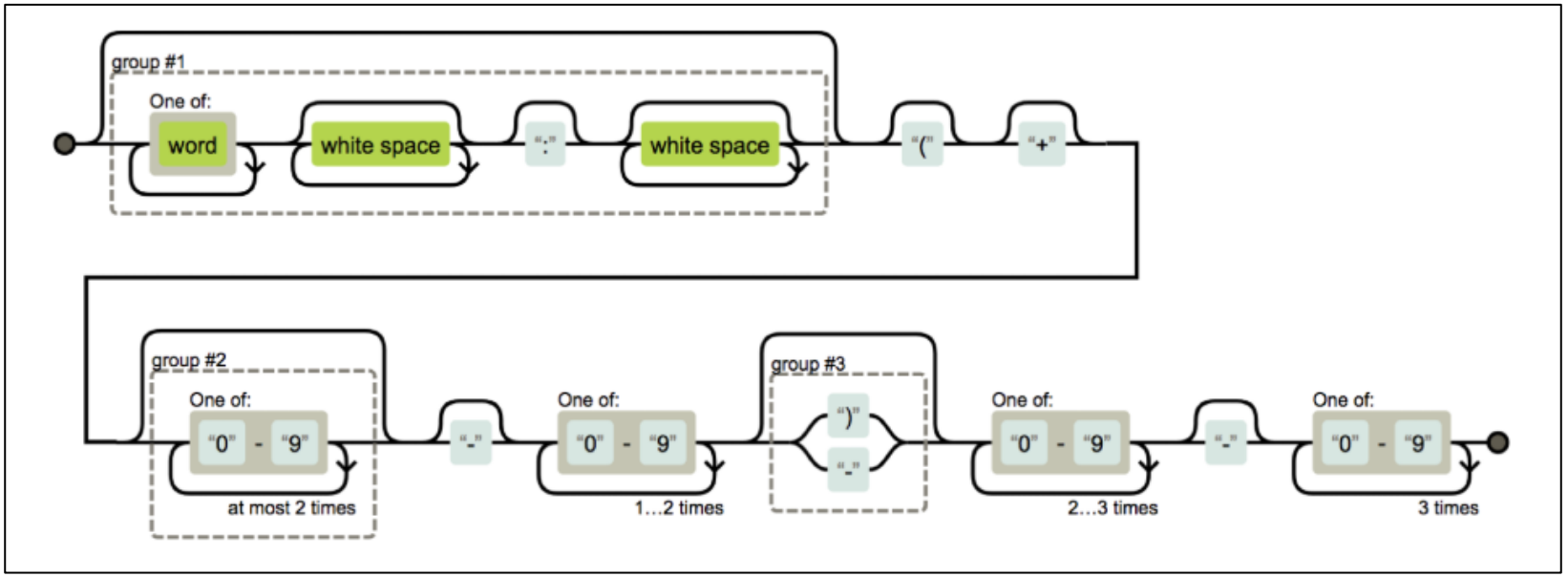
- 정규 표현식

```
([\w]+\s*:?\s*)?\(?\+?([0-9]{1,3})?\-?[0-9]{2,3}(\)|\-)?[0-9]{3,4}\-?[0-9]{4}
```

4.3 정제

- 예제

  - regexper.com에서 그림으로 표현



4.3 정제

- 파이썬에서 정규 표현식 사용

  - re를 import하여 사용

  - 코드

```
import re
regex = r"([₩w]+₩s*:?₩s*)?₩(?₩+?([0-9]{1,3})?₩-?[0-9]{2,3}(₩)|₩-)?[0-9]{3,4}₩-?[0-9]{4}"
x = "Ki: +82-10-9420-4104"
re.sub(regex, "REMOVED", x)
```

```
'REMOVED'
```

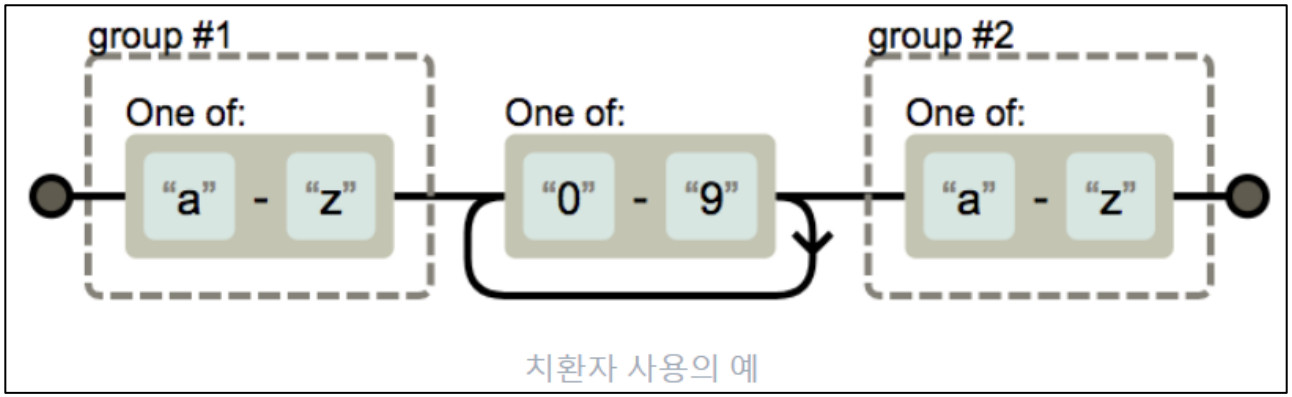
```
x = "CONTENT jiu 02)9420-4104"
re.sub(regex, "REMOVED", x)
```

```
'CONTENT REMOVED'
```

4.3 정제

- 치환자 사용

  - 예제: 알파벳(소문자) 사이에 있는 숫자를 제거



치환자 사용의 예

4.4 문장 단위 분절

- 대부분의 경우 한 라인에 한 문장만 있어야 한다.
- 단순히 마침표만을 기준으로 문장 단위 분절을 수행하면,
- U.S.와 같은 영어 약자나 3.141592와 같은 소수점 등 여러 가지 문제가 발생한다.
- 자연어 처리 툴킷: NLTK(3.2.5버전)

4.4 문장 단위 분절

- 한 라인에 여러 문장이 들어 있는 경우의 파이썬 스크립트 예제

```
import sys, fileinput, re
from nltk.tokenize import sent_tokenize
import nltk


x = ['자연어처리는 인공지능의 한 줄기 입니다. 시퀀스 투 시퀀스의 등장 이후로 딥러닝을 활용한 자연어처리는 ₩
새로운 전기를 맞이하게 되었습니다. 문장을 받아 단순히 수치로 나타내던 시절을 넘어, 원하는대로 문장을 ₩
만들어낼 수 있게 된 것입니다.']


if __name__ == "__main__":
    for line in x:
        if line.strip() != "":
            line = re.sub(r'([a-z])₩.([A-Z])', r'₩1. ₩2', line.strip())

            sentences = sent_tokenize(line.strip())

            for s in sentences:
                if s != "":
                    sys.stdout.write(s + "₩n")
```

```
자연어처리는 인공지능의 한 줄기 입니다.
시퀀스 투 시퀀스의 등장 이후로 딥러닝을 활용한 자연어처리는 새로운 전기를 맞이하게 되었습니다.
문장을 받아 단순히 수치로 나타내던 시절을 넘어, 원하는대로 문장을 만들어낼 수 있게 된 것입니다.
```

4.4 문장 단위 분절

- 여러 라인에 걸쳐 한 문장이 들어 있는 경우의 파이썬 스크립트 예제

```
import sys, fileinput
from nltk.tokenize import sent_tokenize



x = ['자연어처리는 인공지능의 한 줄기 입니다. 시퀀스 투 시퀀스의 등장 이후로 ₩n',
'딥러닝을 활용한 자연어처리는 새로운 전기를 맞이하게 되었습니다. 문장을 ₩n',
'받아 단순히 수치로 나타내던 시절을 넘어, 원하는대로 문장을 만들어낼 수 ₩n',
'있게 된 것입니다.']


if __name__ == "__main__":
    buf = []

    for line in x:
        if line.strip() != '':
            buf += [line.strip()]
            ### "x".join(list) - list의 값 사이에 x를 넣은 후 str으로 변경 ###
            sentences = sent_tokenize(" ".join(buf))

            if len(sentences) > 1:
                buf = sentences[-1:]

                sys.stdout.write('₩n'.join(sentences[:-1]) + '₩n')

    sys.stdout.write(' '.join(buf) + '₩n')
```

```
자연어처리는 인공지능의 한 줄기 입니다.
시퀀스 투 시퀀스의 등장 이후로 딥러닝을 활용한 자연어처리는 새로운 전기를 맞이하게 되었습니다.
문장을 받아 단순히 수치로 나타내던 시절을 넘어, 원하는대로 문장을 만들어낼 수 있게 된 것입니다.
```

4.5 분절

- 한국어 분절

  - Mecab, KoNLPy

- 영어 분절

  - Moses

- 중국어 분절

  - 스탠포드 파서, JIEBA

4.6 병렬 코퍼스 정렬

- **정렬을 수행하기 위한 전체 과정**

1. 소스 언어(source language)와 타깃 언어(target language) 사이의 단어 사전을 준비합니다.
2. 만약 준비된 단어 사전이 없다면 다음 작업을 수행합니다. 만약 이미 단어 사전을 갖고 있다면 7번으로 건너뜁니다.
3. 각 언어에 대해서 코퍼스를 수집 및 정제합니다.
4. 각 언어에 대해서 단어 임베딩 벡터를 구합니다. 단어 임베딩 벡터에 대해서는 추후 다루겠습니다.
5. MUSE를 통해 단어 레벨 번역기를 훈련합니다.
6. 훈련된 단어 레벨 번역기를 통해 두 언어 사이의 단어 사전을 생성합니다.
7. 만들어진 단어 사전을 넣어 Champollion을 통해 기존에 수집된 다중 언어 코퍼스를 정렬합니다.
8. 각 언어에 대해서 단어 사전을 적용하기 위해 알맞은 수준의 분절을 수행합니다.
9. 각 언어에 대해서 정제를 수행합니다.
10. Champollion을 사용하여 병렬 코퍼스를 생성합니다.

4.6 병렬 코퍼스 정렬

- 사전 생성

  - 페이스북의 MUSE는 병렬 코퍼스가 없는 상황에서 사전을 구축하는 방법과 코드

  - MUSE: 비지도 학습

  - MUSE를 통한 사전 생성

```
stories <> 이야기
stories <> 소설
contact <> 연락
contact <> 연락처
contact <> 접촉
green <> 녹색
green <> 초록색
green <> 빨간색
dark <> 어두운
dark <> 어둠
dark <> 짙
song <> 노래
song <> 곡
song <> 음악
salt <> 소금
```

4.7 서브워드 분절

- BPE(byte per encoding) 알고리즘을 통해 서브워드 분절
- 가정: '단어는 의미를 가진 더 작은 서브워드들의 조합으로 이루어진다'
- 서브워드를 활용하여 어휘 수와 희소성을 효과적으로 줄일 수 있다.
- UNK(unknown) 토큰에 대한 효율적인 대처

언어	단어	조합
영어	concentrate	Con(=together) + centr(=center) + ate(=make)
한국어	집중(輯中)	輯(모을 집) + 中(가운데 중)

▶ 영어와 한국어의 서브워드 분절 사례

4.7 서브워드 분절

- **예제**

- 기본 문장

자연어처리는 인공지능의 한 줄기 입니다.
시퀀스 투 시퀀스의 등장 이후로 딥러닝을 활용한 자연어처리는 새로운 전기를 맞이하게 되었습니다.

- 분절 문장

자연어 처리 는 인공지능 의 한 줄기 입니다 .
시퀀스 투 시퀀스 의 등장 이후 로 딥 러닝 을 활용 한 자연어 처리 는 새로운 전기 를 맞이 하 게 되 었 습니다 .

- 서브워드 분절 문장

_자연 어 _처리 _는 _인공지능 _의 _한 _줄기 _입니다 _.
_시퀀스 _투 _시퀀스 _의 _등장 _이후 _로 _딥 _러 닝 _을 _활용 _한 _자연 어 _처리 _는
_새로운 _전기 _를 _맞이 _하 _게 _되 _었 _습니다 _.

4.8 분절 복원

- 영어 원문

Natural language processing is one of biggest streams in artificial intelligence, and it becomes very popular after seq2seq's invention.

- 단어 분절

__Natural __language __processing __is __one __of __biggest __streams __in __artificial __intelligence , __and __it __becomes __very __popular __after __seq2seq 's __invention .

- 서브워드 단위 분절

____Natural ____language ____processing ____is ____one ____of ____biggest ____streams ____in ____artificial ____intelligence __, ____and ____it ____becomes ____very ____popular ____after ____se q 2 se q __'s ____invention __.

4.8 분절 복원

- 공백 제거

____Natural____language____processing____is____one____of____biggest____streams____in____artificial____intelligence__,____and____it____becomes____very____popular____after____seq2seq__'s____invention__.

- ____을 공백으로 치환

Natural language processing is one of biggest streams in artificial intelligence__, and it becomes very popular after seq2seq__'s invention__.

- __을 제거

Natural language processing is one of biggest streams in artificial intelligence, and it becomes very popular after seq2seq's invention.

4.9 토치텍스트

- 자연어 처리 문제 또는 텍스트에 관한 머신러닝이나 딥러닝을 수행하는 데이터를 읽고 전처리하는 코드를 모아둔 라이브러리
- 세 가지 형태로 분류되는 학습 데이터

X 데이터	Y 데이터	활용분야
코퍼스	클래스	텍스트 분류, 감성 분석
코퍼스	-	언어 모델
코퍼스	코퍼스	기계번역, 요약, 질의응답

4장 전처리

감사합니다