OOV & TOKENIZER

안승환[1]

2020년 1월

[1]통계학과
서울시립대학교

Problem

open-vocabulary problem?

- 다양한 언어를 아우르는 NMT
  - 한글, 영어
  - R code
- 끊임없이 만들어지는 새로운 단어의 존재 = OOV
  - 합성어 등
- 단어 사전의 크기를 최소화(데이터 크기 문제)

사용한 데이터: aihub 기계독해 데이터

언어 모형 사용 - 단점

- 사전의 최소 단위 = 단어(word)
  - 기존에 구축되어 있는 사전을 이용하므로 신조어와 같은 OOV 문제에 취약함
- 오타와 같은 데이터 noise에 취약
  - 한글화 정제, 정규화 등에서 어려움
- 구축된 패키지에 의존해야 함(KoNLPy)
- 전처리 Workflow

Byte Pair Encoding

- segmentation = sequence(문장) → word(단어, 최소의미) → subword(부분단어) → character(음절)
- 사전 = {subword : frequency or score, index, ...}
- BPE 개념
- BPE 적용

BPESegmenter

Build Vocabulary

1. Joint Corpus 구축: Source data와 Target data를 concatenate
2. Joint Corpus에 대해 충분히 큰 초기(seed) 사전 구축 by BPE
3. 원하는 비율로 사전을 축소(vocabulary reduction)

BPESegmenter

Tokenize

1. 주어진 문장에 대해 가능한 모든 경우의 수의 tokenized 순서열 계산 $=$ 후보 순서열들
2. 각 후보 순서열에 대해 각 subword의 빈도수를 이용해 확률을 계산
3. 가장 높은 확률을 가지는 후보 순서열을 최종 tokenized 결과로 선택

BPESegmenter

Vocabulary Reduction

1. 각 subword에 대해 score를 저장
2. score는 subword가 어떤 tokenized 결과에 포함되면 이 tokenized 결과들의 확률값들의 합으로 계산
    - tokenize 과정에서 동시에 계산됨
3. score를 기준으로 주어진 비율만큼의 상위 score의 subword만을 남기고 나머지는 삭제
    - OOV 문제 방지를 위해 character 단위(최소 크기)의 subword는 반드시 유지