8강 Natural Language Processing Basics

NLP를 구성하는 3가지 요소

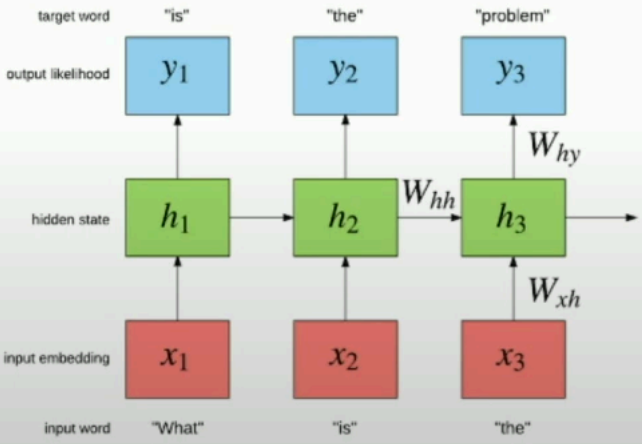
1. 확률(그래프 이론)방법론
2. 언어학적 방법론
3. 알고리즘 방법론

Natural Language Processing

Natural Language Processing

- **Def:** A subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.

- Most Popular Tasks: Language Modeling
  - To compute $P(w_i \mid w_1, \ldots, w_{i-1})$

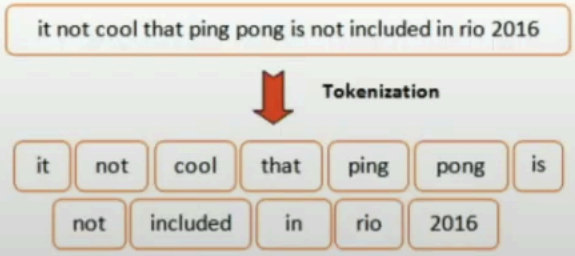


Tokenization

Tokenization

- **How to separate whole texts to several tokens**

- **Simple Methods**:
  - Tokenize with character (eg: ‘a’, ‘b’, …, ‘1’, ‘2’, …, ‘$’, ‘%’, …) or words followed by white space. (‘This ’, ‘is ’, ‘my ’, ‘pen ’, ‘.’)
  - Tokenize based on dictionary, to found idiom or phrase.
  - Rule-based tokenization. (eg. Penn Treebank Tokenization: “don’t” -> ‘do’, ‘n’t’ | “they’ll” -> ‘they’, ‘’ll’ | “$3.88” -> ‘$’, “3.88”)



Tokenization

- **Advanced Methods: Sub-word Tokenization**
  - Frequently used words are assigned a unique id
  - Unfrequently, or rarely used words are split into frequently used sub-words. (eg. “fewest” -> ‘few’, ‘es’, ‘t’ | “Unfriendly” -> ‘Un’, ‘friend’, ‘ly’)
  - **How to decide sub-words?:** BPE (Byte Pair Encoding), Unigram, WordPiece, SentencePiece.

Subwords (_ means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/l/o/_world	7 18085 356 356 137 255
_H/el/l/o/_/world	320 585 356 137 7 12295

Table 1: Multiple subword sequences encoding the same sentence “Hello World”

Sub-word

단어를 쪼개어 각기 다른 단어로 취급함

문제 상황:

단어 단위로 학습을 진행하는 경우 유행어, 신조어 등 사전에 없던 단어들이 등장하면 처리가 불가능한 상황이 나옴

Sub-word 활용의 장점

sub word 형태로 처리하게 되면 신조어 등의 새로운 단어 혹은 잘 쓰이지 않는 희귀한 단어도 기존의 단어들의 조합으로 이루어지는 경우가 많기 때문에 처리가 가능해진다.

Byte Pair Encoding(BPE)

Byte Pair Encoding (BPE) [R. Sennrich *et al.* 2015]

-- To solve out-of-vocabulary (OOV) problems.

0. From character based tokenization on large text corpus,

- [dict:frequency] = [l o w : 5, l o w e r : 2, n e w e s t : 6, w i d e s t : 3]
- [vocab] = [l, o, w, e, r, n, w, s, t, i, d]

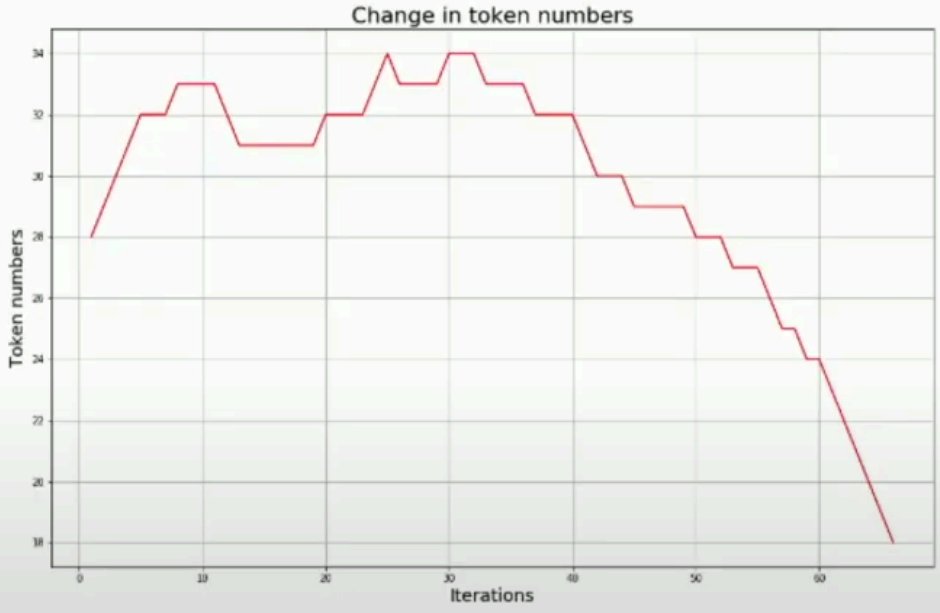
1. Find the highest frequency sub-word pairs and add it to vocab.

- 'e', 's' is the highest frequency(9 times) sub-word pairs with 6 times on newest, 3 times on widest
- [dict:frequency] = [l o w : 5, l o w e r : 2, n e w es t : 6, w i d es t : 3]
- [vocab] = [l, o, w, e, r, n, w, s, t, i, d, es]

2. Iterate 1. until reach desired vocab size.

- 'es', 't' is the highest frequency(9 times) sub-word pairs with 6 times on newest, 3 times on widest.
- [dict:frequency] = [l o w : 5, l o w e r : 2, n e w est : 6, w i d est : 3]
- [vocab] = [l, o, w, e, r, n, w, s, t, i, d, es, est]
- If vocab has 0 frequency after merging, delete it.

Byte Pair Encoding (BPE) [R. Sennrich *et al.* 2015]



BPE(Byte Pair Encoding)는 희귀 단어 문제를 해결하고 어휘 크기를 줄이기 위해 사용되는 하위 단어(Subword) 토큰화 알고리즘

BPE의 주요 목표

1. **Out-of-Vocabulary(OOV) 문제 해결**:
    - 희귀 단어를 더 작은 단위(Subword)로 나누어 처리하므로 새로운 단어에 대해 유연하다.
2. **어휘 크기 축소**:
    - 자주 나타나는 하위 단어를 병합하여 어휘 크기를 제한한다.

BPE 알고리즘 단계

0. 초기 설정

- 입력 데이터를 문자(Character) 기반 토큰화로 시작한다.
- 각 단어를 문자 단위로 분리하고 빈도수를 계산한다.

1. 가장 자주 등장하는 문자 쌍(Subword Pair) 찾기

- 가장 자주 등장하는 문자 쌍을 찾고 이를 병합한다.

2. 병합 반복

- 1단계를 반복하며 자주 등장하는 쌍을 계속 병합한다.
- 원하는 어휘 크기(vocab size)에 도달할 때까지 진행한다.

3. 병합 중단 조건

- 병합 후 빈도수가 0이 된 쌍은 삭제한다.
- 어휘 크기 제한에 도달하면 병합을 중단한다.

BPE의 장점

1. **OOV 문제 해결**:
    - 희귀 단어도 하위 단위로 나뉘어 처리되므로 유연하다.
2. **효율성**:
    - 어휘 크기를 줄여 모델 메모리 사용량을 절약한다.
3. **일반화**:
    - 동일한 하위 단어를 사용해 비슷한 단어를 처리할 수 있다.

BPE의 단점

1. **문맥 정보 손실**:
    - Subword 단위로 분리되므로 문맥 정보가 손실될 수 있다.
2. **동일한 병합 규칙 사용**:
    - 모든 데이터셋에 동일한 병합 규칙을 사용하므로, 특정 데이터셋에 최적화되지 않을 수 있다.