NATURAL LANGUAGE PROCESSING

LECTURE 8: Tokenization

goorm

KAIST AI
Graduate School of AI

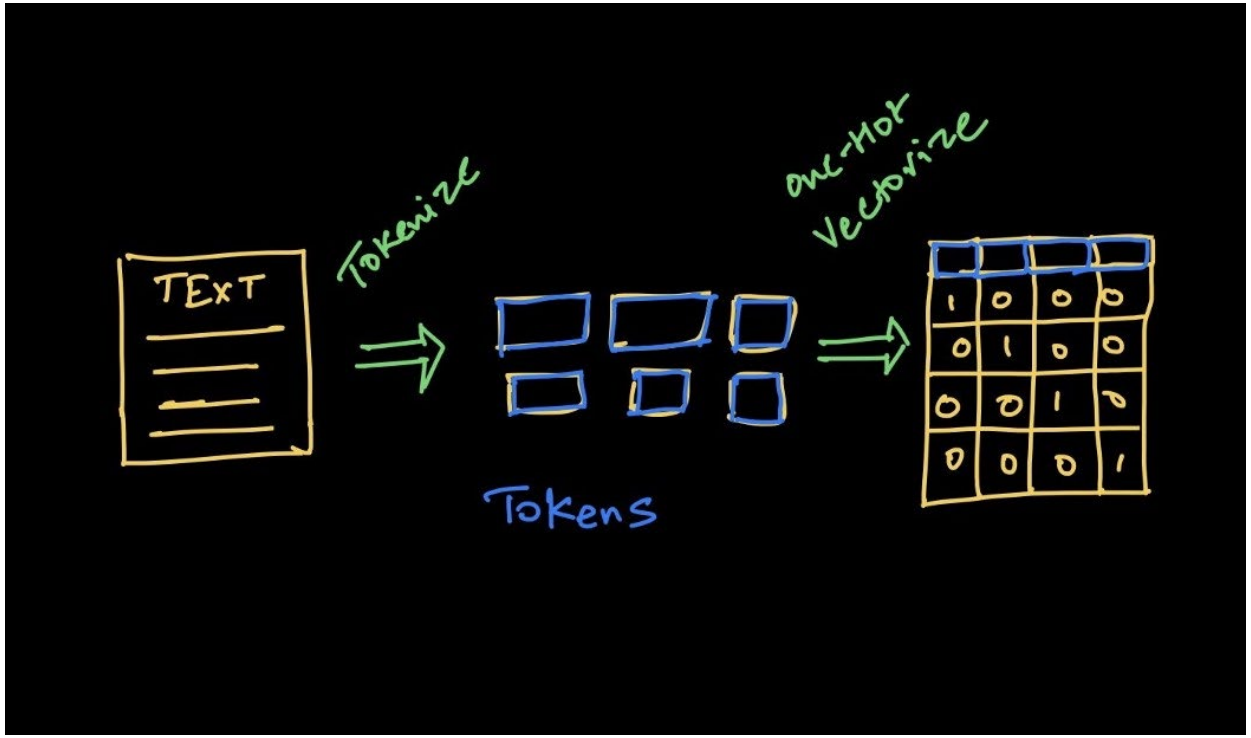
DAVIAN

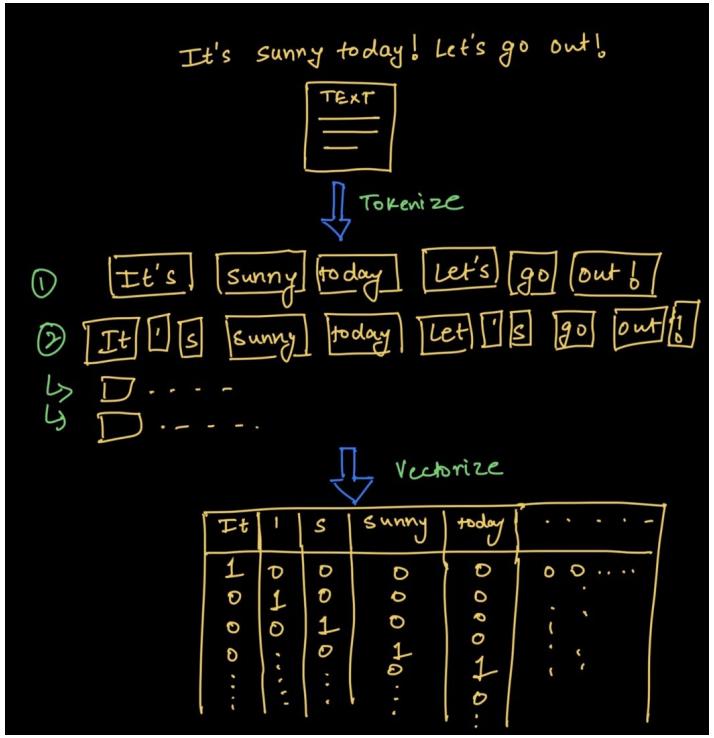
NLP Pipeline

- Data Collection
- Preprocessing
  - Pre-tokenization
  - Tokenization
- Modeling
- Training
  - Training Monitoring
- Evaluation
- Deployment
- Performance Monitoring

Tokenization

- Tokenization is a way of separating a piece of text into smaller units called tokens
- Based on the way of tokenization, our model recognize a sequence





https://towardsdatascience.com/the-evolution-of-tokenization-in-nlp-byte-pair-encoding-in-nlp-d7621b9c1186

Tokenizer

- Tokenizers
  - Word based Tokenizer
  - Character based Tokenizer
  - Subword based Tokenizer
- There can be a pre tokenization step (e.g., space tokenization, Moses, Spacy, ftfy)

Tokenization	Tokenized Sequence
Raw Text	나랑 쇼핑하자.
CV (4.1)	ㄴ/ ㅏ/ㄹ/ ㅏ/ㅇ/★/ㅅ/ㅛ/ㅍ/ ㅣ/ㅇ/ㅎ/ ㅏ/ㅈ/ ㅏ/.
Syllable (4.2)	나/랑/★/쇼/핑/하/자/.
Morpheme (4.3)	나/랑/★/쇼핑/하/자/.
Subword (4.4)	_나랑/_쇼/핑하/자/.
Morpheme-aware Subword (4.5)	_나/_랑/★/_쇼/핑/_하/_자/_.
Word (4.6)	나랑/쇼핑하자/.

Table 1: An input sentence 나랑 쇼핑하자. 'Let's go shopping with me.' is differently tokenized depending on the various tokenization strategies. Slashes (/) are token separators.

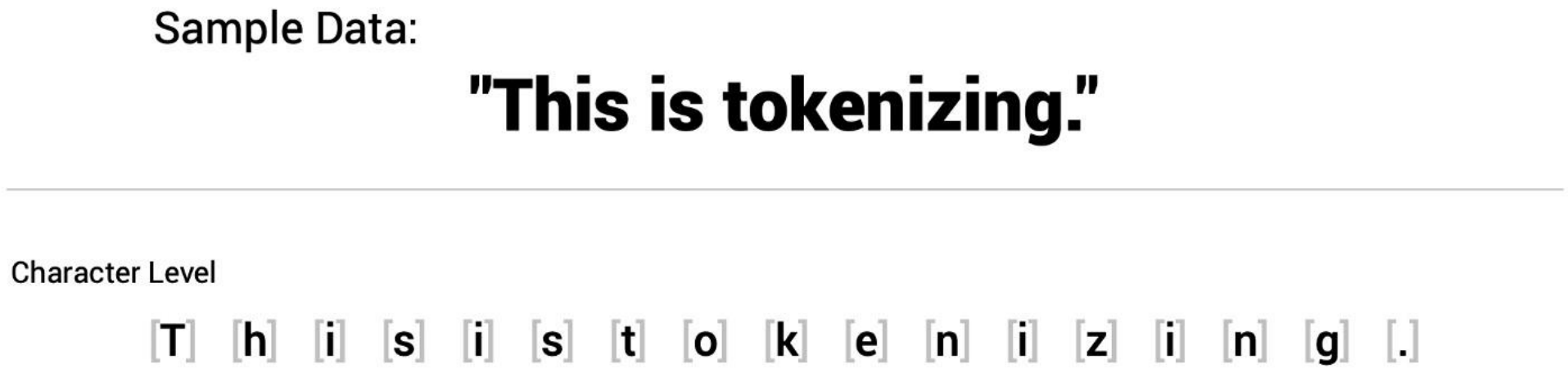
An Empirical Study of Tokenization Strategies for Various Korean NLP Tasks, AACL 2020, Park et al. (https://arxiv.org/pdf/2010.02534.pdf)

Word-based Tokenizer

- Input —['The devil is in the details']
- Output —[' The', 'devil', 'is', 'in', 'the', 'details']

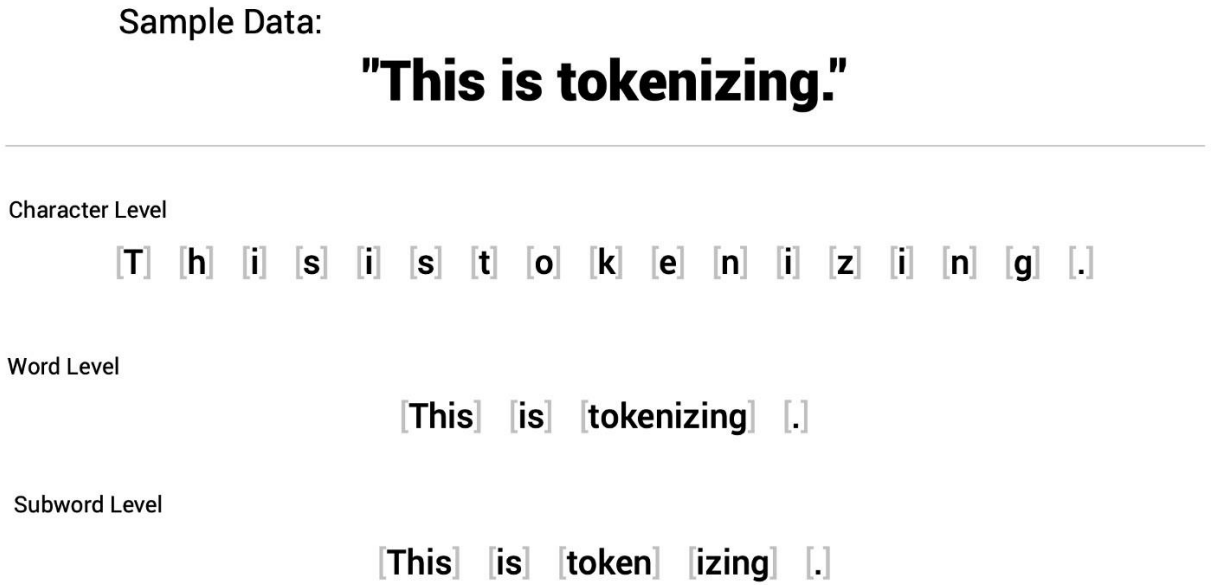
Character-based Tokenizer

- Input–[‘The devil is in the details’]
- Output–[‘T’, ‘h’, ‘e’, ‘d’, ‘e’, ‘v’, ‘i’, ‘l’, ‘i’, ‘s’, ‘i’, ‘n’, ‘t’, ‘h’, ‘e’, ‘d’, ‘e’, ‘t’, ‘a’, ‘i’, ‘l’, 's’]



Subword-based Tokenizer

- Input–['The devil is in the details']
- Output–['The', 'de', 'vil', 'is', 'in', 'the', 'de', 'tail', 's']



Pre-tokenization Recap

- Why should we set the pre-tokenization step?
  - Example
    - 이런 영화에 평점 1 점 날리는 것들은 영화를 볼 줄 모르는 것들이다
    - 어렸을때부터 mbc 드라마를봤었는데이제더이상은안봐야할것같다 막장스토리출생의비밀 뒤바뀐운명자주등장하는대기업그속에서그려지는억지설정도모자라이젠쌍둥이등장
    - I got up early. Then I went to DAVIAN Lab.
  - Korean
    - 조사
  - English
    - Case, uncased

Word vs Character vs Subword

- Word
  - Out-Of-Vocabulary (OOV)

- Character
  - Longsequence([‘T’, ‘h’, ‘e’, ‘d’, ‘e’, ‘v’, ‘i’, ‘l’, ‘i’, ‘s’, ‘i’, ‘n’, ‘t’, ‘h’, ‘e’, ‘d’, ‘e’, ‘t’, ‘a’, ‘i’, ‘l’, ‘s’])
  - Low performance

- Subword
  - Shorter sequence
  - Higher performance
  - (Almost) Free fromOOV

Subword-based Algorithms

- Byte-pair Encoding (BPE)
  - Statistical method(e.g., GPT)
- WordPiece
  - Merge a pair that maximizes the likelihood of the training data once added to the vocab (e.g., BERT, DistillBERT, ELECTRA)
  - E.g., p('ug')/p('u' then 'g')
- Unigram
  - Starts from pretokenizedwords and the most common substrings then trims
- SentencePiece
  - Solve the problem that all the above methods are using space to separate words before tokenization by dealing with the space as a token (e.g., ALBERT, XLNet, T5)

Byte-pair Encoding

- Byte pair Encoding (BPE)
  - BPE ensures that the most common words are represented in the vocabulary as a single token while the rare words are broken down into two or more subword tokens and this is in agreement with what a subword-based tokenization algorithm does.
  - Statistical method (e.g., GPT)

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

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}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

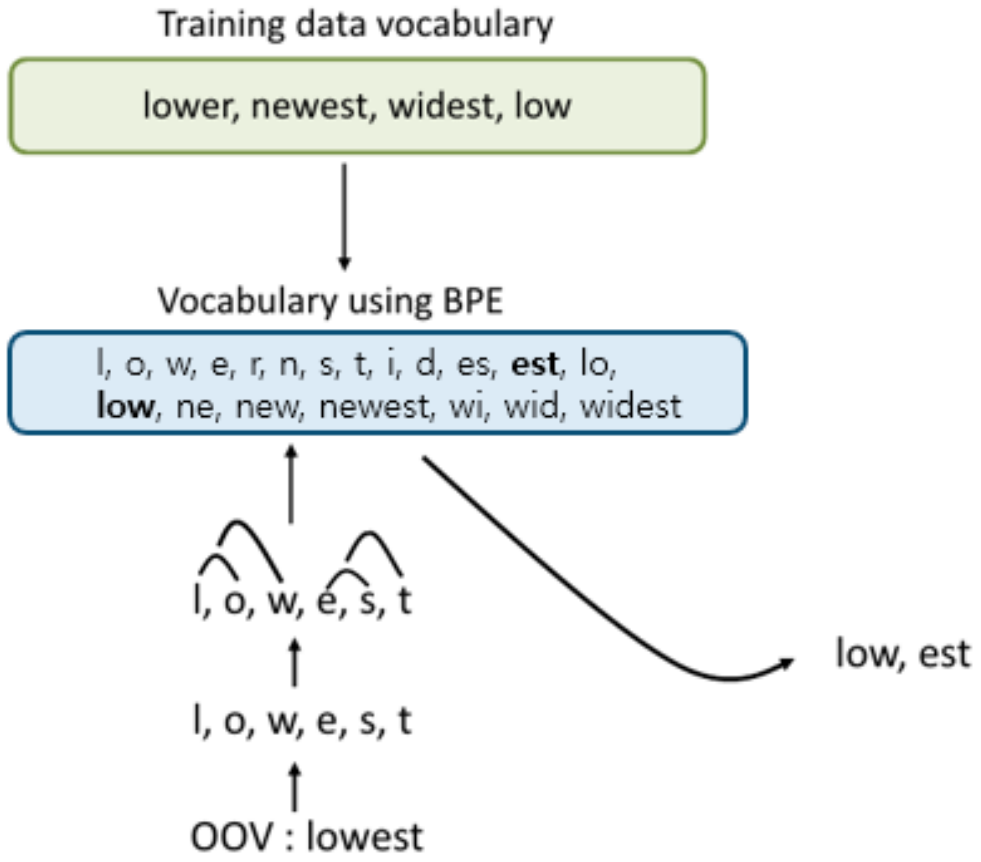
https://towardsdatascience.com/byte-pair-encoding-subword-based-tokenization-algorithm-77828a70bee0

Byte-pair Encoding Example

- aaabdaaabac
  - Z = aa
    - The byte pair **aa** occurs most often, so we will replace it with **Z** as **Z** does not occur in our data.
  - ZabdZabac
  - Y = ab
  - X = ZY # recursive
  - XdXac where X = ZY, Y=ab, Z = aa
    - It cannot be further compressed as there are no byte pairs appearing more than once.

      We decompress the data by performing replacements in reverse order.

Byte-pair Encoding Example



https://wikidocs.net/22592

Issue

- Token after space
  - Example
    - Knowledge graph is one of the external resource in language generation task.
  - Substring match problem
    - Example
      - Sulfhydration
      - 학교에