Integer Encoding and Padding

GYUCHEOL LEE

Contents

- 정수 인코딩(Integer Encoding)
- 패딩(Padding)

정수 인코딩(Integer Encoding)

- 정수 인코딩(Integer Encoding)이란 단어 토큰화 또는 형태소 토큰화를 수행했다면 각 단어에 고유한 정수를 부여하는 과정을 의미합니다.
- 중복이 허용되지 않는 모든 단어들의 집합을 만듭니다.
- 이를 단어 집합(Vocabulary)라고 하며 이를 기반으로 문서를 정수로 인코딩합니다.
- 아래는 정수 인코딩(integer Encoding)을 진행한 결과입니다.

In [1]:
```
import numpy as np
import torch
from torchtext.vocab import build_vocab_from_iterator
from torchtext.data.utils import get_tokenizer
from collections import Counter
```

In [2]:
```
def integer_encoding(text):
    tokenizer = get_tokenizer("basic_english")
    word_counts = Counter(tokenizer(text))
    vocab = build_vocab_from_iterator([word_counts.keys()])
    word_to_int = {word: idx for idx, word in enumerate(vocab.stoi.keys(), 1)}

    print("인코딩 결과:")
    for word, code in word_to_int.items():
        print(f"{word}: {code}")

    encoded_text = [word_to_int[word] for word in tokenizer(text)]

    print("인코딩된 텍스트:")
    print(encoded_text)


text1 = "딥러닝은 인공지능의 한 분야로, 기계학습의 한 방법론입니다."
text2 = "딥러닝은 심층 신경망을 통해 복잡한 데이터 표현을 학습할 수 있습니다."
print(integer_encoding(text1))
print(integer_encoding(text2))
```

```
1lines [00:00, 22192.08lines/s]

인코딩 결과:
<unk>: 1
<pad>: 2
.: 3
.: 4
기계학습의: 5
딥러닝은: 6
방법론입니다: 7
분야로: 8
인공지능의: 9
한: 10
인코딩된 텍스트:
[6, 9, 10, 8, 3, 5, 10, 7, 4]
None
1lines [00:00, 37117.73lines/s]

인코딩 결과:
<unk>: 1
<pad>: 2
.: 3
데이터: 4
딥러닝은: 5
복잡한: 6
수: 7
신경망을: 8
심층: 9
있습니다: 10
통해: 11
표현을: 12
학습할: 13
인코딩된 텍스트:
[5, 9, 8, 11, 6, 4, 12, 13, 7, 10, 3]
None
```

패딩(Padding)

- 모든 문장에 대해서 정수 인코딩(Integer Encoding)을 수행하였을 때 길이는 서로 다를 수 있습니다.
- 이때 가상의 단어(0)를 추가하여 길이를 맞춰줍니다.
- 이렇게 하면 컴퓨터가 이를 병렬 연산할 수 있습니다.
- 아래는 정수 인코딩(Integer Encoding)과 패딩(Padding)을 진행한 결과입니다.

In [3]:
```
def integer_encoding(text):
    tokenizer = get_tokenizer("basic_english")
    word_counts = Counter(tokenizer(text))
    vocab = build_vocab_from_iterator([word_counts.keys()])
    word_to_int = {word: idx for idx, word in enumerate(vocab.stoi.keys(), 1)}

    encoded_text = [word_to_int[word] for word in tokenizer(text)]

    return encoded_text

def pad_sequence(sequence, max_length, padding_value=0):
    if len(sequence) >= max_length:
        return sequence[:max_length]
    else:
        return sequence + [padding_value] * (max_length - len(sequence))

text1 = "딥러닝은 인공지능의 한 분야로, 기계학습의 한 방법론입니다."
text2 = "딥러닝은 심층 신경망을 통해 복잡한 데이터 표현을 학습할 수 있습니다."

encoded_text1 = integer_encoding(text1)
encoded_text2 = integer_encoding(text2)

max_length = max(len(encoded_text1), len(encoded_text2))

padded_text1 = pad_sequence(encoded_text1, max_length)
padded_text2 = pad_sequence(encoded_text2, max_length)

print(padded_text1)
print(padded_text2)
```

```
1lines [00:00, 47127.01lines/s]
1lines [00:00, 49344.75lines/s]
```

```
[6, 9, 10, 8, 3, 5, 10, 7, 4, 0, 0]
[5, 9, 8, 11, 6, 4, 12, 13, 7, 10, 3]
```

In [4]:
```
matrix = np.array([padded_text1, padded_text2])
print(matrix)
```

```
[[ 6  9 10  8  3  5 10  7  4  0  0]
 [ 5  9  8 11  6  4 12 13  7 10  3]]
```