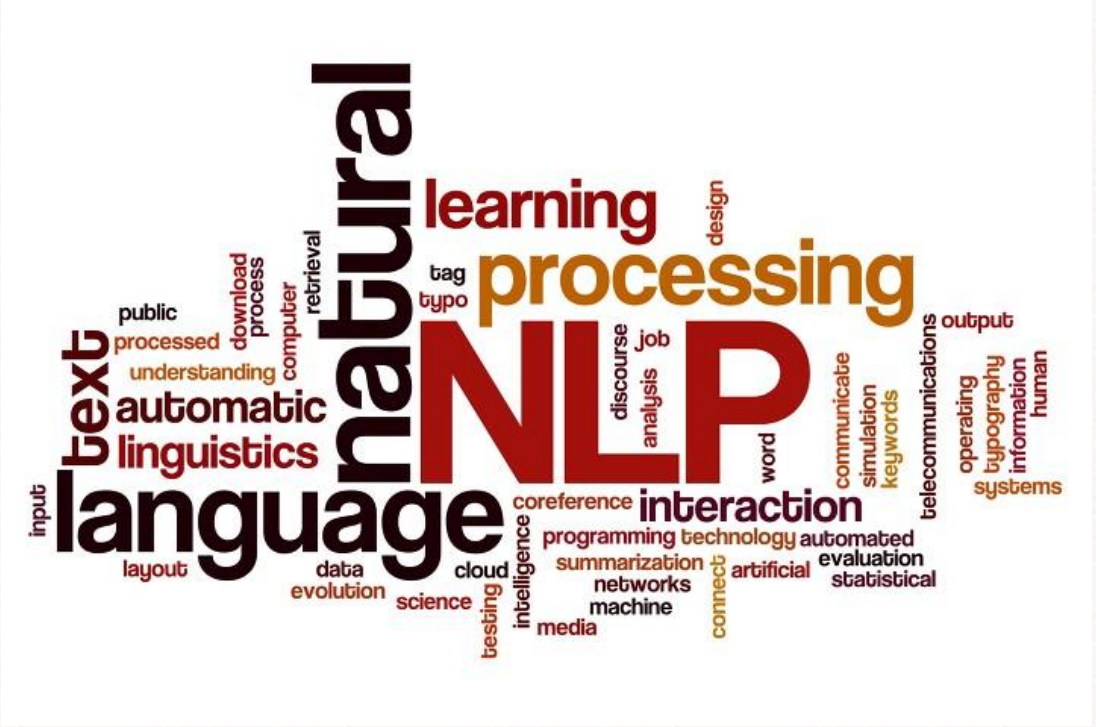
순환 신경망으로 IMDB 리뷰 분류하기

비타민 10기 정규세션 6주차

2조 이주석, 이두진, 임홍주

목차



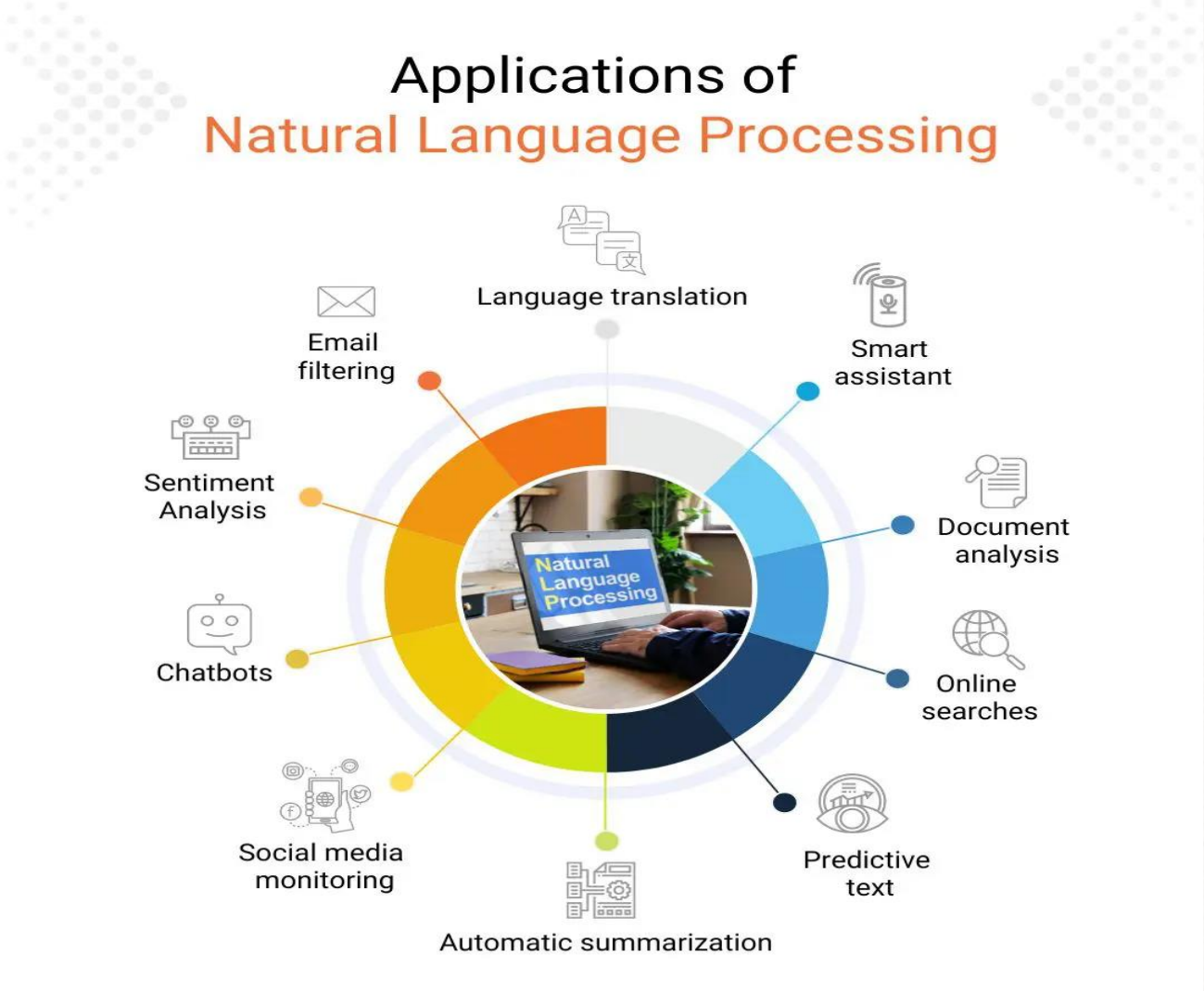
1. 자연어 처리란?

자연어 처리(NLP)란?

자연어란?
컴퓨터에서 사용하는 프로그램 작성 언어 또는 기계어와 구분하기 위해 인간이 일상생활에서 의사소통을 위해 사용하는 언어를 가리키는 말
[네이버 지식백과] 자연어

자연어 처리(Natural Language Processing)은 인공지능의 한 분야로써 머신러닝을 사용하여 텍스트와 데이터를 처리하고 해석하는 것.
텍스트의 구조와 의미를 파악하는 것이 주요 목적

자연어 처리(NLP)란?


Applications of
Natural Language Processing
Language translation
Email filtering
Smart assistant
Sentiment Analysis
Document analysis
Natural Language Processing
Chatbots
Online searches
Social media monitoring
Predictive text
Automatic summarization

2. 자연어 처리 관련 용어

자연어 처리 관련 용어

Corpus(말뭉치) : NLP에서 의미하는 텍스트 데이터. 일반적으로 원시 텍스트와 연관된 메타데이터를 포함

Token(토큰) : 문법적으로 더 이상 나눌 수 없는 요소

Tokenization(토큰화) : 주어진 Corpus(텍스트)에서 토큰 단위로 나누는 작업

N-gram(N-그램) : 텍스트에 있는 고정 길이(N)의 연속된 토큰 시퀀스

Uni-gram(유니그램) : N-gram 중 토큰이 하나(N=1)로 이루어진 것

Bi-gram(바이그램) : N-gram 중 토큰이 두 개(N=2)로 이루어진 것

Lemma(표제어) : 단어의 기본형

Lemmatization(표제어 추출) : 토큰을 표제어로 바꾸어 벡터 표현의 차원을 줄이는 방법

3. 임베딩

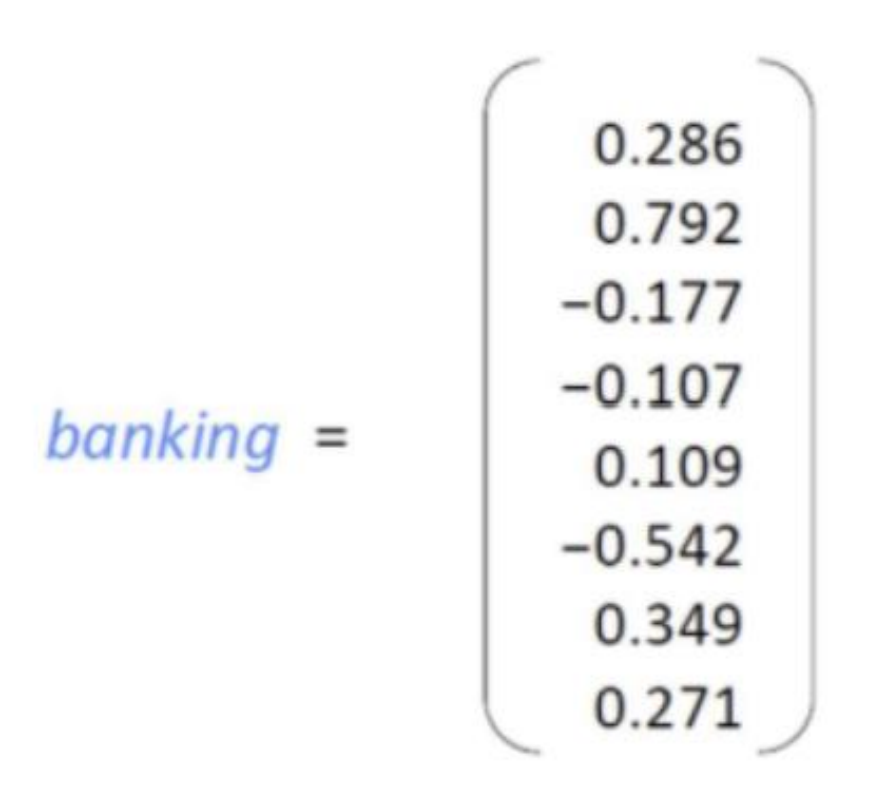
임베딩(Embedding)

사람이 쓰는 자연어를 기계가 이해할 수 있는 숫자의 나열인 벡터로 바꾼 결과 혹은 그 과정 전체

‘banking’이라는 단어를 컴퓨터가 이해할 수 있을까요?

임베딩(Embedding)

당연히 이해할 수 없겠죠? 컴퓨터가 이해하게 하려면 어떻게 해야 할까요?



숫자 벡터로 표현하여
컴퓨터가 이해 가능!

임베딩(Embedding)

임베딩 방법

1. One-Hot Encoding(원-핫 인코딩)

단어	단어 인덱스	원-핫 벡터
you	0	[1, 0, 0, 0, 0, 0, 0]
say	1	[0, 1, 0, 0, 0, 0, 0]
goodbye	2	[0, 0, 1, 0, 0, 0, 0]
and	3	[0, 0, 0, 1, 0, 0, 0]
I	4	[0, 0, 0, 0, 1, 0, 0]
say	5	[0, 0, 0, 0, 0, 1, 0]
hello	6	[0, 0, 0, 0, 0, 0, 1]

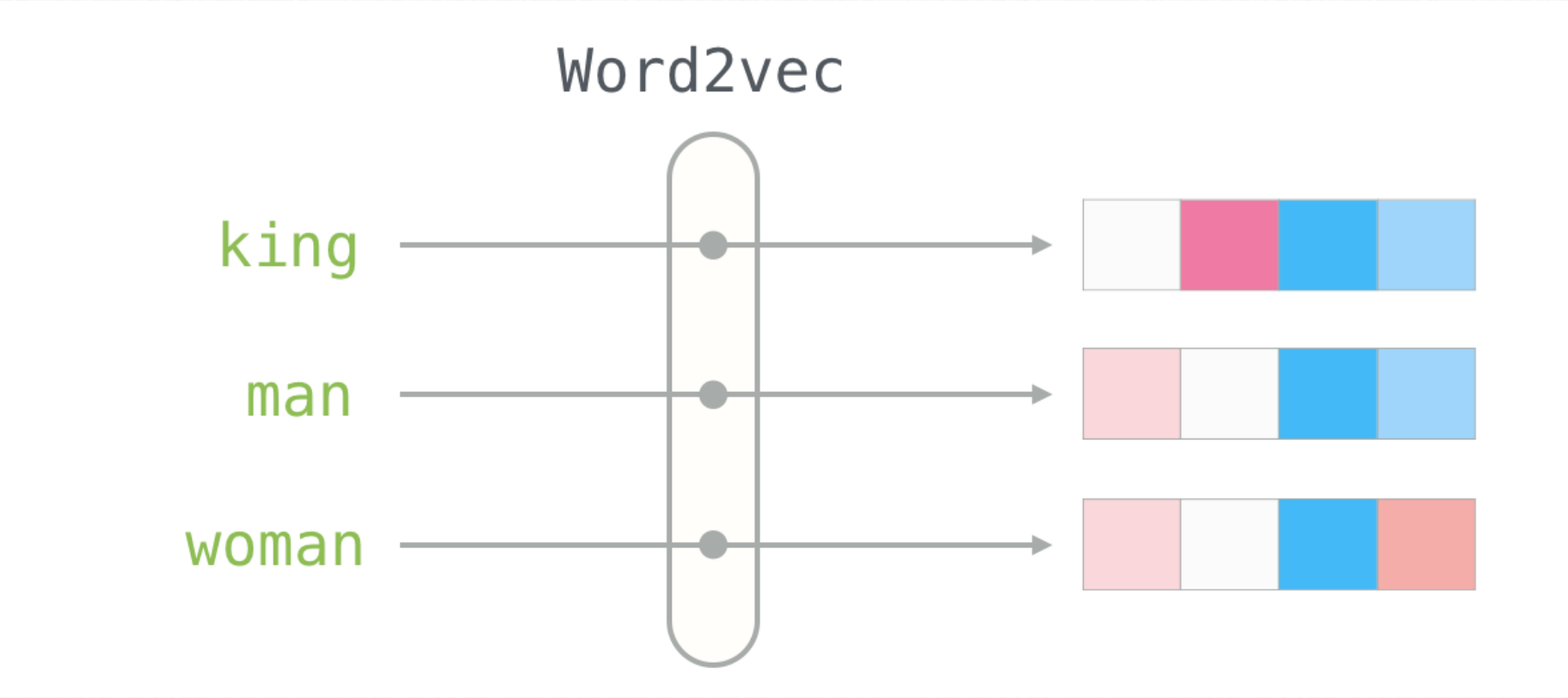
장점 : 간편

단점 : 단어 개수에 따라 벡터 개수가 늘어남.
효율성 감소. 계산량 증가

임베딩(Embedding)

임베딩 방법
2. Word2Vec

Neural Net Language Model을 기반으로 대량의 문서를 벡터 공간에 고수준의 의미를 지닌 벡터를 가지도록 하는 모델
단어의 의미는 주변 단어에 의해 형성.

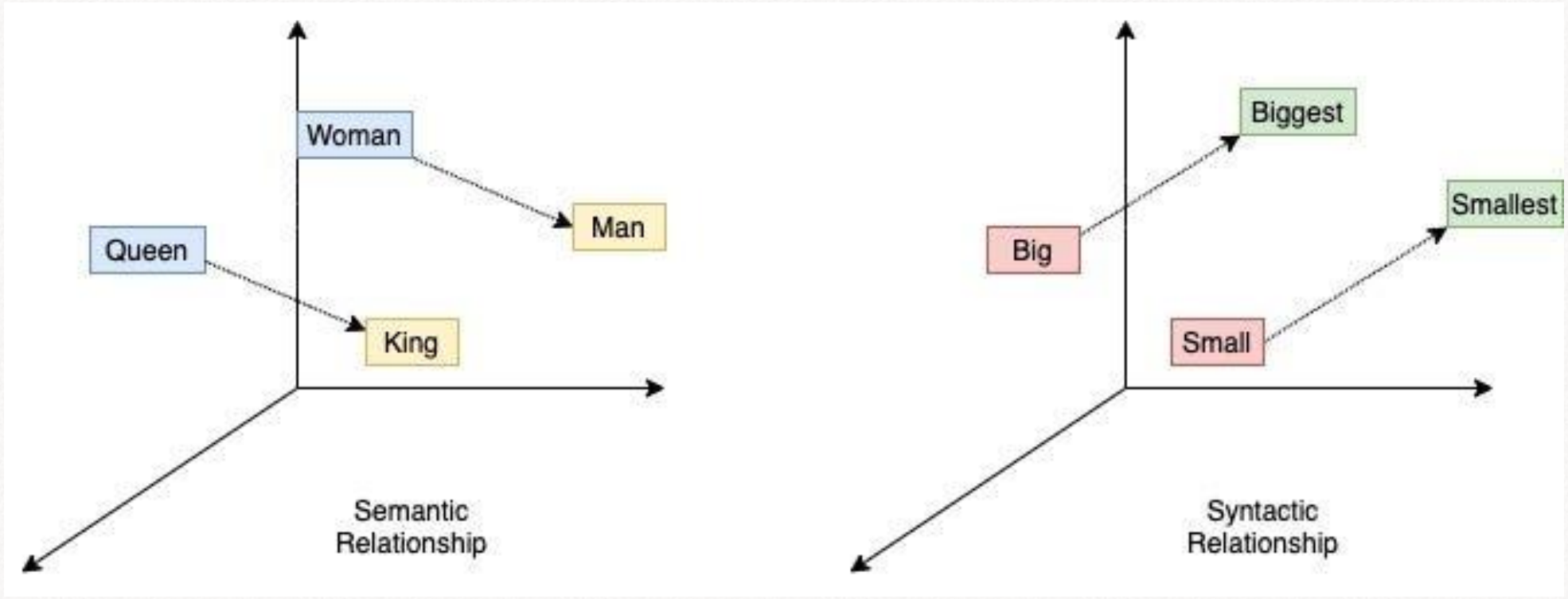


임베딩(Embedding)

임베딩 방법

2. Word2Vec

참고) Semantic : 문장의 의미론적
Syntactic : 문장의 통사론적(문법적)

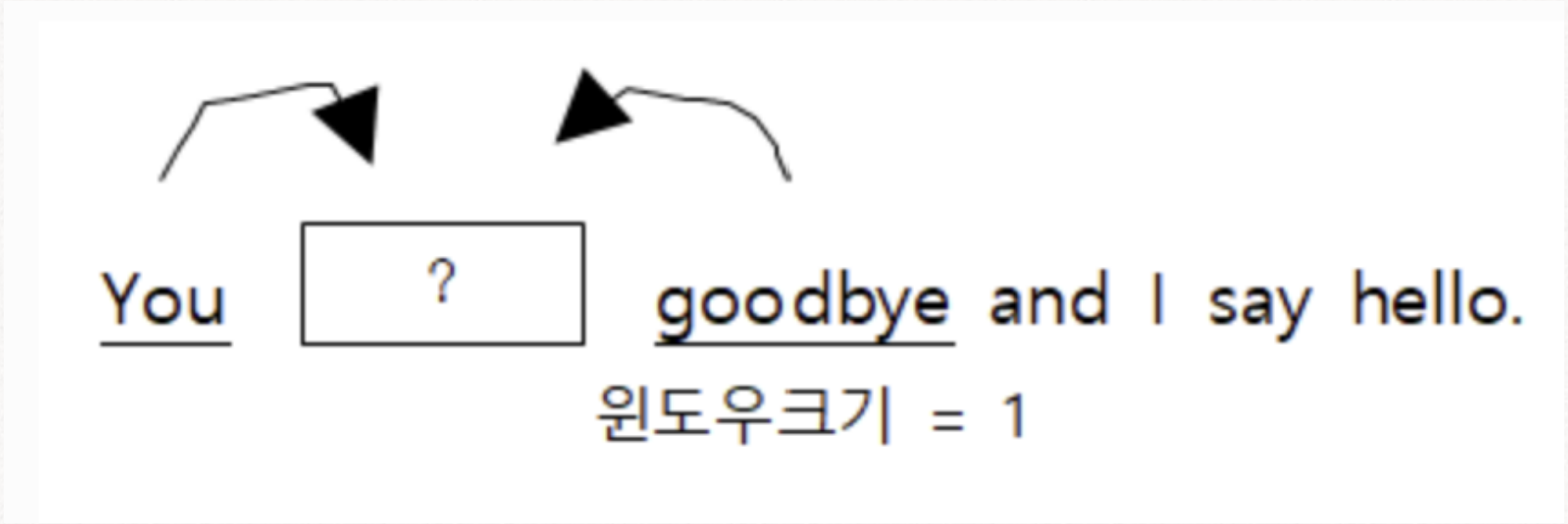


임베딩(Embedding)

2. Word2Vec

(1) CBOW(Continuous Bag of Words)

Context Word(주변 단어)로부터 Center Word(중심 단어)를 추측하는 방식



임베딩(Embedding)

2. Word2Vec

(1) CBOW(Continuous Bag of Words)

Context Word(주변 단어)로부터 Center Word(중심 단어)를 추측하는 방식

말뭉치	주변단어	중심단어
you say goodbye and I say hello.	you, goodbye	say
you say goodbye and I say hello.	say, and	goodbye
you say goodbye and I say hello.	goodbye, I	and
you say goodbye and I say hello.	and, say	I
you say goodbye and I say hello.	I, hello	say
you say goodbye and I say hello.	say, .	hello

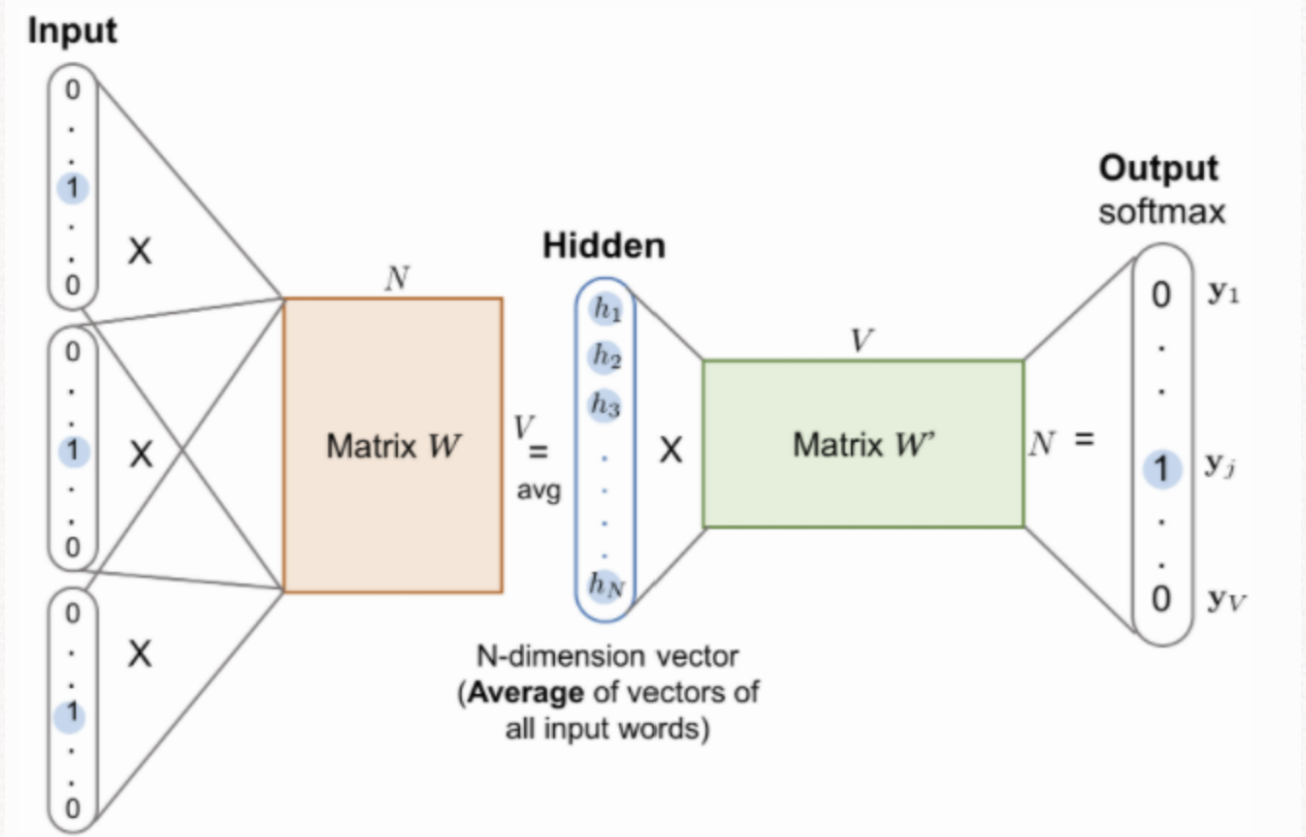
Widow? : Center Word를 예측하기 위해서 앞 뒤로 몇 개의 단어를 볼지를 정했다면 그 개수를 뜻함
Sliding Window? : Window를 계속 움직여서 단어 선택을 바꿔가며 학습을 위한 데이터셋을 만드는 것

임베딩(Embedding)

2. Word2Vec

(1) CBOW(Continuous Bag of Words)

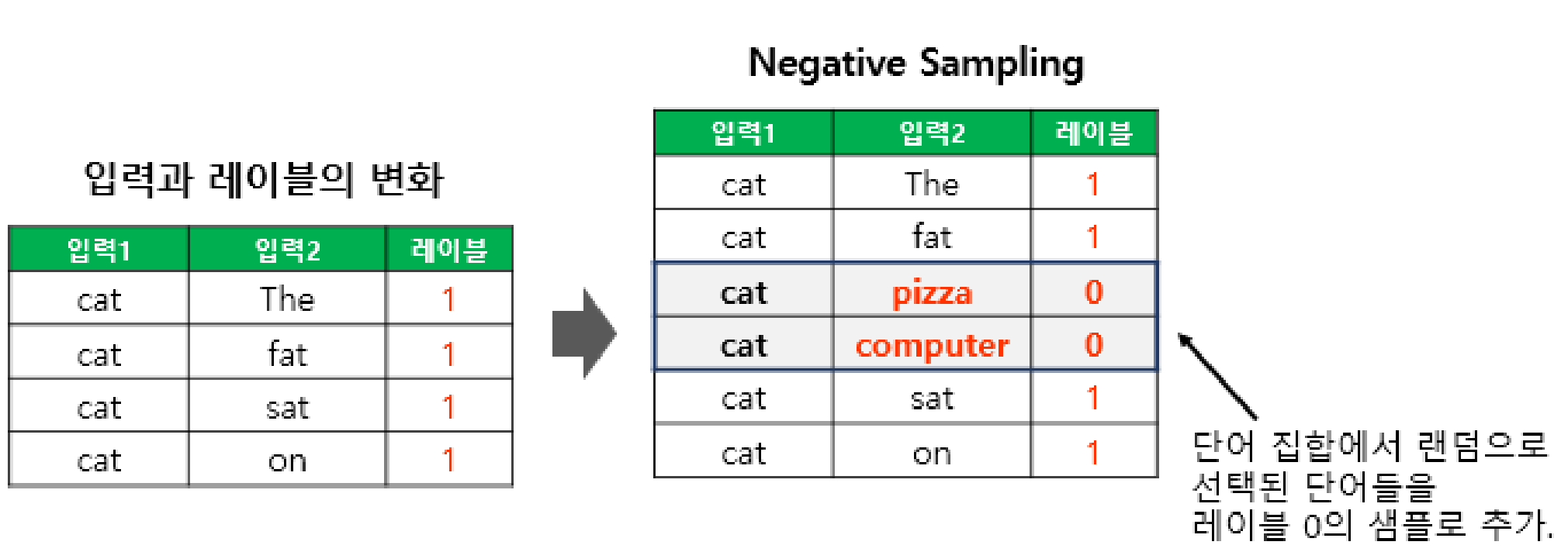
Context Word(주변 단어)로부터 Center Word(중심 단어)를 추측하는 방식



임베딩(Embedding)

2. Word2Vec

(참고) Negative Sampling https://arxiv.org/abs/1310.4546

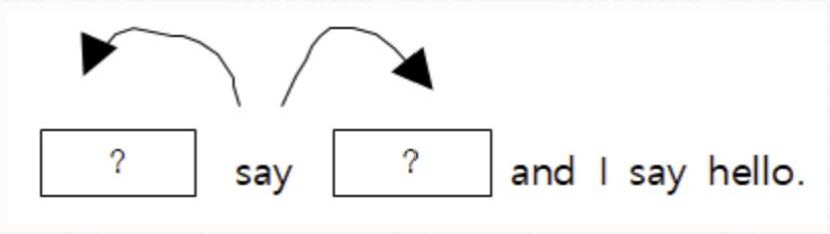


임베딩(Embedding)

2. Word2Vec

(1) Skip-Gram

Center Word(중심 단어)로부터 Context Word(주변 단어)를 추측하는 방식



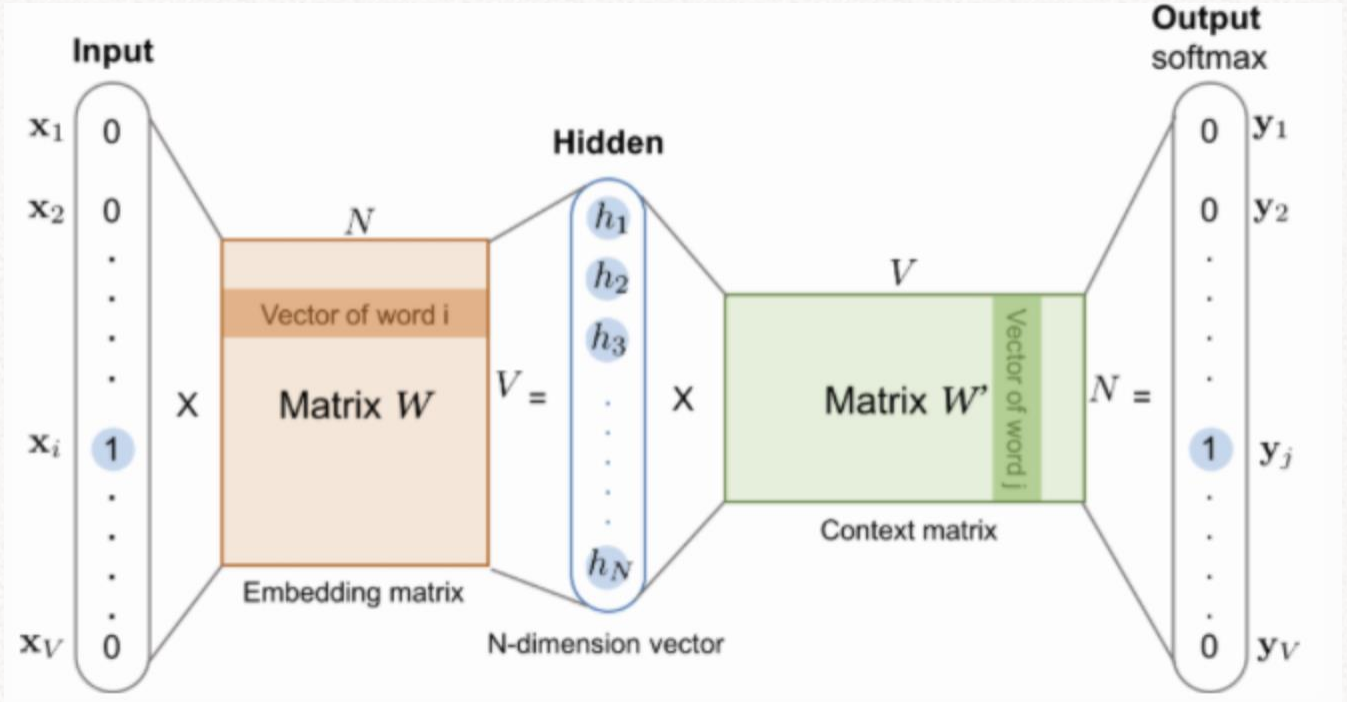
말뭉치	주변단어	중심단어
you say goodbye and I say hello.	you, goodbye	say
you say goodbye and I say hello.	say, and	goodbye
you say goodbye and I say hello.	goodbye, I	and
you say goodbye and I say hello.	and, say	I
you say goodbye and I say hello.	I, hello	say
you say goodbye and I say hello.	say, .	hello

임베딩(Embedding)

2. Word2Vec

(2) Skip-Gram

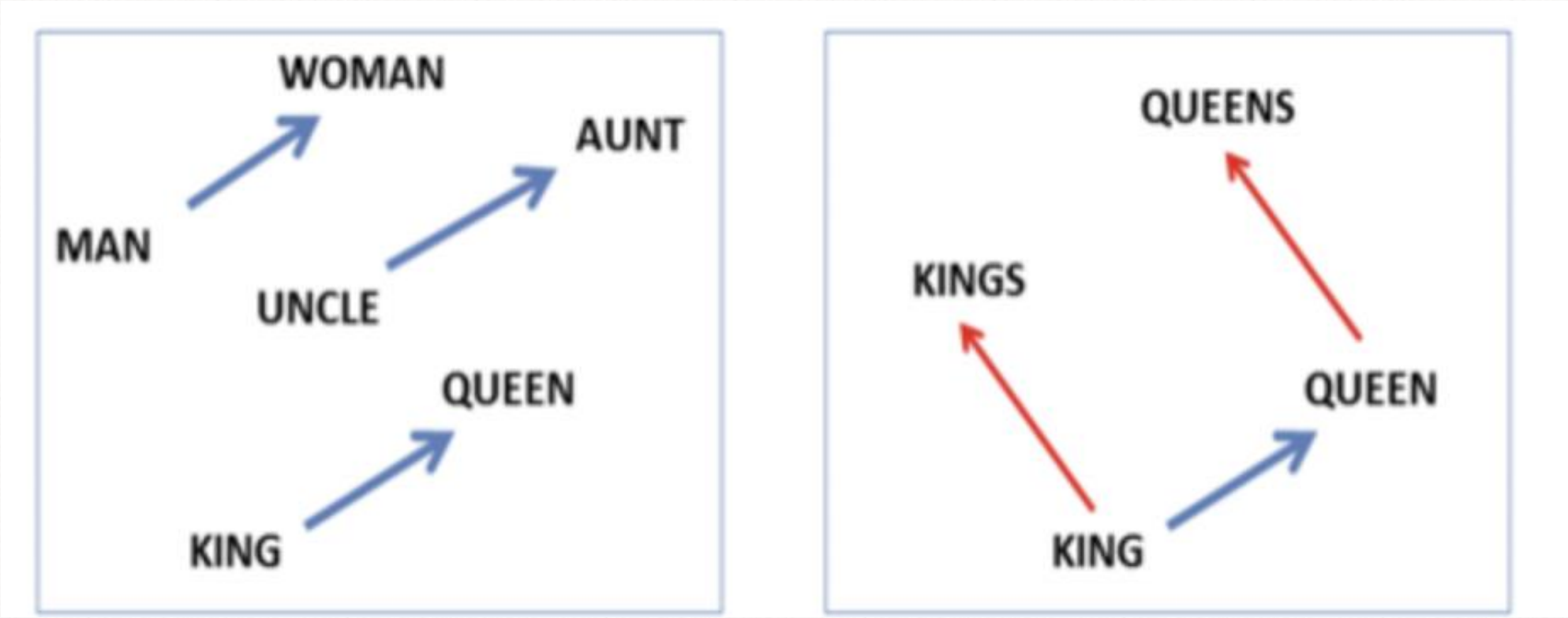
Center Word(중심 단어)로부터 Context Word(주변 단어)를 추측하는 방식



학습시간은 CBOW가 빠르지만 Corpus가 커질수록 Skip-Gram 성능이 더 뛰어나다고 알려져 있음.

임베딩(Embedding)

단어 사이의 유사성뿐만 아니라, 패턴도 학습 가능



4. 패딩

패딩(Padding)

길이가 다른 문장들의 길이를 같게 맞춰주어 행렬로 한 번에 연산할 수 있게 하는 것

1	2	3	4	5	6	7
I	Love	My	Dog			
I	Love	My	Cat			
You	Love	My	dog!			
Do	you	think	my	dog	is	amazing?

왜?
딥러닝에서는 계산 속도가 매우 중요.
행렬로 한 번에 계산하지 못한다면 계산 속도 매우 느려짐
⇒ 따라서 병렬연산을 위해 문장의 길이를 맞춰주어 빠르고 간편한 계산 가능

보통 앞에 0을 붙이거나(pre-padding), 뒤에 0을 붙임(post-padding)
둘에는 차이가 없으나, 일관성 있는 패딩 방법을 사용해야 함

IMDB 데이터 설명 및 전처리

IMDB 데이터 설명 및 전처리

미국의 영화 정보 사이트
수만 개의 영화에 대한 다양한 리뷰가 존재
대표적인 감성 분석 데이터셋
(참고) http://ai.stanford.edu/~amaas/papers/wvSent_acl2011.pdf

	review	sentiment
0	My family and I normally do not watch local mo...	1
1	Believe it or not, this was at one time the wo...	0
2	After some internet surfing, I found the "Home...	0
3	One of the most unheralded great works of anim...	1
4	It was the Sixties, and anyone with long hair ...	0

IMDB 데이터 설명 및 전처리

Field 속성을을 이용하여 데이터셋을 간단히 불러올 수 있음.(최신 버전에서는 depricated)

텐서로 변환할 지시사항과 함께 정의하는 것이라 되어있습니다. Field는 텐서로 표현 될 수 있는 텍스트 데이터 타입을 처리

sequential : 데이터가 순차적인 정보를 갖고 있는지 명시(True)

batch_first :

lower : text 데이터를 다 소문자로 변환. 소문자 변환은 왜?

LABEL 객체에서 sequential = False인 이유? => LABEL은 특정 영화에 대한 Review가 긍정인지, 부정인지만 나타내어주기 때문에 순서 정보가 포함되어 있지 않기 때문

```
[5]  TEXT = Field(sequential = True, batch_first = True, lower = True)
     LABEL = Field(sequential = False, batch_first = True)
```

IMDB 데이터 설명 및 전처리

splits method를 사용하여, 앞에서 정의해준 TEXT와 LABEL 객체를 이용하여 간편히 train 및 test 데이터셋을 불러올 수 있음.

```
[6]  trainset, testset = torchtext.legacy.datasets.IMDB.splits(TEXT, LABEL)
```

```
downloading aclImdb_v1.tar.gz
aclImdb_v1.tar.gz: 100%|██████████| 84.1M/84.1M [00:08<00:00, 9.48MB/s]
```

데이터셋 알아보기

text와 label로 구성되어 있음을 확인할 수 있다.

```
[7]  print('trainset의 구성 요소 출력 : ', trainset.fields)
```

```
trainset의 구성 요소 출력 :  {'text': <torchtext.legacy.data.field.Field object at 0x7fcddcd74940>, 'label': <torchtext.legacy.data.field.Field
```

```
[8]  print('testset의 구성 요소 출력 : ', testset.fields)
```

```
testset의 구성 요소 출력 :  {'text': <torchtext.legacy.data.field.Field object at 0x7fcddcd74940>, 'label': <torchtext.legacy.data.field.Field o
```

IMDB 데이터 설명 및 전처리

vars([object])

모듈, 클래스, 인스턴스 또는 **dict** 어트리뷰트가 있는 다른 객체의 **dict** 어트리뷰트를 돌려줍니다.

text에는 리뷰에 쓰인 단어가 리스트로, label에는 긍정인지 부정인지를 단일 문자형으로 갖고 있습니다. 'pos' : positive(긍정), 'neg' : negative(부정)입니다.

```
[9]  print(vars(trainset[0])) # 1번째 trainset 데이터
```

```
{'text': ['this', 'is', 'a', 'really', 'fun,', 'breezy,', 'light', 'hearted', 'romantic', 'comedy.', 'you', 'cannot', 'go', 'wrong', 'with', 'meg
```

IMDB 데이터 설명 및 전처리

데이터 전처리

+ 코드 + 텍스트

build_vocab : 단어집합 생성

min_freq는 학습 데이터에서 최소 5번 이상 등장한 단어만을 단어 집합에 추가하겠다는 의미. 이때 학습 데이터에서 5번 미만으로 등장한 단어는 Unknown이라는 의미에서 unk이라는 토큰으로 대체

```
[10] TEXT.build_vocab(trainset, min_freq=5)
     LABEL.build_vocab(trainset)
```

단어 집합의 크기가 46159 : 중복을 제외한 단어의 개수가 46159개이다. 여기에는 unk도 포함

클래스는 긍정 또는 부정

```
[15] vocab_size = len(TEXT.vocab)
     n_classes = 2
     print('단어 집합의 크기 : {}'.format(vocab_size))
     print('클래스의 개수 : {}'.format(n_classes))
```

```
단어 집합의 크기 : 46159
클래스의 개수 : 2
```

IMDB 데이터 설명 및 전처리

stoi로 단어와 각 단어의 정수 인덱스가 저장되어져 있는 딕셔너리 객체에 접근

unk가 0번이고, 패딩이 1번, 그 뒤로 단어 사전에 포함된 단어가 고유한 인덱스를 갖게 된 것을 확인 가능

```
[16] print(TEXT.vocab.stoi)
```

```
defaultdict(<bound method Vocab._default_unk_index of <torchtext.legacy.vocab.Vocab object at 0x7fcddcd749d0>>, {'<unk>': 0, '<pad>': 1, 'the': 2
```

+ 코드 + 텍스트

```
print(TEXT.vocab.stoi)
```

```
e': 407, 'war': 408, 'classic': 409, 'father': 410, '.': 411, 'production': 412, 'house': 413, 'home': 414, 'camera': 415, 'else': 416, 'wanted':
```

IMDB 데이터 설명 및 전처리

https://gmihaila.github.io/tutorial_notebooks/pytorchtext_bucketiterator/ : BucketIterator 공부하실 분은 여기로!

Defines an iterator that batches examples of similar lengths together.

Minimizes amount of padding needed while producing freshly shuffled batches for each new epoch. See pool for the bucketing procedure used.

모든 텍스트를 배치 처리하는 것을 지원하고, 단어를 인덱스 번호로 대체

---

shuffle : 에포크마다 데이터셋 섞음

device : 배치를 Load할 device 설정

repeat : Repeat the iterator for multiple epochs.

sort_key : A key to use for sorting examples in order to batch together examples with similar lengths and minimize padding. The sort_key provided to the Iterator constructor overrides the sort_key attribute of the Dataset, or defers to it if None.

sort : Sort all examples in data using `sort_key`.

sort_within_batch=True : Use `sort_key` to sort examples in each batch.

```
[18]  batch_size = 10

      train_iter, val_iter, test_iter = BucketIterator.splits(
              (trainset, valset, testset),
              batch_size = batch_size,
              repeat = False,
              sort_key=lambda x: len(x.text),
              sort=False,
              sort_within_batch=True,
              shuffle=True)
```

신경망 구현 및 훈련

신경망 구현 및 훈련

모형 구축

감성분석을 위한 Vanilla RNN 코드

self.embedding : torchtext에서 자동으로 임베딩을 만들어주는 도구. padding_idx = 1로 설정하여 자동으로 패딩 값을 1로 만들어준다.

self.rnn : Vanilla RNN을 구현해줌.

RNN으로 학습한 후, 선형 층을 쌓고 활성화 함수로 시그모이드를 사용한다. 감성이 긍정인지, 부정인지 알아보는 이진분류이기 때문이다.

```
[29]  class SentimentClassifier(nn.Module):
          def __init__(self, vocab_size, embedding_dim, hidden_dim, num_layers, output_dim):
              super().__init__()

              self.embedding = nn.Embedding(vocab_size, embedding_dim, padding_idx = 1)
              self.rnn = nn.RNN(embedding_dim, hidden_dim, num_layers, batch_first=True)
              self.fc = nn.Linear(hidden_dim, output_dim)
              self.act = nn.Sigmoid()

          def forward(self, text):
              embedded = self.embedding(text)
              output, hidden = self.rnn(embedded)
              last_hidden = hidden[-1]
              linear = self.fc(last_hidden)
              out = self.act(linear)
              return out
```

신경망 구현 및 훈련

Hyperparameter, model, optimizer, criterion, metric defining

```
[33]  # Set up the model, loss function, and optimizer
      input_dim = vocab_size
      embedding_dim = 100
      hidden_dim = 256
      num_layers = 3
      output_dim = 1

      model = SentimentClassifier(input_dim, embedding_dim, hidden_dim, num_layers, output_dim)
      criterion = nn.BCELoss()
      optimizer = optim.Adam(model.parameters(), lr=1e-3)

      # Send the model and criterion to the device
      model = model.to(device)
      criterion = criterion.to(device)

      # Define the accuracy metric
      def accuracy(preds, y):
          rounded_preds = torch.round(torch.sigmoid(preds))
          correct = (rounded_preds == y).float()
          acc = correct.sum() / len(correct)
          return acc
```

신경망 구현 및 훈련

```
# Train the model
N_EPOCHS = 5

for epoch in range(N_EPOCHS):
    train_loss = 0
    train_acc = 0

    for batch in train_iter:
        optimizer.zero_grad()
        text, labels = batch.text.to(device), batch.label.to(device)
        labels = labels - 1 # labels가 [0, 1]이 아닌 [1, 2]로 코딩되었기 때문에 오류 발생
        labels = labels.type(torch.float32) # int를 float으로 변경
        preds = model(text).squeeze()
        loss = criterion(preds, labels)
        acc = accuracy(preds, labels)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        train_acc += acc.item()

    # Evaluate the model on the val set
    val_loss = 0
    val_acc = 0
```

신경망 구현 및 훈련

```
with torch.no_grad():
    for batch in val_iter:
        text, labels = batch.text.to(device), batch.label.to(device)
        labels = labels - 1 # labels가 [0, 1]이 아닌 [1, 2]로 코딩되었기 때문에 오류 발생
        labels = labels.type(torch.float32) # int를 float으로 변경
        preds = model(text).squeeze()
        loss = criterion(preds, labels)
        acc = accuracy(preds, labels)
        val_loss += loss.item()
        val_acc += acc.item()

# Print the epoch, training loss, training accuracy, val loss, and val accuracy
print(f'Epoch: {epoch+1:02}')
print(f'\tTrain Loss: {train_loss/len(train_iter):.3f} | Train Acc: {train_acc/len(train_iter):.2%}')
print(f'\t Val Loss: {val_loss/len(val_iter):.3f} |  Val Acc: {val_acc/len(val_iter):.2%}')
```

감사합니다!