Embedding Input in Pytorch

Reference

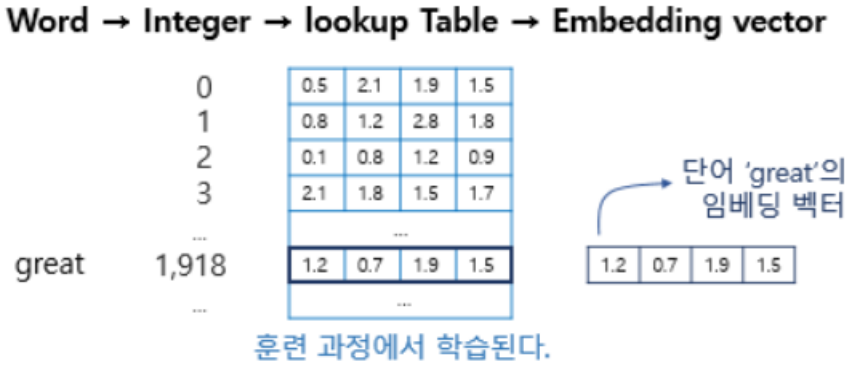
- pytorch로 시작하는 딥러닝 입문 (wikidocs)

자연어를 처리하는 딥러닝 모델은 단어를 있는 그대로 받지 않는다. 컴퓨터는 자연어를 바로 인식하지 못하기 때문에 이를 숫자로 구성된 벡터로 바꿔줘야 한다. 단어를 벡터로 바꾸는 방법은 Embedding이라고 불리며 이는 word2vec, glove, fasttext 등 하나의 학문으로 발전될 만큼 자연어 처리에서 중요한 분야이다. 자연어를 처리하는 딥러닝 모델은 입력으로 자연어를 임베딩한 벡터를 받는다. 우선 pytorch에서는 임베딩 벡터를 자체적으로 만들어주기도 하고, 사전에 학습된 벡터를 입력으로 넣을 수도 있다. 차례대로 살펴보자.

nn.Embedding()

pytorch에서 단어를 벡터로 바꿔주는 함수이다. 먼저 nn.Embedding()은 다음과 같이 사용한다.

어떤 단어 $\rightarrow$ 단어에 부여된 고유한 정수 값 $\rightarrow$ 임베딩 층 통과 $\rightarrow$ 밀집 벡터



임베딩 층을 통과한다는 것은 lookup table에서 대응되는 단어의 행을 찾는다는 뜻이다. 즉, 위 그림에서 great 단어가 1918를 부여받았다고 하자. lookup table에서 1918번째 행을 찾으면 그 행이 great의 4차원 임베딩 벡터가 되는 것이다. 여기서 lookup table은 backpropagation을 통해서 학습된다고 한다. 다시 말하면, lookup table은 corpus에 등장하는 unique 단어의 수 $\times$ 임베딩된 단어의 차원의 크기를 가지는 거대한 행렬이다. 위 그림을 보면, 단어에 대해서 one-hot vector을 생성하는 과정은 없다. 즉, pytorch에서는 one-hot vector가 필요 없고 각 단어를 정수로만 변환하면 된다. wikidocs의 간단한 예시를 살펴보자.

```
train_data = 'you need to know how to code'
word_set = set(train_data.split()) # 중복을 제거한 단어들의 집합인 단어 집합 생성.
vocab = {word: i+2 for i, word in enumerate(word_set)}  # 단어 집합의 각 단어에 고유한 정수 맵핑.
vocab['<unk>'] = 0
vocab['<pad>'] = 1
print(vocab)
```

```
{'code': 2, 'you': 3, 'know': 4, 'to': 5, 'need': 6, 'how': 7, '<unk>': 0,
'<pad>': 1}
```

위와 같이 각 단어를 정수로 mapping한다. 그리고 예시로 룩업 테이블을 만든다. 여기서는 각 정수에 대응되는 단어가 8개이고 벡터의 차원 수는 3으로 정하면 룩업 테이블의 크기는 8×3 이다.

```
# 단어 집합의 크기만큼의 행을 가지는 테이블 생성.
embedding_table = torch.FloatTensor([
                                [ 0.0,  0.0,  0.0],
                                [ 0.0,  0.0,  0.0],
                                [ 0.2,  0.9,  0.3],
                                [ 0.1,  0.5,  0.7],
                                [ 0.2,  0.1,  0.8],
                                [ 0.4,  0.1,  0.1],
                                [ 0.1,  0.8,  0.9],
                                [ 0.6,  0.1,  0.1]])
```

이제 임의의 문장 'you need to run'에 대해서 룩업 테이블을 통해 임베딩 벡터를 가져와보자.

```
# 임의의 샘플 문장
sample = 'you need to run'.split()
idxes=[]
# 각 단어를 정수로 변환
for word in sample:
  try:
    idxes.append(vocab[word])
  except KeyError: # 단어 집합에 없는 단어일 경우 <unk>로 대체된다.
    idxes.append(vocab['<unk>'])
idxes = torch.LongTensor(idxes)

# 룩업 테이블
lookup_result = embedding_table[idxes, :] # 각 정수를 인덱스로 임베딩 테이블에서 값을
가져온다.
print(lookup_result)
```

```
tensor([[0.1000, 0.5000, 0.7000],
        [0.1000, 0.8000, 0.9000],
        [0.4000, 0.1000, 0.1000],
        [0.0000, 0.0000, 0.0000]])
```

임베딩 벡터를 가져올 때, try, except 구문을 통해서 단어가 없는 경우에 대해, ''의 벡터를 가져오도록 했다.

이제 nn.Embedding()을 사용해보자. 우선 전처리는 동일한 과정을 거친다.

```
train_data = 'you need to know how to code'
word_set = set(train_data.split()) # 중복을 제거한 단어들의 집합인 단어 집합 생성.
vocab = {tkn: i+2 for i, tkn in enumerate(word_set)}  # 단어 집합의 각 단어에 고유
한 정수 맵핑.
vocab['<unk>'] = 0
vocab['<pad>'] = 1
```

```
import torch.nn as nn
embedding_layer = nn.Embedding(num_embeddings = len(vocab),
                               embedding_dim = 3,
                               padding_idx = 1)
print(embedding_layer.weight)
```

num_embeddings는 룩업 테이블의 행, embedding_dim은 룩업 테이블의 열의 크기이다. 만약 3의 값을 가지는 단어의 임베딩 벡터를 룩업하고 싶다면 아래와 같이 적으면 된다.

```
embedding_layer(torch.LongTensor([[0,1]]))
```

사전 훈련된 워드 임베딩 사용하기

훈련 데이터가 많지 않아서 임베딩 벡터를 같이 학습하는 것보다는 방대한 데이터로 이미 학습된 임베딩 벡터를 사용하는 것이 좋을 수도 있다. 예를 들어, word2vec이나 glove의 워드 벡터는 아주 방대한 데이터로 학습된 임베딩 벡터가 공개되어 있다. 이를 사용하면 더 좋은 결과를 낼 수도 있다.

여기서는 torch.text에서 제공하는 IMDB 리뷰 데이터를 훈련에 사용한다.

```
from torchtext import data, datasets
TEXT = data.Field(sequential=True, batch_first=True, lower=True)
LABEL = data.Field(sequential=False, batch_first=True)
```

그리고 사전에 훈련된 word2vec 모델을 불러온다. 미리 훈련한 모델은 'word2vec_model'의 directory에 있다고 가정한다. 이제 이 모델의 벡터들을 IMDB 리뷰 데이터의 단어들에 대응시킨다.

```
import torch
import torch.nn as nn
from torchtext.vocab import Vectors

vectors = Vectors(name="word2vec_model") # 사전 훈련된 Word2Vec 모델을 vectors에 저장

TEXT.build_vocab(trainset, vectors=vectors, max_size=10000, min_freq=10) # Word2Vec 모델을 임베딩 벡터값으로 초기화

embedding_layer = nn.Embedding.from_pretrained(TEXT.vocab.vectors, freeze=False)
```

torch.text에서 제공하는 사전 훈련된 word embedding

torch.text에서는 영어 단어들의 사전 훈련된 임베딩 벡터를 아래와 같이 제공한다.

- fasttext.en.300d
- fasttext.simple.300d
- glove.42B.300d
- glove.840B.300d
- glove.twitter.27B.25d
- glove.twitter.27B.50d
- glove.twitter.27B.100d
- glove.twitter.27B.200d

- glove.6B.50d
- glove.6B.100d
- glove.6B.200d
- glove.6B.300d <= 이걸 사용해보자.

```
from torchtext.vocab import Glove

TEXT.build_vocab(trainset, vectors=GloVe(name='6B', dim=300), max_size=10000,
min_freq=10)
LABEL.build_vocab(trainset)
```

위 코드를 통해 300차원으로 임베딩된 glove 벡터를 다운받아 임베딩 벡터 초기화에 사용한다. 마찬가지로 이를 아래와 같이 nn.Embedding()의 초기값으로 사용한다.

```
embedding_layer = nn.Embedding.from_pretrained(TEXT.vocab.vectors, freeze=False)
```