LSTM 및 Seq2seq를 이용한 프랑스어-영어 번역 프로그램

1. Intro

자연어 처리는 이미지 처리와 더불어 인공지능의 발전 및 활용이 가장 활발하게 이루어지는 분야이다. 지금까지의 주된 주제는 이미지 처리였고, 자연어 처리에 활용되는 RNN에 대해서는 마지막에 소개만 짧게 들을 수 있었다. 개인적으로 인공지능을 활용한 자연어 처리에 관심이 있었기에 심화된 내용을 알고 싶었다. 그래서 사이코 고키의 '밑바닥부터 시작하는 딥러닝2'를 사용하여 간단한 자연어 처리에 대해 공부했다.

책에서는 RNN과 LSTM의 구조를 자세히 살펴보고, seq2seq 기법을 사용한 시계열 데이터 변환이 주된 내용이었다. 실습으로 날짜의 형식 변환이나, 텍스트 생성 등의 단순한 시계열 데이터를 사용한 프로그램을 만들어 볼 수 있었다.

책을 공부한 후, 이 내용을 활용하여 시계열 데이터를 활용한 프로젝트를 진행하고자 했다. 특히, 내가 공부한 자연어 처리의 내용은 굉장히 기초적인 내용이었는데, 이 내용만으로 자연어 처리에서 얼마만큼의 Performance를 보여 줄 수 있는지가 궁금했다. 그래서 선택한 주제는 '번역기' 였다. 한 언어에서 다른 언어로 번역하는 프로그램은 대중적으로 굉장히 주목받는 분야로 많은 발전이 이루어지고 있다. 최근 '파파고' 등에서 굉장한 성능을 보여주는 머신러닝을 활용한 번역기를 기초적인 내용으로 직접 구현해보는 것은 의미가 있을 것이라 생각했다. 기초 수준의 시계열 머신러닝의 성능을 알기 위한 목적에 안성맞춤이라 생각하여, 번역기 개발을 프로젝트 주제로 선택하고 직접 구현해보았다.

처음 진행했던 번역기는 한국어-영어 번역기였지만 각 언어의 구성 방식의 차이가 너무 커서 어려움이 있었다. 그 대안으로 라틴어에서 파생한 프랑스어와 라틴어에 직간접적으로 영향을 받은 영어로 언어를 선택하여 번연기를 개발했다.

2. Method

LSTM을 이용하여 Encoder-Decoder 모델을 생성한다.

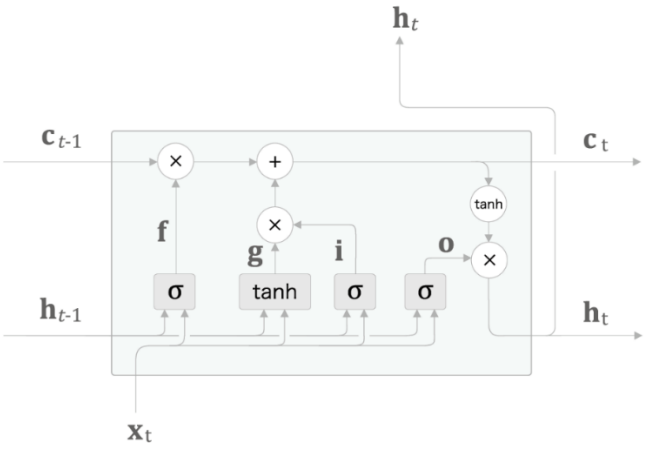
1) LSTM

LSTM은 RNN의 단점을 보완하기 위해 나온 기법이다. 기존의 RNN은 Activation function으로 주로 tanh 함수를 사용한다. tanh 함수의 미분은 그 값이 1.0 이하이고, 입력이 0으로부터 멀어질수록 작아진다. 역전파에서는 기울기가 tanh 노드를 지날 때마다 값이 계속 작아진다. 이렇게 발생하는 기울기 소실은 RNN의 약점이다.

RNN의 기울기 소실을 해결하기 위한 것이 LSTM이다. LSTM 계층의 인터페이스에는 c라는 경로가 있다. c는 기억 셀이라고 하며, LSTM 전용 기억 메커니즘이다. 기억 셀은 LSTM 계층 내에서만

완결되고, 다른 계층으로는 출력하지 않는다. 반면 은닉 상태는 h는 RNN가 마찬가지로 다른 계층으로 출력된다. 기억 셀 c는 과거로부터 현재까지 필요한 모든 정보가 저장되어 있고, 이를 바탕으로 h를 출력하게 된다. LSTM에는 기존의 RNN에서 3개의 게이트가 추가되는데, 이 게이트들은 기억 셀 c가 새로운 입력을 얼마나 반영할 지, 은닉 상태 h가 기억 셀 c와 입력 x를 얼마나 반영할지를 결정한다.

그림 6-18 input 게이트 추가



2) seq2seq

Encoder-Decoder 모델을 생성한다. Encoder에 프랑스어 데이터를 입력받아, Encoder의 LSTM 계층에서 그 특징을 저장하는 고정 길이 벡터를 출력한다. Encoder에서 출력한 벡터는 Decoder의 LSTM 계층에 넘겨준다. Decoder에는 입력 문장(출발어)로 빈 문자(eos)를 입력해주고, Encoder에서 받아온 고정 벡터를 사용하여 새로운 문장을 출력한다. 출력된 문장과 정답인 영어 데이터와 비교하여 모델을 최적화한다. 벡터는 단어 및 문장의 맥락과 잠재 정보를 포함하여 이를 넘겨준다.

Embedding 계층은 원핫 표현 시 거대한 벡터와 가중치 행렬을 곱하는 것이 비용이 크기 때문에 대신 사용된다. 결과적으로 수행하는 일은 행렬의 특정 행을 추출하는 것이기 때문에 Embedding 계층은 행렬 곱 계산을 하지 않는다. 이는 분산 표현이라고도 한다.

그림 7-5 Encoder와 Decoder가 번역을 수행하는 예

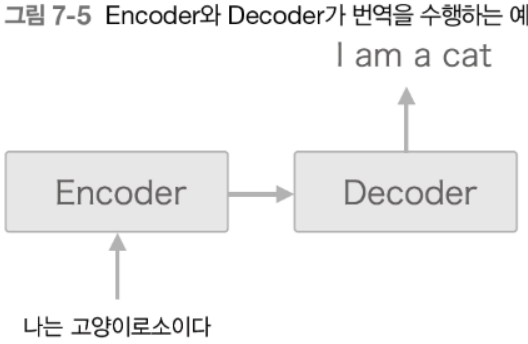
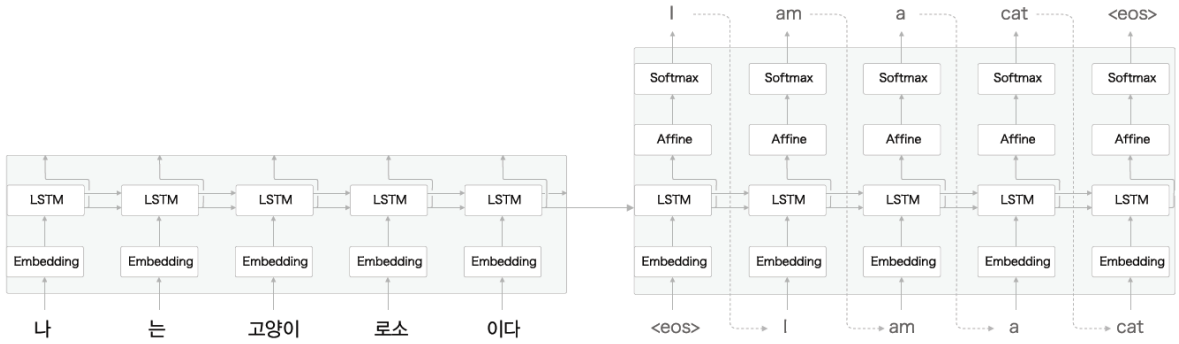


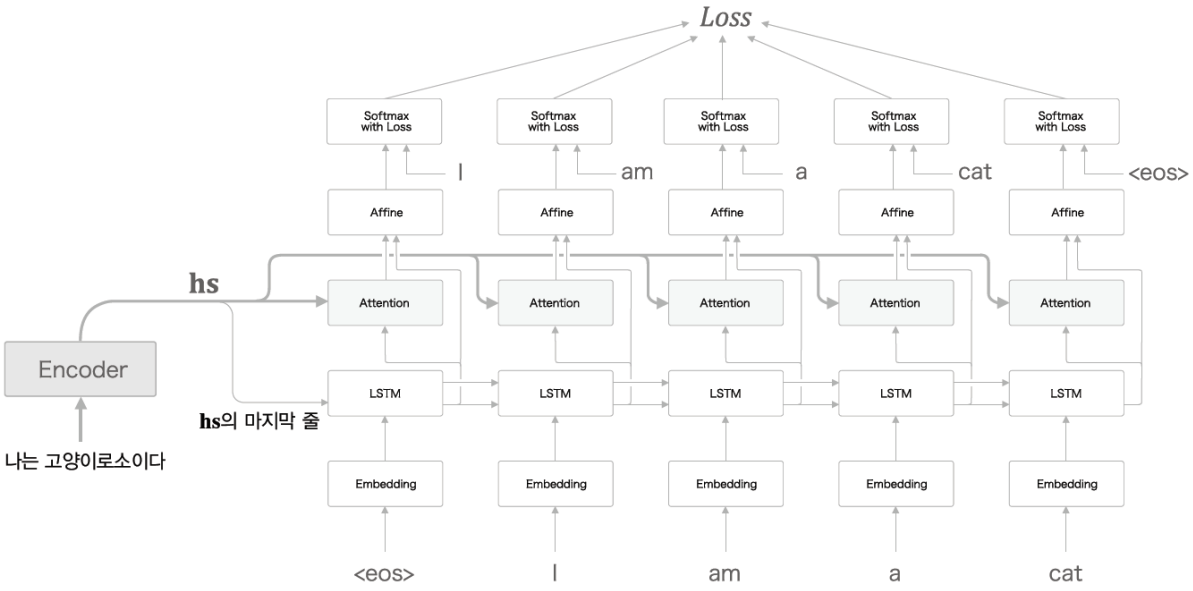
그림 7-9 seq2seq의 전체 계층 구성



3) Attention

어텐션은 Decoder에서 출력 단어를 예측하는 매 시점마다, 인코더에서의 전체 입력 문장을 다시 한번 참고한다. 하지만, 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야 할 단어와 연관이 있는 입력 단어 부분을 좀 더 집중(attention)해서 보게 된다.

그림 8-18 Attention 계층을 갖춘 Decoder의 계층 구성



3. Experiment

전처리와 번역을 실행하는 en_fr_sequence.py, en_fr_train.py 파일은 직접 구현했으며, 각 파일에서 사용하는 class 및 method는 '밑바닥부터 시작하는 딥러닝2'의 Github을 참고했다. 참고한 코드는 구현 코드에 맞춰 수정된 부분이 있다. 사용된 프로그래밍 언어는 Python3이고, 라이브러리로는 Numpy, Matplotlib, Cupy가 사용되었다.

1) Data

기계번역 워크숍(WMT)은 2000 년대 중반부터 NAACL, ACL, EMNLP 등 세계적 권위를 가진 자연어처리 학회와 함께 개최된 워크숍이다. 기계번역 워크숍에서는 매년 영어-독일어, 영어-프랑스어, 영어-힌두어 등 다양한 언어 쌍의 번역 품질 경진대회를 개최하면서 관련 병렬 코퍼스와 단독 언어 데이터를 함께 공개하고 있다. 학습 및 테스트에 사용할 데이터는 WMT19 에서 제공하는 French-German 데이터이다. 해당 데이터는 같은 의미를 가진 프랑스어, 영어 문장이 약 200 만개가 들어 있다. 200 만개를 전부 사용하면 좋겠지만, 컴퓨터 성능이 큰 데이터를 처리하지 못하는 관계로 5 만개만 사용했다.

데이터의 x 는 프랑스어(europal-v7.fr.txt)를 사용하고 y 는 영어(europal-v7.en.txt)를 사용한다.

2) Corpus preprocessing

en_fr_sequence.py 파일이 실행한다.

200만개의 문장 중 앞 5만개를 사용하여 어휘 사전을 생성하고, 데이터를 학습 데이터와 테스트 데이터로 나눈다.

```
def load_data(file_name='europarl-v7.en.txt', seed=1984, answer=False):
    file_path = os.path.dirname(os.path.abspath(__file__)) + '/' + file_name

    if not os.path.exists(file_path):
        print('No file: %s' % file_name)
        return None

    questions = []

    for line in open(file_path, 'r', encoding='UTF8'):
        idx = line.find('\n')
        questions.append(line[:idx])

    questions = questions[:5000]
```

어휘 사전은 각 단어를 사용하기 편하게 하기 위해 데이터 상의 모든 단어에 key값을 준다. 프로그램에서는 Key값을 통해 각 단어를 가리킨다. 단어를 나누기 위해 모든 문자의 문장 부호 및 괄호를 모두 없애고, 이를 띄어쓰기로 나눈 후 번호를 부여한다.

```
id_to_char = {}
char_to_id = {}
id_to_char[0] = ''
char_to_id[''] = 0
def _update_vocab(txt):
    chars = list(txt)

    for i, char in enumerate(chars):
        if char not in char_to_id:
            tmp_id = len(char_to_id)
            char_to_id[char] = tmp_id
            id_to_char[tmp_id] = char
```

```
for line in range(len(questions)):
    questions[line] = questions[line].lower()
    questions[line] = questions[line].replace('(', '')
    questions[line] = questions[line].replace(')', '')
    questions[line] = questions[line].replace('?', ' ?')
    questions[line] = questions[line].replace('!', ' !')
    questions[line] = questions[line].replace('.', ' .')
    questions[line] = questions[line].replace(',', ' ,')
    questions[line] = questions[line].replace('\'', ' \'')
    if answer:
        questions[line] = '<eos> ' + questions[line]
    questions[line] = questions[line].split(' ')

# 어휘 사전 생성
maxCount = 0
for i in range(len(questions)):
    q = questions[i]
    _update_vocab(q)
    Count = len(questions[i])
    if Count > maxCount:
        maxCount = Count
```

어휘 사전을 만든 후, 데이터를 학습 데이터와 테스트 데이터로 나눠준다. 테스트 데이터는 전체 데이터의 10%, 즉 5000개이다.

```
# 뒤섞기
indices = numpy.arange(len(x))
if seed is not None:
    numpy.random.seed(seed)
numpy.random.shuffle(indices)
x = x[indices]

# 검증 데이터셋으로 10% 할당
split_at = len(x) - len(x) // 10
train, test = x[:split_at], x[split_at:]

return train, test
```

어휘 사전 및 학습, 테스트 데이터는 pickle 파일로 저장된다.

```
x_train, x_test = load_data('europarl-v7.fr.txt')
t_train, t_test = load_data('europarl-v7.en.txt', answer=True)
char_to_id, id_to_char = get_vocab()

params = {}
params['x_train'] = x_train
params['x_test'] = x_test
params['t_train'] = t_train
params['t_test'] = t_test
params['char_to_id'] = char_to_id
params['id_to_char'] = id_to_char
pkl_file = 'en_fr_paramsTiny_50000.pkl'
with open(pkl_file, 'wb') as f:
    pickle.dump(params, f, -1)
```

3) Train

en_fr_train.py 파일이 실행한다. Pickle로 저장된 어휘 사전 및 데이터를 사용한다.

Seq2seq의 개선을 위해 입력 데이터를 반전시킨다. 입력 문장의 첫 부분에서 반전 덕분에 대응하는 변환 후 단어와 가까워지므로, 기울기가 더 잘 전해져서 학습 효율이 좋다.

```
is_reverse = True
if is_reverse:
    x_train, x_test = x_train[:, ::-1], x_test[:, ::-1]
```

모델을 만들기 위한 하이퍼 파라미터를 지정해주고, 모델을 만든다.

전체 단어의 개수(vocab_size)는 단어장의 단어 수이다. 문장을 처리할 때 고정 길이로 처리했기 때문에 최대 문자의 크기(wordvec_size)는 128로, 128자가 들어올 수 있다. 컴퓨터의 성능이 부족할 경우, 배치 사이즈(batch_size)의 크기를 줄여야 한다.

모델은 기본적인 Seq2seq, Attention을 사용하는 AttentionSeq2seq, 그리고 언급하지 않았지만 decoder의 각각 LSTM에 h를 통째로 넘겨주는 PeekySeq2seq을 쓸 수 있다. 글에서는 AttentionSeq2seq를 언급할 것이다.

```
vocab_size = len(char_to_id)
wordvec_size = 128
hidden_size = 128
batch_size = 64
max_epoch = 50
max_grad = 5.0

# model = Seq2seq(vocab_size, wordvec_size, hidden_size)
# model = PeekySeq2seq(vocab_size, wordvec_size, hidden_size)
model = AttentionSeq2seq(vocab_size, wordvec_size, hidden_size)
optimizer = Adam()
trainer = Trainer(model, optimizer)
```

프랑스어 문장을 입력으로, 영어 문장을 정답으로 하여 모델을 실행한다. 모델은 각 단어의 key를 가리키고 이를 조합하여 하나의 문장을 만든다. 이 결과와 정답을 비교하여 모델을 학습하고, 결과를 확인할 수 있다.

```
for epoch in range(max_epoch):
    trainer.fit(x_train, t_train,max_epoch=1,
                batch_size=batch_size, max_grad=max_grad)

    correct_num = 0
    for i in range(len(x_test)):
        question, correct = x_test[[i]], t_test[[i]]
        verbos = i < 10
        correct_num += eval_seq2seq(model, question,correct, id_to_char, verbos, is_reverse=True)

    acc = float(correct_num) / len(x_test)
    acc_list.append(acc)
    print('검증 정확도 %.3f%%' % (acc * 100))
```

4. Result

학습 및 번역 결과는 en_fr_attention.out를 참조한다. 결과의 Q는 x 데이터, T는 y데이터, 세번째 줄은 Q의 번역 결과이다.

1) 1 에폭 경과

```
| 에폭 1 |  반복 701 / 703 | 시간 725[s] | 손실 0.83
```

```
Q voulons-nous la survie des régions ultrapériphériques qui dépendent à 100 % des bananes , comme les canaries , ou voulons-nous la richesse de dole et de chiquita  ?
T do we want the outermost regions which depend 100% on bananas , like the canaries , to survive , or do we want dole and chiquita to accumulate more wealth ?
ESC[91m☒ESC[0m the president , the commission , the commission , the commission , the commission , the commission , the commission , the commission , the commission , the commission , the commi
```

```
Q cela s ' applique autant au contenu qu ' à l ' aspect financier .
T this applies both in terms of the substance and the financial aspect .
ESC[91m☒ESC[0m the president , the commission , the commission , the commission , the commissio
```

```
Q pour atteindre cet objectif , une coordination plus efficace au sein de l ' union européenne s ' avère dès lors indispensable .
T to this end , we need swifter coordination within the european union .
ESC[91m☒ESC[0m the president , the commission , the commission , the commission , the commission , the commission , the commission , t
```

```
검증 정확도 0.000%
```

1 에폭 경과 후, test data를 test 해본 결과이다. 전혀 학습이 이루어지지 않고, 알 수 없는 단어만 반복하고 있는 모습을 볼 수 있다. 테스트 정확도는 역시 0이다.

2) 50 에폭 경과

```
| 에폭 50 |  반복 701 / 703 | 시간 721[s] | 손실 0.14
```

```
Q ce message trouve son origine dans la position adoptée par la commission juridique dans notre résolution du 4 novembre 1999 .
T that message stems from the position adopted by the committee on legal affairs in our resolution of 4 november 1999 .
ESC[91m☒ESC[0m this message is the message that the european commission believes that position in november 1999 has taken adopted in 1999 our resolution of 1999
```

```
Q ce message trouve son origine dans la position adoptée par la commission juridique dans notre résolution du 4 novembre 1999 .
T that message stems from the position adopted by the committee on legal affairs in our resolution of 4 november 1999 .
[91m☒[0m this message is the message that the european commission believes that position in november 1999 has taken adopted in 1999 our resolution of 1999 .
```

```
Q pour atteindre cet objectif , une coordination plus efficace au sein de l ' union européenne s ' avère dès lors indispensable .
T to this end , we need swifter coordination within the european union .
ESC[91m☒ESC[0m in order to achieve this coordination is essential in terms of creating a european union union capable , therefore .
```

```
Q pour atteindre cet objectif , une coordination plus efficace au sein de l ' union européenne s ' avère dès lors indispensable .
T to this end , we need swifter coordination within the european union .
[91m☒[0m in order to achieve this coordination is essential in terms of creating a european union union capable , therefore .
```

```
Q cela s ' applique autant au contenu qu ' à l ' aspect financier .
T this applies both in terms of the substance and the financial aspect .
ESC[91m⊠ESC[0m this applies to the content of financial aspect .
```

```
Q cela s ' applique autant au contenu qu ' à l ' aspect
financier .
T this applies both in terms of the substance and the financial aspect .
[91m⊠[0m this is being applied as this aspect of financial aspect
```

```
검증 정확도 1.700%
```

위의 1, 2번 문장은 50에폭을 진행 후 가져온 결과이고, 3번째 문장은 49에폭을 진행 후 가져온 결과이다. 아직 대부분의 문장, 특히 길이가 긴 문장에 대해서는 단어는 유사하고 읽는 것이 가능해졌지만, 문법과 의미가 정확하지 않은 모습을 보인다.

하지만, 이 결과는 1에폭 학습 결과와 사뭇 다른 모습을 보여준다. 먼저 특정 단어를 반복하기만 하며 아무런 의미도 가지지 않았던 1에폭과 달리, 50에폭 학습 결과는 다양한 단어가 결과로 출력되었다. 여기서 이 단어들이 아무런 이유가 없이 나오지는 않았다. 단어들은 프랑스어 문장과 비슷한 의미를 지닌 단어들이 출력되었고, 일부 프랑스어와 영어의 표기가 비슷한 단어들에 대해서는 정답 데이터와 정확히 같은 단어를 출력하기도 했다.

결과는 통해 문장의 의미를 유추할 수도 있었다. 모든 문장의 단어 조합은 완벽하지는 않지만, 이해가 가능은 한 정도로 문법에 알맞은 모습을 보인다. 위의 세 예시는 번역 결과 문장의 의미가 정답 데이터의 의미와 유사한 결과들이다. 첫 번째 문장의 번역 결과와 정답 모두에서 '위원회가 1999년 11월 입장을 채택했다.'라는 의미를 읽어낼 수 있다. 두 번째 문장의 번역 결과와 정답 모두 '유럽연합은 조정이 필요하다'라는 의미를 읽어낼 수 있다. 세 번째 문장은 '이것은 실질적 재정적 측면에서 모두에 적용된다.'라는 의미이고, 그 결과는 '이것은 재무 측면에서 적용된다'라는 의미로 '실질적' 부분을 제외하고는 같은 의미를 갖는다.

검증 정확도 또한 1.7%로 전혀 맞추지 못했던 학습 전과는 다르다. 1.7이라는 수치가 작아 보일 수 있지만, 전체 테스트 데이터가 5000개임을 감안한다면 85개의 문장에서 정확도를 보인 것이다. 정확도는 정답 데이터와 번역 결과나 정확히 같은 경우를 확인하기 때문에 이 결과는 소수의 문장에서 정확한 번역이 일어났음을 알려준다.

5. Conclusion

LSTM 및 Seq2seq를 이용하여 프랑스어-영어 번역기를 만들었다. 해당 번역기는 번역을 완벽하게 실행하지는 못했지만, 의미 있는 결과를 냈다. 모든 문장의 번역 결과는 정답 데이터와 유사한 단어들이 조합되었고, 그 문법 또한 어느 정도 들어맞는 모습을 보여준다. 일부 문장에 대해서는 많은 부분 비슷한 의미를 지니도록 번역되었으며, 정확히 번역된 문장도 존재한다. 이 결과는 10년 전 사용해봤던 구글 번역기 결과와 유사했다. 기초적인 자연어 처리 지식과 LSTM과 단순한 Seq2seq 구조를 사용해서 의미를 가지는 번역기 프로그램을 만들 수 있었다.

하지만, 이 프로그램이 좋은 번역기라고는 할 수 없다. 일부 문장을 제외한 결과에서 의미를 정확하게 예측하지 못했다. 이 번역기를 믿고 번역을 맡기는 것은 불가능할 것이다. 번역기를 만들면서 여러 한계가 있었고 이를 해결한다면 더 좋은 번역기를 만들 수 있을 것이다.

먼저 모델을 학습시키기 위한 컴퓨터의 성능이 부족했다. 성능의 한계에 부딪혀 타협을 보았던 것은 데이터의 수와 에폭 수였다. 데이터 수는 5만개로 너무 적었다. 수많은 사람이 사용하는 언어에서 문장 5만개는 불충분하다. 컴퓨터 성능의 한계로 5만개에 그쳤으나, 이를 데이터 전체(200만 개)를 넘어 더 많은 데이터를 사용한다면 번역 결과가 개선될 것으로 예상된다. 시간이 오래 걸리는 탓에 에폭 수를 50개로 한정 지었다. 학습을 오래 할 수 있는 환경이 있으면 더 많은 학습을 할 수 있을 것이고, 성능이 충분하다면 배치 수를 늘릴 수도 있을 것이다.

번역기는 numpy로 구현되었다. 강의에서 사용했던 Tensorflow를 비롯하여 머신러닝을 처리하는 많은 라이브러리가 존재한다. 이 라이브러리를 사용하면 내장 함수와 그래픽 카드를 효율적으로 사용하여 그 비용이 줄고 더 좋은 결과를 얻을 수 있을 것이다. 책에서 구현된 코드가 numpy 라이브러리 더하여 cupy 라이브러리로 구성되어 있었지만, 이를 머신러닝 라이브러리를 사용한다면 성능이 개선될 것이다.

프로젝트를 진행하며 한 가지 아쉬웠던 것은 고정 길이 시계열 데이터만 처리가 처리할 수 있었던 것이다. 지식의 한계로 가변 길이에 대해서는 데이터를 처리할 수 없었고, 문장의 길이를 가변 길이로 만들어야 했다. 이를 해결한다면 유연성 있게 시계열 처리가 가능했을 것이다. 이 외에도 기초에서 벗어나 고도화된 자연어 처리 기법들을 사용한다면 더 좋은 결과를 낼 수 있을 것이다.

아무도 사용하지 않던 인공지능 번역기는 이제 '파파고' 등과 같이 널리 사용된다. 인공지능을 이용한 번역기를 개발하던 초기에 이 프로젝트 같은 시도가 있었을 것이라 생각한다. 부족한 번역기지만, 그들의 경험을 재현하며, 그 결과와 한계를 경험하는 것이 이 프로젝트의 의의이다.

6. Reference

참고 자료:

사이토 고키, 『밑바닥부터 시작하는 딥러닝2』, 한빛미디어, 2019

참고 코드:

밑바닥부터 시작하는 딥러닝2 Github, https://github.com/WegraLee/deep-learning-from-scratch-2

데이터:

WMT19, https://www.statmt.org/wmt19/translation-task.html#download