chatper 7 - RNN을 사용한 문장 생성

- 이번 장에서는 언어 모델을 사용해 '문장 생성'을 수행합니다. 구체적으로는 우선 말뭉치를 사용해 학습한 언어 모델을 이용하여 새로운 문장을 만들어냅니다.
- 여기서 멈추지 않고 seq2seq라는 새로운 구조의 신경망도 다룹니다. seq2seq는 한 시계열 데이터를 다른 시계열 데이터로 변환하는 걸 말합니다.

7.1 언어 모델을 사용한 문장 생성

7.1.1 RNN을 사용한 문장 생성의 순서

- 여기서 주목할 것은 이렇게 생성한 문장은 훈련 데이터에는 존재하지 않는, 말 그대로 새로 생성된 문장이라는 것입니다. 왜냐하면 언어 모델은 훈련 데이터를 암기한 것이 아니라, 훈련 데이터에서 사용된 단어의 정렬 패턴을 학습한 것이기 때문이죠. 만약 언어 모델이 말뭉치로부터 단어의 출현 패턴을 올바르게 학습할 수 있다면, 그 모델이 새로 생성하는 문장은 우리 인간에게도 자연스럽고 의미가 통하는 문장일 것이므로 기대할 수 있습니다.

7.2 seq2seq

7.2.1 seq2seq의 원리

- seq2seq를 Encoder - Decoder 모델이라고도 합니다. 이름이 말해주듯이 여기에는 2개의 모듈, Encoder와 Decoder가 등장합니다. 문자 그대로 Encoder는 입력 데이터르 인코딩(부호화)하고, Decoder는 인코딩된 데이터를 디코딩(복호화)합니다.
- Encoder : 나는 고양이다 => Decoder : I am a cat.
- seq2seq 모델은 Encoder와 Decoder가 협력하여 시계열 데이터를 다른 시계열 데이터로 변환한다는 것입니다. 그리고 Encoder와 Decoder로는 RNN을 사용할 수 있습니다.
- 인코딩한다라함은 결국 임의 길이의 문장을 고정 길이 벡터로 변환하는 작업이 됩니다.
- Decoder은 앞 절의 신경망과 완전히 같은 구조입니다. 단 한 가지만 빼고 말이죠. 바로 LSTM 계층이 벡터 h를 입력받는다는 점이 다릅니다. 참고로, 앞 절의 언어 모델에서는 LSTM 계층이 아무것도 받지 않았습니다.(굳이 따지자면, 은닉 상태로 '영벡터'를 받았다고 할 수 있습니다.)

7.2.2 시계열 데이터 변환용 장난감 문제

- 그런데 우리는 짖금까지 word2vec이나 언어 모델 등에서 문장을 '단어' 단위로 분할해왔습니다. 하지만 문장을 반드시 단어로 분할해야 하는 건 아니랍니다.

7.2.3 가변 길이 시계열 데이터

- 미니배치 수행 시엔 미니배치에 속한 샘플들의 데이터 형상이 모두 똑같아야 합니다.
- 가변 길이 시계열 데이터를 미니배치로 학습하기 위한 가장 단순한 방법은 [패딩]을 사용한느 것입니다.
- 패딩을 적용해야 하지만 정확성이 중요하다면 seq2seq에 패딩 전용 처리를 추가해야 합니다.
- 예컨대 Decoder에 입력된 데이터가 패딩이라면 손실의 결과에 반영하지 않도록 합니다(Softmax

with Loss 계층에 '마스크' 기능을 추가해 해결할 수 있습니다). 한편 Encoder에 입력된 데이터가 패딩이라면 LSTM 계층이 이전 시각의 입력을 그대로 출력하게 합니다. 즉, LSTM 계층은 마치 처음부터 패딩이 존재하지 않았던 것처럼 인코딩할 수 있습니다.

7.3 seq2seq 구현

7.3.1 Encoder 클래스

- [그림 7-14]와 같이 Encoder 클래스는 Embedding 계층과 LSTM 계층으로 구성됩니다. Embedding 계층에서는 문자를 문자 벡터로 변환합니다. 그리고 이 문자 벡터가 LSTM 계층으로 입력됩니다.
- LSTM 계층은 오른쪽(시간 방향)으로는 은닉 상태와 셀을 출력하고 위쪽으로는 은닉 상태만 출력합니다. 이 구성에서 더 위에는 다른 계층이 없으니 LSTM 계층의 위쪽 출력은 폐기됩니다. [그림 7-14]에서 보듯 Encoder에서는 마지막 문자를 처리한 후 LSTM 계층의 은닉 상태 h를 출력합니다. 그리고 이 은닉 상태 h가 Decoder로 전달됩니다.
- 그런데 우리는 시간 방향을 한꺼번에 처리하는 계층을 Time LSTM 계층이나 Time Embedding 계층으로 구현했습니다. 이러한 Time 계층을 이용하면 우리의 Encoder는 [그림 7-15]처럼 됩니다.

7.3.2 Decoder 클래스

- Decoder 클래스는 [그림 7-16]에서 보듯, Encoder 클래스가 출력한 h를 받아 목적으로 하는 다른 문자열을 출력합니다.
- 앞 절에서 설명한 것처럼 Decoder과 RNN으로 구현할 수 있습니다. Encoder과 마찬가지로 LSTM 계층을 사용하면 되며, 이때 Decoder의 계층 구성은 [그림 7-17]처럼 됩니다.

7.3.4 seq2seq 평가

- seq2seq의 학습은 기본적인 신경망의 학습과 같은 흐름으로 이뤄집니다.
  1) 학습 데이터에서 미니배치를 선택하고, 2) 미니배치로부터 기울기를 계산하고, 3) 기울기를 사용하여 매개변수를 갱신한다.

7.4 seq2seq 개선

7.4.1 입력 데이터 반전(Reverse)

- 입력 문장의 첫 부분에서는 반전 덕분에 대응하는 변환 후 단어와 가까우므로, 기울기가 더 잘 전해져서 학습 효율이 좋아진다고 생각할 수 있습니다. 다만, 입력 데이터를 반전해도 단어 사이의 '평균'적인 거리는 그대로입니다.

7.4.2 엿보기(Peeky)

- 앞에서 설명했듯이 Encoder는 입력 문장(문제 문장)을 고정 길이 벡터 h로 변환합니다. 이때 h 안에는 Decoder에게 필요한 정보가 모두 담겨 있습니다. 즉, h가 Decoder의 유일한 정보인 셈입니다. 그러나 현재의 seq2seq는 [그림 7-25]와 같이 최초 시각의 LSTM 계층만이 벡터 h를 이용하고 있습니다. 이 중요한 정보인 h를 더 활용할 수는 없을까요?
- 여기서 seq2seq의 두 번째 개선안이 등장합니다. 중요한 정보가 담긴 Encoder의 출력 h를 Decoder의 다른 계층에게도 전해주는 것이죠.

- 그 전 모델과 비교하면, 기존에는 하나의 LSTM만이 소유하던 중요 정보 h를 여러 계층이 공유함을 알 수 있습니다. 이는 집단지성에 비유할 수 있겠군요. 즉, 중요한 정보를 한 사람이 독점하는 게 아니라 많은 사람과 공유한다면 더 올바른 결정을 내릴 가능성이 커질 겁니다.

7.5 seq2seq를 이용하는 애플리케이션

- 기계번역 / 자동요약 / 질의응답 / 메일 자동 응답