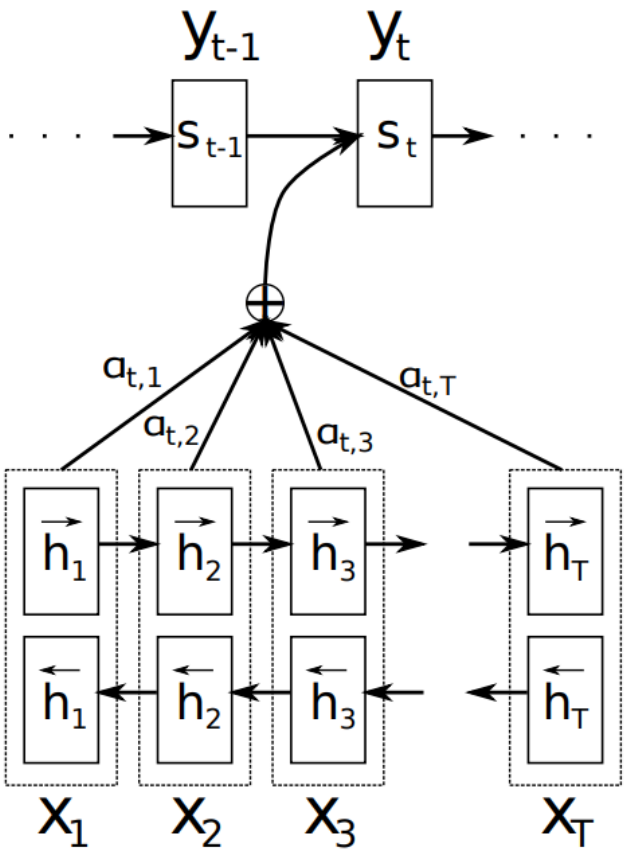
요약

- 논문: http://cs224d.stanford.edu/reports/Lichy.pdf

Attention Model

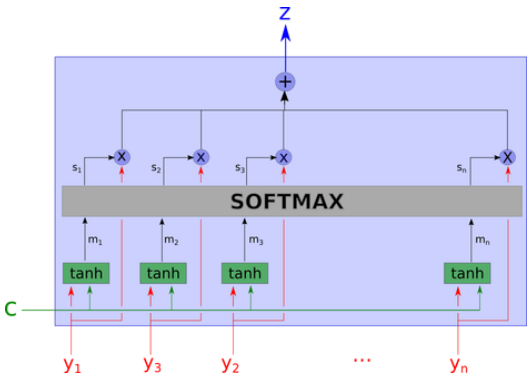
- 딥러닝 모델이 벡터 Sequence 중에서 가장 중요한 벡터에 집중하도록 하는 모델
  - State 를 고려하여 가장 중요도가 높은 벡터를 중심으로 하나의 벡터로 정리하는 모델.
  - Input
    - y_1, y_2, …, y_n: 입력 벡터. 1 차원 데이터인 Sequence 뿐만 아니라 2 차원 데이터인 이미지 등도 받을 수 있다.
    - c: Context. 현재 상태(문맥 등)을 나타내는 벡터.
  - Output
    - z: context 와 sequence 를 고려하여, y 벡터 중 가장 중요한 벡터를 위주로 summary 된 값.

Attention Model 의 구조와 동작 방법



Attention Model 은 개념적으로 아래와 같이 동작한다.

1. Input 으로 들어온 벡터들의 중요도/유사도를, 현재 state 를 고려하여 구한다.
2. 각각의 중요도를, 총 합이 1 이 되는 상대값으로 바꾼다.
3. 상대값 중요도를 가중치로 보고, Sequence 에 있는 벡터들을 가중치합한다.



Attention Model 의 기본 구조

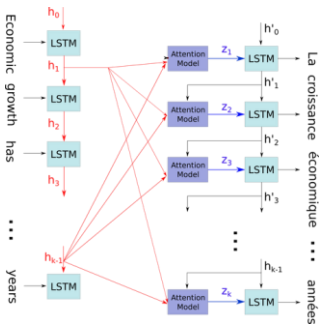
위 과정을 수학적인 방법으로 설명하면 아래와 같다.

1. State C 에 W_c 행렬을 내적한 값과, 각각의 Sequence 의 벡터 y_i 에 W_y 행렬을 내적한 값을 더한다. 그리고 이 값을 tanh 함수에 통과킨 값을 m_i 이라고 한다.
2. m_i 값을 Softmax 함수에 통과시켜 확률을 구한 값을 s_i 라고 한다.
3. s_i 값과 y_i 값을 내적한다. 이 값을 전부 합친 것이 출력값이 된다.

- 여기서 W_c, W_y 행렬은 학습을 통해 값을 정한다.
- 1 에서 사용한 방법은 y_i 값과 C 값을 하나로 섞는 이상, 어떠한 방법을 사용해도 무방하다. (예: y_i 와 C 값을 내적)

Encoder/Decoder 와 Attention Machanism

- 값을 입력받아 벡터로 만드는 encoder(인코더), 인코더가 출력한 벡터를 바탕으로 원하는 결과를 출력하는 decoder(디코더)가 있다고 하자.
  - 일반적인 모델: 인코더의 모든 출력 벡터를 골고루 보고 결과를 출력한다.
  - Attention Mechanism: 인코더의 출력 벡터 중 중요한 벡터에 집중한다.
  - 예) “Artificial Intelligence”를 “인공 지능”으로 번역한다고 가정한다. 이 때 모델이 ’지능’을 예측할 때 ’Intelligence’에 주목하게 된다.
- Decoding 과정에서 불필요한 벡터를 보지 않기 때문에 성능이 향상된다.



RNN 을 활용한 기계 번역에 활용된 Attention Model

- RNN Encoder/Decoder 를 사용할 때 Attention Model 의 Input
  - y_1, y_2, ..., y_n: 보통 이전 RNN layer 가 출력한 값의 Sequence 로 사용.
  - c: Attention Model 의 Output 을 사용하는 RNN 모델의 바로 직전 State 를 사용할 수 있다. 예를 들어 RNN 모델의 세 번째 Output 을 계산한다면, 두 번째 Output 의 hidden state 값을 사용할 수 있다.

Attention Model 의 Code Example

이전 layer 의 출력 값이 1 차원인 경우

```
# 이전 Layer 의 전체 값들을 Input 으로 받는다
inputs = Input(shape=(input_dims,))

# 각각의 값에 대한 중요도를 구한다.
attention_probs = Dense(input_dims, activation='softmax', name='attention_probs')(inputs)

# 각각의 값 Matrix 와 중요도 Matrix 를 행렬곱한다. 즉, 입력 값들을 중요도에 따라 가중치합한다.
attention_mul = merge([inputs, attention_probs], output_shape=input_dims, name='attention_mul', mode='mul')
```

- philipperemy/keras-attention-mechanism 에서 소스를 가져와 주석을 달음.
- 수학적인 의미의 행렬(Matrix)를 Dense Layer 로 대체하여 구현.

이전 layer 의 출력이 2 차원인 경우

```
# inputs.shape = (batch_size, time_steps, input_dim)
input_dim = int(inputs.shape[2])

 # Input Matrix 를 Transpose 한다.
a = Permute((2, 1))(inputs)
a = Reshape((input_dim, TIME_STEPS))(a)

# 각각의 값에 따라 중요도를 구한다. (LSTM 의 cell 별)
# Matrix 로 보면, 아래 layer 의 출력은 행: sequence index, 열: input_dim
a = Dense(TIME_STEPS, activation='softmax')(a)

# 다시 Matrix 를 Transpose 해서 Input Matrix 와 차원의 의미가 같도록 수정
a_probs = Permute((2, 1), name='attention_vec')(a)
# a_probs - 행: input_dim, 열: sequence index

# Input 값들을 중요도에 따라 가중치합한다.
output_attention_mul = merge([inputs, a_probs], name='attention_mul', mode='m
ul')
```

- philipperemy/keras-attention-mechanism 에서 소스를 가져와 주석을 달음.
- Input Matrix 를 Transpose 하는 이유
  - "각각의 Sequence 의 벡터 y_i 에 W_y Matrix 를 내적한 값을 더한다."를 Dense layer 로 구현하기 위해.
  - y_i 를 전체로 모은 Y Matrix 와 M_y Matrix 를 내적한다고 생각하면 된다.
  - Dense layer 하나를, 학습해야 할 Matrix 로 생각하면 이해할 수 있다.

Attention Mechanism 의 활용

- Output 이 Sequence 형태라면, Input 이 어떤 형태든 간에, Input 의 특정 부분을 강조해서 보는 형태로 Attention Mechanism 을 사용할 수 있다.

RNN 의 Long-Term Dependency 해결

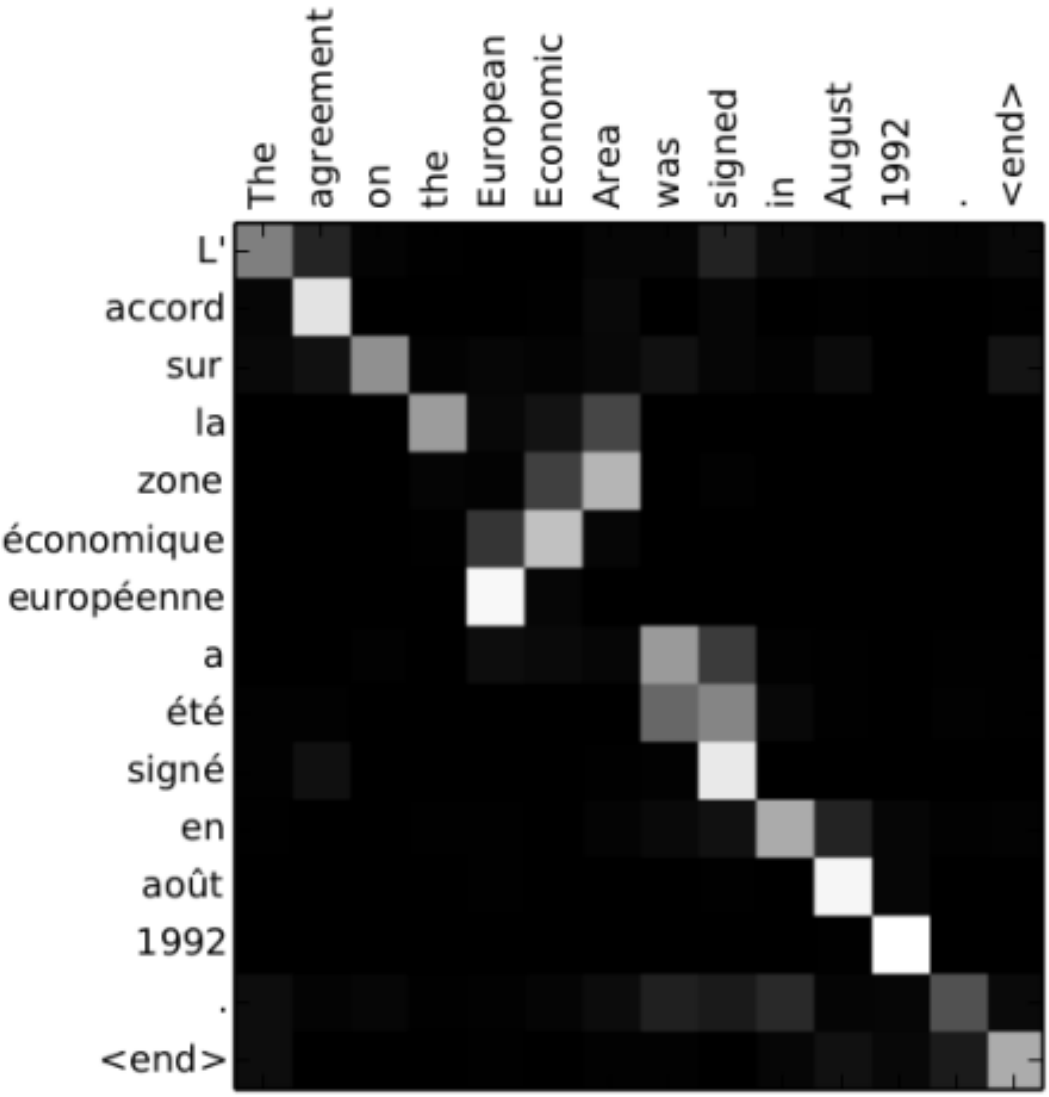
- Long-Term Dependency(장기 의존성)

  - 제공된 데이터와 배워야 할 정보의 입력 차이(Gap)가 큰 경우 두 정보의 문맥을 연결하기 어려운 현상.
  - LSTM(Long Short Term Memory Network) 등을 활용하여 해결할 수 있다.
- Attention Mechanism 을 이용하면 Sequence 가 길더라도 그 중에서 중요한 벡터에 집중할 수 있으므로, Long-Term Dependency 를 해결할 수 있다.
- Seq2Seq 를 이용한 영어-독일어간 번역 모델을 가지고 실험을 한 결과, Attention Model 을 적용하면 성능이 소폭 상승한 결과도 있다.

딥러닝 모델이 잘 학습되었는지 확인

- 딥러닝 모델이 제대로 학습되지 않았으면, 위와 같이 연관된 벡터 간의 중요도가 높게 나오지 않을 것이다. 이런 현상을 통해 딥러닝 모델이 제대로 학습되었는지 확인 가능하다.

- Output 에서 Sequence Input 중에서 어느 값이 중요하게 사용되었는지 알 수 있다.



  - 위 이미지에서 보면, 두 단어의 뜻이 같은 부분이 활성화가 되어 있는 것을 확인할 수 있다.
  - 예를 들어 불어 accord 와 영어 agreement 는 같은 뜻이므로 Attention Model 의 Output 이 큰 것을 확인할 수 있다.
- 이미지 캡셔닝 모델의 경우, 각 단어별로 이미지의 어떤 부분 때문에 그 단어가 생성되었는지 확인 가능하다.

참고자료

논문

- Natural Language Inference, Sentence representationand Attention Mechanism
- Attention Is All You Need

기타

- Attention Mechanism (한국어 번역)
- Effective Approaches to Attention-based Neural Machine Translation (한국어 해설)
- Attention Mechanism 시각화)
- LSTM(RNN) 소개