ISSN 2288-4866 (Print)
ISSN 2288-4882 (Online)
http://www.jiisonline.org
J Intell Inform Syst 2017 June: 23(2): 71~88 http://dx.doi.org/10.13088/jiis.2017.23.2.071

한국어 음소 단위 LSTM 언어모델을 이용한 문장 생성*

안성만
국민대학교 경영학부
(*sahn@kookmin.ac.kr*)

정여진
국민대학교 경영학부
(*ychung@kookmin.ac.kr*)

이재준
국민대학교 데이터사이언스 학과
(*leejj4787@gmail.com*)

양지헌
국민대학교 데이터사이언스 학과
(*jihunyang@hanmail.net*)

언어모델은 순차적으로 입력된 자료를 바탕으로 다음에 나올 단어나 문자를 예측하는 모델로 언어처리나 음성인식 분야에 활용된다. 최근 딥러닝 알고리즘이 발전되면서 입력 개체 간의 의존성을 효과적으로 반영할 수 있는 순환신경망 모델과 이를 발전시킨 Long short-term memory(LSTM) 모델이 언어모델에 사용되고 있다. 이러한 모형에 자료를 입력하기 위해서는 문장을 단어 혹은 형태소로 분해하는 과정을 거친 후 단어 레벨 혹은 형태소 레벨의 모형을 사용하는 것이 일반적이다. 하지만 이러한 모형은 텍스트가 포함하는 단어나 형태소의 수가 일반적으로 매우 많기 때문에 사전 크기가 커지게 되고 이에 따라 모형의 복잡도가 증가하는 문제가 있고 사전에 포함된 어휘 외에는 생성이 불가능하다는 등의 단점이 있다. 특히 한국어와 같이 형태소 활용이 다양한 언어의 경우 형태소 분석기를 통한 분해과정에서 오류가 더해질 수 있다. 이를 보완하기 위해 본 논문에서는 문장을 자음과 모음으로 이루어진 음소 단위로 분해한 뒤 입력 데이터로 사용하는 음소 레벨의 LSTM 언어모델을 제안한다. 본 논문에서는 LSTM layer를 3개 또는 4개 포함하는 모형을 사용한다. 모형의 최적화를 위해 Stochastic Gradient 알고리즘과 이를 개선시킨 다양한 알고리즘을 사용하고 그 성능을 비교한다. 구약성경 텍스트를 사용하여 실험을 진행하였고 모든 실험은 Theano를 기반으로 하는 Keras 패키지를 사용하여 수행되었다. 모형의 정량적 비교를 위해 validation loss와 test set에 대한 perplexity를 계산하였다. 그 결과 Stochastic Gradient 알고리즘이 상대적으로 큰 validation loss와 perplexity를 나타냈고 나머지 최적화 알고리즘들은 유사한 값들을 보이며 비슷한 수준의 모형 복잡도를 나타냈다. Layer 4개인 모형이 3개인 모형에 비해 학습시간이 평균적으로 69% 정도 길게 소요되었으나 정량지표는 크게 개선되지 않거나 특정 조건에서는 오히려 악화되는 것으로 나타났다. 하지만 layer 4개를 사용한 모형이 3개를 사용한 모형에 비해 완성도가 높은 문장을 생성했다. 본 논문에서 고려한 어떤 시뮬레이션 조건에서도 한글에서 사용되지 않는 문자조합이 생성되지 않았고 명사와 조사의 조합이나 동사의 활용, 주어 동사의 결합 면에서 상당히 완성도 높은 문장이 발생되었다. 본 연구결과는 현재 대두되고 있는 인공지능 시스템의 기초가 되는 언어처리나 음성인식 분야에서 한국어 처리를 위해 다양하게 활용될 수 있을 것으로 기대된다.

주제어 : 언어 모델, 순환 신경망 모형, LSTM 모형, 문장 생성 모형

논문접수일 : 2017년 4월 3일 논문수정일 : 2017년 6월 15일 게재확정일 : 2017년 6월 16일
원고유형 : 일반논문 교신저자 : 정여진

* 이 논문은 미래창조과학부의 재원으로 한국연구재단의 신진연구자지원사업 지원을 받아 수행된 것임(No. 2016R1C1B1010940).

1. 서론

언어 모델(language model)은 말뭉치(corpus) 데이터를 학습하여 주어진 문장이 인간의 언어와 얼마나 유사한지를 확률적으로 표현하는 모델로서 음성인식과 언어처리 분야에서 활발히 연구되고 있다. 특히 인공지능과 딥러닝이 대두되면서 딥러닝 알고리즘을 기반으로 한 신경망 언어 모델(neural language model)은 큰 주목을 받고 있다. 언어모델의 구축은 효율적인 문서 압축(Rissanen and Langdon, 1979)이나 장애인의 원활한 컴퓨터 활용을 위한 어플리케이션(Ward et al., 2000) 등에 직접적으로 적용할 수 있을 뿐 아니라 보다 근본적으로는 인간의 지성에 상응하는 이해를 전제로 한다는 점에서(Hutter, 2006) 지능형 시스템을 구성하는 중요한 문제로 여겨진다.

딥러닝 알고리즘은 최근 컴퓨팅 기술의 발전에 따라 국외뿐 아니라 국내의 다양한 분야에서 주목받고 있다(Ahn, 2016; Kim et al., 2016; Lee et al., 2016). 그 중 하나인 순환신경망(recurrent neural network; RNN) 모델과 이를 발전시킨 Long short-term memory(LSTM) 모델은 순차적인 정보를 하나씩 입력 받아 처리하여 내부 노드에 저장하고 모형에 반영한다. 이는 언어모델과 같이 이전에 입력된 자료가 뒤이어 입력되는 자료와 높은 상관관계를 가지는 데이터를 학습하기에 적당하여 언어 모델에 활발히 적용되고 있다(Gers and Schmidhuber, 2001; Mikolov et al., 2010; Sundermeyer et al., 2012). 위의 모형에서 한 번에 입력되는 정보의 단위는 단어 레벨(word-level) 혹은 형태소 레벨(morpheme-level)로 규정하는 것이 일반적이다. 하지만 한국어를 비롯하여 러시아어, 터키어, 헝가리어, 핀란드어와 같이 형태소의 활용이 다양한 언어의 경우 형태소를 단위로 하는 언어 모델을 적용한다면 불완전한 형태소 분석기로 인해 오류가 발생될 가능성이 추가 된다. 뿐만 아니라 데이터가 포함하는 단어나 형태소의 수가 많아 사전의 크기가 커지면서 모델의 복잡도가 증가하는 문제가 있고 학습 데이터에 포함되지 않은 어휘가 모형을 통해 생성될 수 없다는 등의 단점이 있다(Lankinen et al., 2016).

이러한 문제를 보완하기 위한 방법으로 형태소보다 더 작은 단위인 문자 레벨(character-level)로 데이터를 분해하여 언어모델을 적용하는 방법이 최근 주목을 받고 있다(Lankinen et al., 2016; Lee et al., 2016b; Chung et al., 2016b; Ling et al., 2015; Bojanowski et al., 2015; Jozefowicz et al., 2016; Kim et al., 2015). Kim et al.(2016)은 한국어를 문자 단위로 분할한 뒤 순환신경망 모델을 기반으로 한 언어모델을 적용하여 형태소 레벨, 단어 레벨의 모형과 성능을 비교하였다. 하지만 한글은 영미권의 다른 언어들과 다르게 하나의 문자가 초성, 중성, 종성이 결합되어 이루어지기 때문에 문자 단위의 언어모델 역시 상당히 큰 크기의 사전을 사용할 수밖에 없다.

본 논문에서는 텍스트를 문자보다 더 작은 단위인 자음과 모음으로 이루어진 음소 단위로 분할하여 사전을 구성하고 이를 기반으로 문장 생성 시스템을 구축한다. 이 경우 사전 크기가 현저히 작아지기 때문에 임베딩 계층을 통해 사전 크기보다 작은 차원의 공간으로 데이터를 매핑하는 과정을 생략할 수 있고 앞서 말한 모형 복잡도 증가, 어휘의 다양성 제한 등의 단점을 보완할 수 있다. 본 연구에서는 3개 혹은 4개의 LSTM 계층을 사용하여 모델을 수립하고 다양한 최적화 알고리즘을 적용해 모델을 학습시킨다.

구약성경 자료를 활용한 실험에서 학습된 모델이 한국어의 특징에 맞는 문장을 잘 생성하는지 정성적, 정량적으로 살펴볼 것이다.

본 논문의 구성은 아래와 같다. 2절에서는 언어모델에 관한 연구들을 살펴보고 3절에서는 순환신경망 모델과 최적화 알고리즘에 대해 설명한다. 4절에서는 본 논문에서 사용된 음소 단위의 문장생성 모델을 소개하고 5절에서는 실제 한글 데이터를 사용하여 실험한 결과를 설명한 뒤 6절에서 결론을 도출한다.

2. 관련 연구

N-gram 모델은 전통적으로 사용되어 온 언어모델로 문장을 N개의 연속적인 단위 개체(단어, 형태소, 음절, 문자 등)로 분절하고 뒤이어 나오는 개체를 Markov 모형과 같은 확률적 모형을 사용하여 예측하는 언어모델이다. 이 모형은 N개로 제한된 과거 시점 데이터 만을 사용하고 출현 빈도에 기반하여 문장을 생성하기 때문에 단위 개체들 간의 의존 관계가 모형에 충분히 반영되지 않는다. 하지만 RNN 모델과 그의 일종인 LSTM 모델을 사용한 언어모델은 단위 개체 간의 순차적 의존관계를 효과적으로 모델에 반영할 수 있기 때문에 기존의 N-gram 모델보다 더 나은 성능을 가진다고 알려져 있다(Gers and Schmidhuber, 2001; Mikolov et al., 2010; Sundermeyer et al., 2012).

신경망 언어모델은 문장을 분할하여 총 T개의 개체를 포함하는 사전을 구축한 뒤 각 입력개체를 T의 길이를 가지는 원-핫 벡터(one-hot vector)로 변환한 값을 입력값으로 가진다. 입력된 개체들에 뒤이어 나올 개체의 발생확률을 출력층에서 softmax 활성화 함수를 사용하여 생성한 뒤 가장 확률이 높은 개체를 출력값으로 나타낸다. 이 때 입력개체는 단어, 형태소, 혹은 문자로 구성하는 것이 일반적이다. 예를 들면, ‘unbreakable’이란 단어가 데이터에 포함되어 있을 때 단어 자체를 사전에 포함시키거나, ‘un’, ‘break’, ‘able’와 같이 분해된 형태소들을 사전에 포함시키거나, ‘u’, ‘n’, ‘b’, ‘r’, ‘e’, ‘a’, ‘k’, ‘a’, ‘b’, ‘l’, ‘e’과 같이 문자를 사전에 포함시킬 수 있다.

문자 단위의 언어모델은 학습 데이터에 포함되지 않았지만 나올 법한 어휘를 생성하는 것이 가능하고 드물게 나타나는 형태소 활용 형태도 생성 가능하다는 장점이 있기 때문에 한국어와 같이 형태소 활용이 다양한 언어들에 적용하기 적합하다(Chung et al., 2016). 또한 사전이 포함하는 단위 개체의 개수가 적어 학습이 빠르고 쉬울 뿐 아니라(Sutskever and Martens, 2011) 형태소 분석기를 거치치 않고 바로 모형화가 가능하기 때문에 불완전한 형태소 분석으로 인한 언어적 오류를 감소시킬 수 있다(Chung et al., 2016). Lankinen et al. (2016)은 형태소 활용이 다양한 언어 중 하나인 핀란드어에 문자 단위의 모형을 적용하여 LSTM기반 언어모델을 학습시킨 결과 학습 데이터에 존재하지 않는 형태소 활용 형태를 모형이 생성 가능하였고 문법적으로 옳은 문장을 생성한다는 것을 보여주었다.

한글 자료에 순환신경망 모형을 적용하여 문장을 분석한 연구로는 문장 간 구문 유사도 측정하거나(Lee et al., 2016a) 의미역 결정에 적용한 예가 있다(Bae and Lee, 2015). 한글 문장생성 모형에 순환신경망 모형을 사용한 연구로는 Kim et al.(2016)이 LSTM 모형을 어절, 형태소, 음절 단위로 학습시킨 후 그 결과를 비교한 바가 있

다. 가장 작은 단위로 나누어진 음절 단위의 모델의 학습이 가장 적은 시간이 걸렸으나 문장 생성 결과는 가장 큰 단위로 나누어진 어절 단위의 모델이 가장 뛰어난 것으로 나타났다. 또한 N-gram 모델, 기본 순환신경망 모델에 비해 LSTM 모델이 가장 좋은 정량적 지표를 보여주었다.

한글의 각 문자는 초성, 중성, 종성으로 이루어진 음소의 조합으로 이루어져 있기 때문에 조합 가능한 문자의 수가 다른 영미권 언어에 비해 많다. 그러므로 문자 단위의 모형 역시 높은 복잡도를 가지고 있을 가능성이 있다. 본 논문에서는 한글로 이루어진 텍스트를 문자보다 더 작은 단위인 자음과 모음으로 이루어진 음소 단위로 분할하여 사전을 구성하고 이를 기반으로 문장 생성 시스템을 구축한다.

3. 순환신경망 모델

3.1 기본 순환신경망(RNN) 모델

RNN 모델은 문장이나 음성과 같이 순차적인 정보를 처리하는데 적합한 모델이다. 기존의 신경망은 각 입력 데이터가 독립이라고 가정하지만 문장을 입력 데이터로 사용하는 경우 t시점의 입력값은 $t-1$시점을 비롯한 과거의 입력값들로부터 독립적일 수 없다. 순환 신경망은 은닉노드(hidden node)를 재귀적으로 모형화 함으로써 이러한 구조를 표현하였다. Figure 1의 왼쪽 그림이 나타내는 재귀적 형태를 각 시점에 따라 풀어내면 오른쪽 그림과 같이 표현된다.

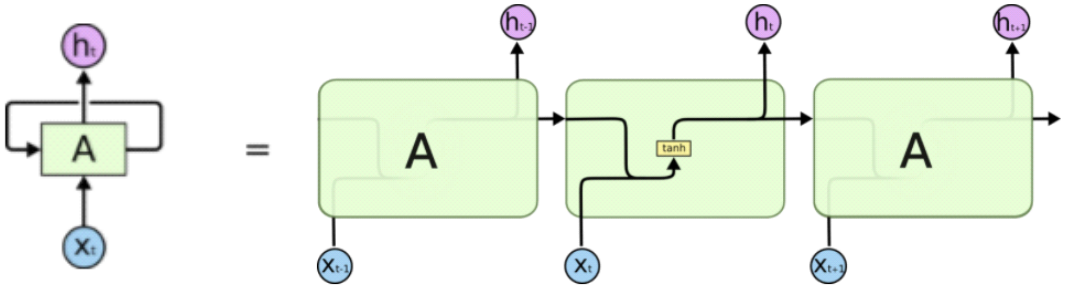
은닉층과 출력층의 값은 아래의 형태로 계산된다.

$$h_t = f(Ux_t + Wh_{t-1} + b)$$

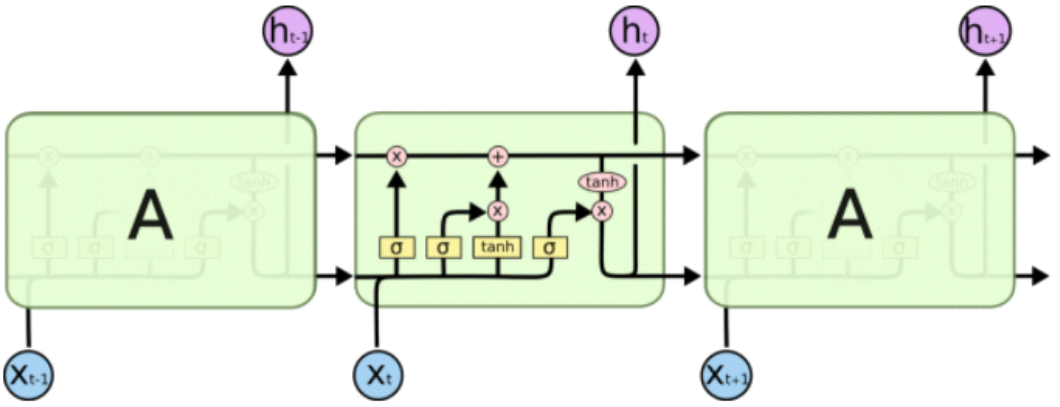
$$o_t = g(Vh_t + c)$$

위의 식에서 t시점의 은닉노드 값(h_t)은 t시점의 입력값(x_t)과 $t-1$ 시점의 은닉노드 값(h_{t-1})의 선형결합을 활성화 함수(activation function) f를 통해 나타낸다. 활성화 함수 f로 tanh, ReLU와 같은 비선형 함수가 주로 사용된다. 출력값 o_t는 t시점의 은닉노드의 선형함수를 softmax와 같은 활성화 함수 g를 통해 0과 1사이 값으로 변환시켜 예측한다. 위의 모형의 모수인 가중치 행렬 U, V, W과 bias인 b, c는 역전파(back propagation) 알고리즘을 통해 추정된다.

Figure 1의 오른쪽 그림을 반복적으로 확장하



〈Figure 1〉 Recurrent neural network (Olah, 2015)



〈Figure 2〉 Long Short-Term Memory network (Olah, 2015)

면 RNN 모델에 의한 예측은 무한한 과거 시점 관측값에 의존(long term dependency)한다는 것을 알 수 있다. 이는 가중치를 추정 하기 위한 목적함수의 편도함수 값이 0에 가까워 지거나(vanishing gradient) 매우 큰 값(exploding gradient)이 되는 문제를 발생시킬 수 있다(Goodfellow et al., 2016). 편도함수가 0에 가까워지면 목적함수를 최적화 시키기 위해 모수벡터가 움직여야 하는 방향을 찾기 어렵고 편도함수가 지나치게 큰 값이면 학습이 불안정해지는 문제가 있다.

3.2 Long short-term memory (LSTM)

위에서 기술한 RNN의 문제점을 해결하기 위한 하나의 방법으로 제안된 LSTM 모델(Hochreiter and Schmidhuber, 1997)은 RNN의 일종으로써 내부노드를 메모리셀(memory cell)이라 불리우는 형태로 대체하여 오랜 기간동안 정보를 축적하거나 이전 정보를 잊을 수 있도록 고안된 개폐장치를 사용한다.

$$\tilde{c}_t = \tanh(x_t U^g + h_{t-1} W^g + b_c)$$

$$c_t = c_{t-1} \circ f_t + \tilde{c}_t \circ i_t$$

$$h_t = \tanh(c_t) \circ o_t$$

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i + b_i\right)$$

$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f + b_f\right)$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o + b_o)$$

LSTM의 구조는 RNN의 구조와 유사하지만 내부노드를 구성하는 형태가 Figure 2와 같이 상대적으로 복잡한 형태를 띤다. 동일시점의 입력값(x_t)과 이전 시점의 은닉노드값(h_{t-1})의 선형결합으로 내부기억노드의 후보($\tilde{c}_t$)를 계산하고 이를 이전 시점의 기억노드 값(c_{t-1})과 결합하여 현 시점의 내부기억노드 값(c_t)을 계산한다. 이때 0과 1사이의 값을 가지는 input gate(i_t)와 forget gate(f_t)가 얼만큼의 새로운 정보를 통과시킬 것인지 혹은 이전 상태값의 정보를 얼만큼 통

과시킬 것인지를 조정하는 가중치의 역할을 한다. 마지막으로 tanh 활성화 함수를 통한 현 시점의 내부기억노드 값을 얼만큼 외부 네트워크로 통과시킬 것인지 output gate(o_t)에 의해 조절한 뒤 은닉노드 값을 출력한다. 각 input gate, forget gate, output gate 값은 위의 식과 같이 현 시점의 입력값과 이전 시점의 은닉노드 값의 선형결합으로 표현된다. 여기서 W, U들은 가중치를 포함한 매트릭스들이고 ∘는 벡터의 각 요소 간의 곱을 의미한다.

3.3 최적화 알고리즘

모형으로부터 예측된 출력값이 실제 데이터를 설명하는데 얼만큼 비효율적인지를 cross-entropy 비용함수를 통해 수량화하고 이를 최소화 하는 모수를 학습을 통해 찾는다. 총 길이 T를 갖는 데이터에서 유일한 음소들의 모임인 사전의 크기(vocabulary size)를 $|V|$라고 하자. 시점 t에 j번째 음소가 나타나면 $y_{t,j}$를 1, 나타나지 않으면 0으로 코딩하고 이에 대한 예측 확률값을 $\hat{y}_{t,j}$로 표현하면 t시점에 대한 cross-entropy 비용함수는 아래와 같다(Socher and Mundra, 2016).

$$J^{(t)}(\boldsymbol{\theta}) = -\sum_{j=1}^{|V|} y_{t,j} \log(\hat{y}_{t,j}) \qquad (1)$$

여기서 θ는 모형에서 추정하는 모수들을 벡터로 표현한 것이다. 모든 시점의 데이터의 비용함수를 합한 뒤 모수 θ에 대해 편미분한 gradient 함수 $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$를 아래와 같이 표현할 수 있다.

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} J^{(t)}(\boldsymbol{\theta}) \qquad (2)$$

Gradient descent 알고리즘(Cauchy, 1847)은 식(1)에 표현된 비용함수가 가장 빠르게 감소하는 방향인 $-\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ 방향으로 학습률 ε 만큼씩 이동을 반복하여 $\boldsymbol{\theta} \longrightarrow \boldsymbol{\theta} - \epsilon \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$의 형태로 비용함수의 최소값을 찾아가는 방법이다. 식(2)에서 나타나는 바와 같이 gradient는 모든 데이터를 사용하여 계산되는데 이로 인해 알고리즘의 속도가 느려진다. 이를 개선하기 위해 기본 gradient descent 알고리즘 보다는 아래의 stochastic gradient descent 알고리즘이 주로 사용된다.

- Stochastic gradient descent

Stochastic gradient descent(SGD) 알고리즘은 gradient 함수가 기대값이라는 데 착안 하여 모든 관측치를 사용하여 기대값을 계산하지 않고 매 단계에서 무작위로 선택된 실험 데이터의 부분집합(minibatch)을 사용하여 기대값을 근사한다. 적은 수의 관측치로 gradient 함수 값을 계산하므로 기존의 gradient descent 알고리즘에 비해 속도가 훨씬 빠르고 매 단계에서 계산되는 비용함수의 변동성이 크기 때문에 알고리즘이 극소값에 매몰되지 않고 더 작은 비용함수를 가지는 모수 공간으로 이동할 가능성이 높아진다. SGD 알고리즘은 비용함수의 최적화 방법 중 가장 기본이 되는 방법이지만 학습률 선택이 어렵고 국소적 극소값(local minimum)으로 수렴이 가능하다는 문제를 가지고 있다. 이를 해결하기 위한 방법으로 SGD에서 학습률 선택방법을 개선하거나 gradient 함수의 계산을 개선하는 방법으로 아래의 다양한 알고리즘이 제안되었다.

- Adagrad

SGD 알고리즘에서는 모든 모수에 대해 동일

한 학습률을 적용하는 반면 adagrad 알 고리즘(Duchi et al., 2011)은 모수 별로 학습률을 조정한다. 각 모수에 대해 이전 단계의 gradient 제곱합의 제곱근에 반비례하는 학습률을 적용한다. 이를 통해 데이터가 희소할 수록 높은 학습률을 적용하게 되어 고차원 데이터에서 문제가 되는 희소 데이터 집합에 대해 덜 민감하게 반응하는 경향이 있다(Dean et al., 2012).

- RMSprop

RMSprop(Hinton et al., 2012)는 볼록이 아닌(non-convex) 비용함수에서 더 잘 작동 하도록 adagrad를 개선한 것으로 이전 단계 gradient 제곱들을 지수적으로 감소하는 가중치로 이동평균을 취하여 이 값의 제곱근에 반비례하도록 학습률을 계산한다. Adagrad는 과거의 모든 gradient 값을 사용하는 데 비해 RMSprop는 최근의 값들에 많은 가중치를 주어 계산을 하기 때문에 비용함수의 볼록한 부분을 찾은 후에는 빠르게 수렴하는 경향이 있다.

- Adadelta

Adadelta(Zeiler, 2012) 알고리즘은 adagrad의 확장으로 모든 과거 gradient 값의 제곱평균이 아니라 고정된 구간에 해당하는 과거값들만을 사용하여 제곱평균을 취하고 이 값의 제곱근에 반비례 하도록 학습률을 계산한다. 이 방법은 학습률에 대한 초기값을 지정할 필요가 없다는 장점이 있다.

- Adam

Adam(Kingma and Ba, 2014)은 학습률을 개선하기 위한 RMSprop과 gradient 값을 계산하는 방법을 개선하기 위한 momentum 알고리즘을 결합한 형태이다. Momentum(Polyak, 1964)은 지수적으로 감소하는 가중치를 사용하여 과거의 gradient 값들의 이동평균 값을 사용하여 SGD알고리즘의 $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$를 대체하는 방법이다. Adam 알고리즘에서 gradient는 momentum과 같이 이전 gradient의 이동평균으로, 학습률은 RMSprop와 마찬가지로 gradient 제곱의 이동평균에 반비례하도록 설정된다. 이 알고리즘은 일반적으로 hyperparameter의 선택에 덜 민감하다고 알려져 있다(Goodfellow et al., 2016).

- Adamax

Adamax는 Adam 알고리즘에서 학습률을 L_2 norm이 아닌 L_p norm에 반비례 하도록 확장하는 방법에서 착안된 것이다. 일반적으로 p가 커짐에 따라 알고리즘이 불안해 지는 경향이 있는데 $p = \infty$ 인 경우, 즉 gradient 절대값의 최대값에 반비례 하도록 설정되었을 때 알고리즘이 간단하면서도 안정적이라고 알려져 있다(Kingma and Ba, 2014). 이 특수한 경우인 L_∞를 사용한 해당 알고리즘을 Adamax라고 일컫는다.

- Nadam

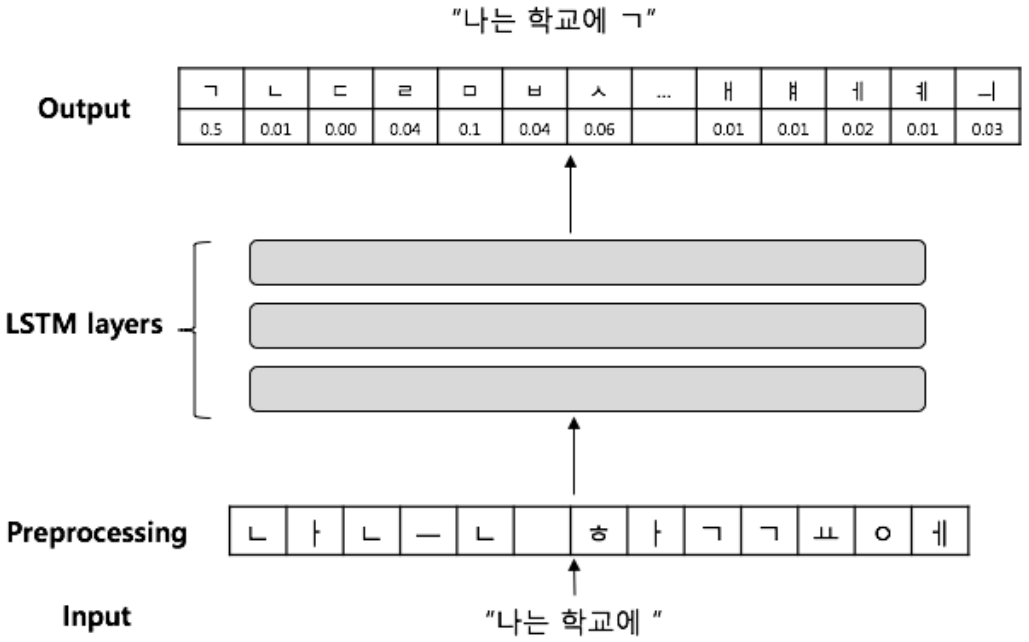
Nadam(Dozat, 2015) 방법은 adam 과 같이 gradient 값과 학습률을 함께 조정하는 방법으로 adam이 gradient 조정을 위해 momentum 알고리즘을 사용한 것과 달리 Nesterov's accelerated gradient 방법을 사용한다. Nesterov's accelerated gradient 알고리즘은 gradient 계산 이전에 모수를 업데이트 시킴으로써 더 나은 방향으로 알고리즘을 진행시킨다.

4. 음소 단위 문장생성 모형

본 논문에서는 불완전한 문장을 음소 단위로 분해하여 입력하여 뒤에 나올 음소를 예측하는 방식으로 문장을 생성하는 모형을 구축한다. 실험 데이터 상의 문장을 음소 단위의 문자로 분할하여 유일한 문자들로 이루어진 사전을 구축하고, n 개의 문자로 이루어진 벡터 $c = (c_1, c_2, ..., c_n)^T$를 입력하였을 때 사전의 각 문자가 c_{n+1} 이 될 확률을 계산하여 가장 확률이 높은 문자를 출력한다.

Figure 3은 전반적인 문장생성 처리 과정을 그림으로 나타내고 있다. 불완전한 문장인 '나는 학교에'를 입력값으로 사용하면 이를 음소단위로 분할하여 'ㄴㅏㄴㅡㄴ ㅎㅏㄱㄱㅛㅇㅔ'와 같이 표현한다. 실험 데이터에 포함된 문자들의 집합으로 구성된 사전에서 입력 값의 각 음소를 one-hot vector로 변환한 후 LSTM 계층들로 구성된 모델에 입력한다. 학습을 통해 사전의 각 음소에 대해 발생 확률을 계산하여 가장 높은 확률을 가지는 음소인 'ㄱ'을 출력한다. 이 과정을 사전에 설정한 횟수만큼 반복하여 원하는 길이의 문장을 생성한다.

음소 단위의 문장생성 모형은 단어, 형태소 단위의 문장생성 모형에 비해 사전의 크기(vocabulary size)가 현저히 적어 사전의 크기를 제한하여 빈도가 작은 단어를 제외하거나 embedding layer를 사용하는 등의 작업을 진행할 필요가 없고 예측의 불확실성이 낮아지는 효과가 있다. 반면 동일한 길이의 문장을 생성하기 위해 더 많은 단계를 거쳐야 하므로 LSTM memory cell 구조를 통해 장기 기억을 모델에 반



〈Figure 3〉 Example of sentence generation process

영할 필요가 있다.

본 논문에서는 3개와 4개의 LSTM 계층을 포함한 모델을 사용하여 3.3절에서 소개한 최적화 방법들을 사용하여 학습시키고 그 결과를 비교한다. 학습된 모델의 비교는 자동으로 생성된 문장의 완성도를 정성적으로 평가하는 방법과 정량적인 지표를 사용하여 비교하는 방법이 있다. 정량지표로 식 (1)에 표현된 cross entropy 비용값을 사용한다. 이와 함께 $2^{J(\theta)}$로 정의되는 perplexity를 사용한다. Perplexity는 예측의 명료성에 대한 지표로서 값이 작을 수록 다음 문자를 예측하는데 있어서 높은 확률을 가지고 확실하게 예측하는 것을 의미한다.

5. 실험

실험에서 사용된 데이터는 구약성경 텍스트이다. 전체 데이터를 음소 단위로 분할하는 전처리 과정을 거쳐 자음, 모음, 숫자, 괄호, 따옴표 등이 포함된 74개의 문자로 이루어진 데이터를 구축하였다. 연속된 20개의 음소로 이루어진 입력 데이터와 뒤이어 나오는 21번째 음소로 이루어진 출력 데이터로 총 1,023,411개의 입출력 데이터를 구성하였고 이를 70:15:15의 비율로 training, validation, test set으로 나누었다. 모든 실험은 Intel Xeon CPU 1개(16 코어)와 NVIDIA GeForce GTX 1080 GPU 1개가 장착된 PC에서 Theano(Theano Development Team, 2016)를 백그라운드로 설정한 Keras(Chollet, 2015) 패키지를 활용하여 진행되었다. 모델은 각 데이터 당 최대 100개의 epoch를 500개의 batch size를 사용하여 학습되었으며 validation loss 값이 더 이상 감소하지 않는 epoch 수에서 학습을 멈추었다. 최종 epoch 수는 Table 1과 2에 나타나 있다. 앞서 소개한 7개의 서로 다른 최적화 알고리즘을 사용하여 각 3개와 4개의 LSTM layer를 포함한 모형을 학습하였다. 각 LSTM layer는 512개의 output unit으로 구성하였고 과적합을 방지하기 위해 (Srivastava et al., 2014) 매 단계에서 40%의 unit을 무작위로 dropout 하였다.

5.1 Validation loss와 perplexity

두 모형과 7개의 최적화 알고리즘에 대한 validation loss값과 perplexity가 오름차순으로 정렬되어 Table 1과 Table 2에 수록되어 있고

〈Table 1〉 Simulation statistics for the model with 3 LSTM layers

Optimizer	Num. of epochs	Loss	Perplexity	Time (in hours)
nadam	31	1.06	2.88	2.34
adam	20	1.07	2.92	1.67
adamax	20	1.08	2.95	1.48
rmsprop	22	1.09	2.97	1.85
adadelta	39	1.13	3.10	3.13
adagrad	99*	1.14	3.12	7.65
sgd	99*	1.96	7.10	8.03

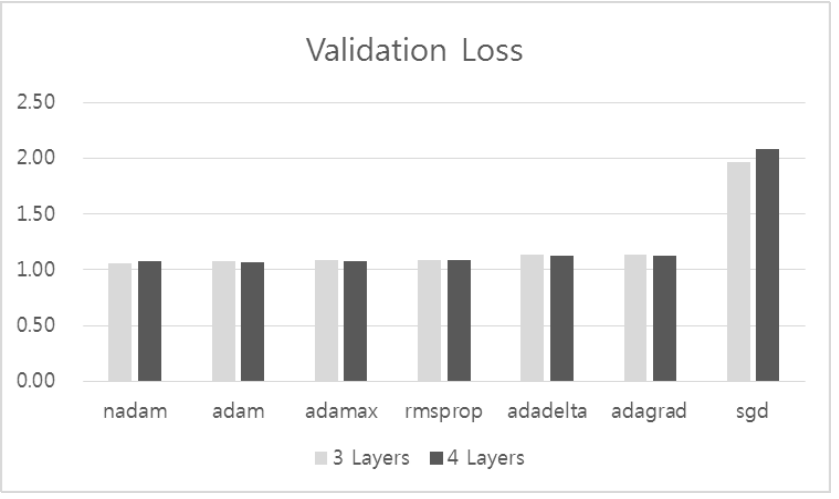
*: optimal validation loss is not attained within 100 epochs

〈Table 2〉 Simulation statistics for the model with 4 LSTM layers

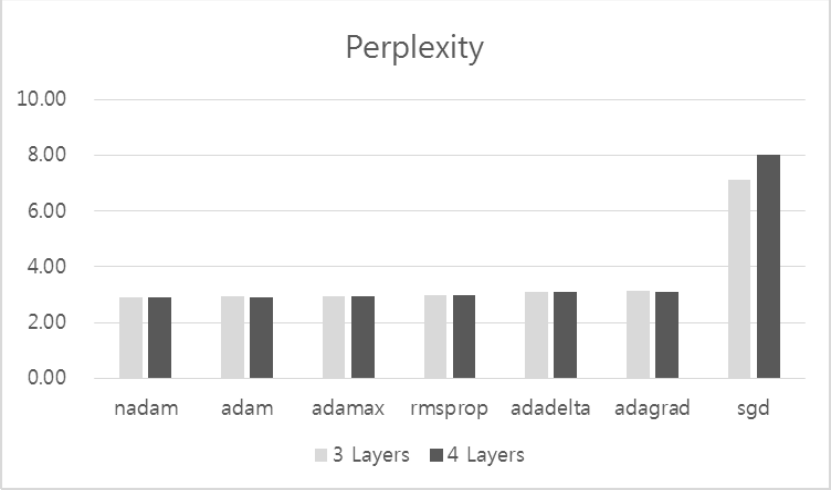
Optimizer	Num. of epochs	Loss	Perplexity	Time (in hours)
adam	30	1.07	2.91	3.05
nadam	57	1.07	2.91	5.78
adamax	27	1.08	2.93	2.75
rmsprop	34	1.09	2.98	3.81
adadelta	45	1.13	3.08	4.41
adagrad	74	1.13	3.08	7.54
sgd	99*	2.08	8.03	9.65

*: optimal validation loss is not attained within 100 epochs

Figure 4와 Figure 5는 이를 그래프로 비교하고 있다. Figure 4를 보면 SGD 방법을 제외한 나머지 최적화 알고리즘들은 대체로 비슷한 loss 값을 가지는 것으로 보이며 SGD 알고리즘은 다른 알고리즘들에 비해 확연히 큰 loss 값을 가지는 것을 알 수 있다. 특히 $\boldsymbol{\theta} \longrightarrow \boldsymbol{\theta} - \epsilon \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ 의 최적화 과정에서 학습률(ε)만 개선하기 보다는 $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$의 계산을 함께 개선한 알고리즘인 nadam, adam, adamax가 두 모형에서 모두 상대적으로 좋은 성능을 나타낸다. SGD 알고리즘을 사용할 때 layer 4개를 사용한 모형이 layer 3개를 사용한 모형보다 더 큰 validation loss 값을 가진



〈Figure 4〉 Loss calculated for validation set



〈Figure 5〉 Perplexity calculated for test set

다. Table 3은 layer가 4개인 모형을 사용할 때 layer가 3개 모형을 사용할 때에 비해 정량지표가 상대적으로 몇 % 증가 혹은 감소했는지 나타낸다. SGD를 제외한 나머지 최적화 알고리즘은 변화량이 1% 내외로서 두 모형 간의 차이가 거의 없으나 SGD 알고리즘을 사용하면 6.2% 가량 loss가 증가하는 것을 볼 수 있다.

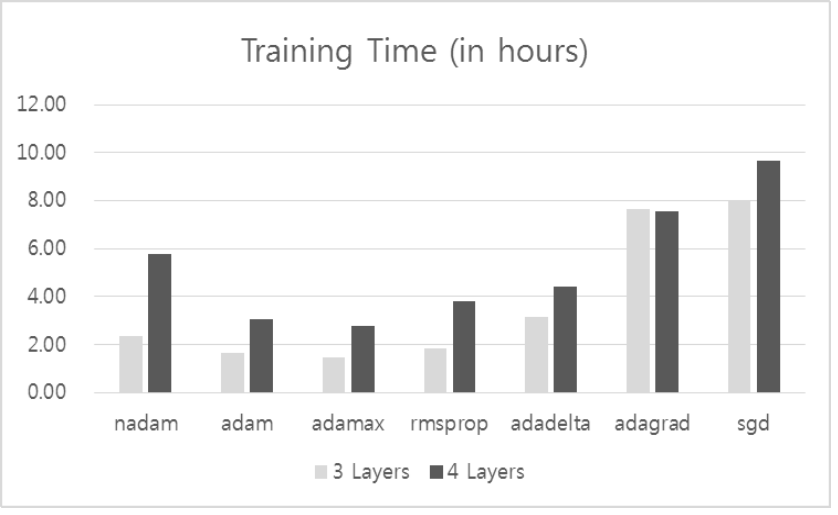
예측의 불확실성에 대한 지표인 perplexity를 test set을 사용해 계산한 결과는 validation loss와 비슷한 패턴을 보인다. Figure 5를 보면 SGD를 제외한 나머지 알고리즘들은 비슷한 perplexity값을 보이고 Table 3을 보면 layer 3개 모형과 layer 4개 모형 간의 차이가 1% 내외이다. 하지만 SGD 알고리즘은 다른 알고리즘들에 비해 확연히 큰 복잡도를 보이고 이는 layer 4개 모형을 사용할 때 13.1% 가량 증가하는 것으로 보인다. 이를 통해 4개의 LSTM layer를 포함한 모델이 가진 복잡성이 과적합을 발생시켜 오히려 높은 validation loss와 perplexity를 가져온 것으로 판단할 수 있다.

〈Table 3〉 Comparison of 4 LSTM-layer model vs. 3 LSTM-layer model. (% of increment)

Optimizer	Loss	Perplexity	Time
nadam	1.1	1.0	146.7
adam	-0.4	-0.4	83.0
adamax	-0.5	-0.6	85.6
rmsprop	0.0	0.4	105.6
adadelta	-0.7	-0.7	40.9
adagrad	-1.1	-1.3	-1.3
sgd	6.2	13.1	20.2

5.2 학습시간

Figure 6는 Table 1과 2에 포함된 학습시간을 그래프를 통해 비교하고 있다. 대부분의 최적화 알고리즘을 사용할 때 4개의 LSTM layer로 구성된 모형이 3개의 LSTM layer로 구성된 모형보다 학습하는데 오랜 시간이 걸렸다. Table 3의 Time 열을 보면 adagrad 알고리즘만 음의 값(-1.3%)을



〈Figure 6〉 Total time to train the model

보여 4개의 layer를 사용할 때 오히려 학습시간이 적게 걸렸다. Nadam 알고리즘은 146.7% 증가분을 보여 4개의 layer를 사용할 경우 두 배 이상의 학습시간이 소요되었다. 7개의 알고리즘에 대한 결과를 총합해 봤을 때 4개의 layer를 사용할 경우 3개의 layer를 사용할 때 보다 평균적으로 약 69%의 학습시간이 더 필요했다.

5.3 정성평가

Validation loss와 perplexity가 가장 작은 알고리즘을 최적 모형으로 간주하여 layer 3개 모형에서는 nadam 알고리즘, layer 4개 모형에서는 adam 알고리즘을 사용한 결과를 최종으로 선택하였다. 학습된 결과를 정성적으로 판단하기 위해 '여호와여 주는 나의'를 입력값으로 설정하여 1000개의 음소를 뒤이어 생성하였다. 생성된 결과에서 처음으로 장 이름(예를들면, 창1:1)이 나타나고 해당 절이 끝나는 부분까지를 발췌하여 Table 4에 수록하였다. Temperature 값으로 0.2와 0.8를 고려하였다. Temperature는 0과 1사이의 값으로 작을 수록 다소 보수적이지만 그럴듯한 텍스트를 생성할 가능성이 높고, 클수록 결과가 다양하지만 오류가 포함될 가능성이 높다.

Table 4에서 보여주는 모든 시뮬레이션 조건에서 한글에 사용되지 않는 형태의 문자가 전혀 생성되지 않았고 앞 뒤 문맥이 완벽히 맞지는 않지만 조사와 명사의 조합이나 동사의 활용, 문장

〈Table 4〉 Sentence generated by the best algorithm for each model

Model	Temperature	Generated sentence
Layer 3	0.2	여호와여 주는 나의 기뻐하는 자가 없으며 그 이름을 인도하셨으니 그들이 이르되 여호와께서 나를 위하여 이르기를 내가 나를 멸하시는 것이 있으며 그 앞에 있는 자가 나를 위하여 그 모든 일을 행하였음이라 사35:11 그러므로 이스라엘 자손이 다 이스라엘 자손을 위하여 그 앞에 있는 자가 없으면 그 중에 그 이름을 이스라엘 자손에게 명령하신 대로 내 아버지의 집에 있는 자가 있으며
	0.8	여호와여 주는 나의 이로 도망하게 하소서 내가 심히 성결한 언약을 받았으며 가축을 분향하리라 하였느니라 욥7:22 여호와의 명령을 긁게 하며 무릇 수송아지와 익제는 다 네게 말하기를 네가 어찌하여 화목제물의 율례를 기억하였으므로 나단이 그들을 이스라엘 자손의 분노를 그의 집에서 인도하여 내신 것이라 그들이 너 반역하고 골짜기에 거주하였으면 여호와께서 아직 이스라엘 자손이 어찌 그리 암몬 자손을 사마에서 만나리니 이는 이스라엘아 다스리느냐 이에 견백하였더라
Layer 4	0.2	여호와여 주는 나의 이름을 위하여 그 일을 알게 하리라 겔26:1 <이스라엘 자손이 이르되 내가 이 모든 것을 일으키리니 그들이 이 모든 길을 가져다가 그 앞에 있는 이스라엘 자손이 다 그들을 사랑하였더라
	0.8	여호와여 주는 나의 피가 되매 그가 또 이르되 이스라엘 자손이 거기에 범괴하며 여호와께서 기름 부음을 받자 함이니라 창2:25 대저 그러하던 일이 이르되 왕이 행한 것을 왕의 재물을 단할 것이니라

의 끝맺음 등에서 대체적으로 문법 상의 오류가 발견되지 않았다. Layer 3개의 모형이 전반적으로 하나의 성경구절을 길게 생성하는 경향을 보인 반면 layer 4개의 모형이 일반적인 성경 구절의 길이와 유사한 구절을 생성하였다. Layer 3개의 모형이 생성한 문장은 길이가 길어지면서 다소 중언부언하는 형태를 보이나 layer 4개의 모형이 생성한 문장은 '여호와여 주는 나의 이름을 위하여 그 일을 알게 하리라'와 같이 상당히 완성도가 높은 문장을 생성하였다.

6. 결론

본 논문에서는 한글 텍스트의 음소 단위 문장 생성모델을 제안하였다. 다양한 최적화 알고리즘과 LSTM layer의 개수에 따른 모형의 성능을 비교하였다. SGD 알고리즘을 사용할 때 다른 알고리즘들에 비해 확연히 큰 validation loss와 perplexity 값을 보이며 예측의 불확실성이 크게 나타났고 나머지 알고리즘들은 대체적으로 비슷한 불확실성을 보였다. 미세한 차이기는 하지만 최적화 과정에서 학습률과 gradient의 계산 방법을 함께 개선한 알고리즘인 nadam, adam, adamax가 학습률만 개선한 알고리즘들 보다 우월한 것으로 나타났다.

Layer가 3개인 모형과 4개인 모형을 비교했을 때 모형학습을 위해 소비되는 시간이 layer가 4개인 모형을 사용할 때 평균적으로 69% 가량 길었을 뿐 아니라 validation loss나 perplexity와 같은 정량지표가 거의 변화가 없거나 SGD 알고리즘을 사용할 때는 오히려 더 악화되는 것을 볼 수 있었다. 하지만 구약성경 텍스트를 사용하여 실제로 모형을 통해 생성된 문장들을 비교해 봤을 때 layer가 4개인 모형이 3개인 모형에 비해 비교적 완성도 높은 결과를 보여주었다.

모형과 최적화 알고리즘 간의 미세한 차이는 있지만 본 논문에서 고려한 어떠한 시뮬레이션 조건에서도 한글에 사용되지 않는 문자가 생성되지 않았고 조사와 명사의 조합과 동사의 활용면에서 상당히 완성도 높은 문장이 생성되었다. 단어 단위나 형태소 단위의 문장생성 모형을 사용할 때에 비해 음소 단위의 모형을 사용하면 상대적으로 사전의 크기가 작아서 모형이 간소화되고 형태소 분석기를 사용하지 않아도 되기 때문에 불완전한 형태소 분해로 인한 오류를 방지할 수 있다. 하지만 동일한 길이의 문장을 생성하기 위해 더 긴 길이의 time step을 지나야 한다는 단점이 있는데 layer 4개의 모형으로 생성된 문장은 상대적으로 앞 뒤 문맥이 상당히 완성도 있는 것으로 보아 깊은 LSTM layer를 포함한 구조를 통해 이러한 단점이 어느 정도 해결되는 것으로 보인다.

이러한 문장 자동생성 모형은 현재 대두되고 있는 다양한 인공지능 시스템의 기초가 되는 음성인식과 언어처리 분야의 기반이 되는 기술로서 영어권에 비해 아직 연구가 부족한 한국어 기반 시스템을 만드는데 활용될 수 있을 것으로 기대된다. 본 논문이 제시하는 모형을 보다 발전시켜 word-to-character, character-to-word 단계를 사용하여 음소 단위의 모형과 단어 단위 모형의 장점을 결합한 모형을 사용하는 등의 좀더 발전된 기법을 사용한다면 더욱 자연어에 가까운 문장을 생성하는 모형을 구축할 수 있을 것으로 보인다.

참고문헌(References)

Ahn, S. "Deep Learning Architectures and Applications." *Journal of Intelligence and Information Systems*, Vol. 22, No. 2 (2016), 127~142.

Bojanowski, P., Joulin, A., and Mikolov, T. "*Alternative Structures for Character-Level RNNs*." arXiv:1511.06303 (2015).

Cauchy, A. "M ′ethode g ′en ′erale pour la r′ esolution des systemes d'′equations simultan ′ees." *Comp. Rend. Sci. Paris,* Vol.25 (1847), 536~538.

Chollet, F. "*Keras*." Available at https://github.com/fchollet/keras (downloaded 1 December, 2016).

Chung, J., Cho, K., and Bengio, Y. "*A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation*." arXiv:1603.06147 (2016).

Olah, Christopher. "*Understanding LSTM Networks.*" Colah's Blog. Available at http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (downloaded 1 December, 2016).

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. "Large Scale Distributed Deep Networks." In *Advances in neural information processing systems*, (2012), 1223~1231.

Dozat, T. "Incorporating Nesterov Momentum into Adam." Technical report, Stanford University, Available at http://cs229. stanford. edu/proj2015/054 report.pdf (2015).

Duchi, J., Hazan, E., and Singer, Y. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." *Journal of Machine Learning Research*, Vol. 12 (2011), 2121~2159.

Gers, F. A. and Schmidhuber, E. "LSTM Recurrent Networks Learn Simple Context-Free and Context-Sensitive Languages." *IEEE Transactions on Neural Networks*, Vol. 12, No. 6 (2001), 1333~1340.

Goodfellow, I., Bengio, Y., and Courville, A. "*Deep Learning.*" MIT Press, Massachusetts, 2016.

Hinton, G., Srivastava, N., and Swersky, K. "*Neural networks for machine learning.*" Coursera, video lectures, Available at https://www.coursera.org/learn/neural-networks (downloaded 1 December, 2016).

Hochreiter, S. and Schmidhuber, J. "Long Short-Term Memory." *Neural Computation*, Vol. 9, No. 8 (1997), 1735~1780.

Hutter, M. "The Human Knowledge Compression Prize." Available at http://prize.hutter1.net/ (2006).

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. "*Exploring the Limits of Language Modeling*." arXiv:1602.02410 (2016).

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. "*Character-Aware Neural Language Models*." arXiv:1508.06615 (2015).

Kim, Y.-h., Hwang, Y.-k., Kang, T.-g., and Jung, K.-m. "LSTM Language Model Based Korean Sentence Generation." *The Journal of Korean Institute of Communications and Information Sciences*, Vol. 41, No. 5 (2016), 592~601.

Kingma, D. and Ba, J. "*Adam: A Method for Stochastic Optimization*." arXiv:1412.6980 (2014).

Lankinen, M., Heikinheimo, H., Takala, P., and Raiko, T. "*A Character-Word Compositional Neural Language Model for Finnish*." arXiv:1612.03266 (2016).

Lee, D., Oh, Kh., and Choi, H.-J. "Measuring the Syntactic Similarity between Korean Sentences Using

RNN." In *Proceedings of Korea Computer Congress* (2016a), 792~794.

Lee, J., Cho, K., and Hofmann, T. "*Fully Character-Level Neural Machine Translation without Explicit Segmentation*." arXiv:1610.03017 (2016b).

Ling, W., Lu´ıs, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. "*Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation*." arXiv:1508.02096 (2015).

Mikolov, T., Karafi´at, M., Burget, L., Cernocky`, J., and Khudanpur, S. "Recurrent Neural Network Based Language Model." In *Proceedings of Interspeech* (2010), 1045~1048.

Mikolov, T. and Zweig, G. "Context Dependent Recurrent Neural Network Language Model." *SLT* (2012), 234~239.

Polyak, B. T. "Some Methods of Speeding Up the Convergence of Iteration Methods." *USSR Computational Mathematics and Mathematical Physics*, Vol. 4, No. 5 (1964), 1~17.

Rissanen, J. and Langdon, G. G. "Arithmetic Coding." *IBM Journal of research and development,* Vol.23, No. 2 (1979), 149~162.

Socher, R. and Mundra, R. S. "*CS 224D: Deep Learning for NLP1*." Available at http://cs224d.stanford.edu/ (downloaded 1 December, 2016).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, Vol. 15, No. 1 (2014), 1929~1958.

Sundermeyer, M., Schlu¨ter, R., and Ney, H. "LSTM Neural Networks for Language Modeling." In *Proceedings of Interspeech* (2012), 194~197.

Sutskever, I. and Martens, J. "Generating Text with Recurrent Neural Networks." In *Proceedings of the 28th International Conference on Machine Learning* (2011), 1017~1024.

Theano Development Team. "*Theano: A Python Framework for Fast Computation of Mathematical Expressions*." arXiv:1605.02688 (2016).

Ward, D. J., Blackwell, A. F., and MacKay, D. J. "Dasher—a Data Entry Interface Using Continuous Gestures and Language Models." In *Proceedings of the 13th annual ACM symposium on User interface software and technology* (2000), 129~137.

Zeiler, M. D. "*ADADELTA: An Adaptive Learning Rate Method*." arXiv:1212.5701 (2012).

Abstract

Korean Sentence Generation Using Phoneme-Level LSTM Language Model

SungMahn Ahn* · Yeojin Chung** · Jaejoon Lee*** · Jiheon Yang****

Language models were originally developed for speech recognition and language processing. Using a set of example sentences, a language model predicts the next word or character based on sequential input data. N-gram models have been widely used but this model cannot model the correlation between the input units efficiently since it is a probabilistic model which are based on the frequency of each unit in the training set. Recently, as the deep learning algorithm has been developed, a recurrent neural network (RNN) model and a long short-term memory (LSTM) model have been widely used for the neural language model (Ahn, 2016; Kim et al., 2016; Lee et al., 2016). These models can reflect dependency between the objects that are entered sequentially into the model (Gers and Schmidhuber, 2001; Mikolov et al., 2010; Sundermeyer et al., 2012). In order to learning the neural language model, texts need to be decomposed into words or morphemes. Since, however, a training set of sentences includes a huge number of words or morphemes in general, the size of dictionary is very large and so it increases model complexity. In addition, word-level or morpheme-level models are able to generate vocabularies only which are contained in the training set. Furthermore, with highly morphological languages such as Turkish, Hungarian, Russian, Finnish or Korean, morpheme analyzers have more chance to cause errors in decomposition process (Lankinen et al., 2016).

Therefore, this paper proposes a phoneme-level language model for Korean language based on LSTM models. A phoneme such as a vowel or a consonant is the smallest unit that comprises Korean texts. We construct the language model using three or four LSTM layers. Each model was trained using Stochastic Gradient Algorithm and more advanced optimization algorithms such as Adagrad, RMSprop, Adadelta, Adam, Adamax, and Nadam. Simulation study was done with Old Testament texts using a deep learning

* School of Business Administration, Kookmin University
** Corresponding Author: Yeojin Chung
School of Business Administration, Kookmin University
77 Jungrungro, Sungbukku, Seoul 02707, Korea
Tel: +82-2-910-5614, E-mail: ychung@kookmin.ac.kr
*** Department of Data Science, Kookmin University
**** Department of Data Science, Kookmin University

package Keras based the Theano. After pre-processing the texts, the dataset included 74 of unique characters including vowels, consonants, and punctuation marks. Then we constructed an input vector with 20 consecutive characters and an output with a following 21st character. Finally, total 1,023,411 sets of input-output vectors were included in the dataset and we divided them into training, validation, testsets with proportion 70:15:15. All the simulation were conducted on a system equipped with an Intel Xeon CPU (16 cores) and a NVIDIA GeForce GTX 1080 GPU.

We compared the loss function evaluated for the validation set, the perplexity evaluated for the test set, and the time to be taken for training each model. As a result, all the optimization algorithms but the stochastic gradient algorithm showed similar validation loss and perplexity, which are clearly superior to those of the stochastic gradient algorithm. The stochastic gradient algorithm took the longest time to be trained for both 3- and 4-LSTM models. On average, the 4-LSTM layer model took 69% longer training time than the 3-LSTM layer model. However, the validation loss and perplexity were not improved significantly or became even worse for specific conditions. On the other hand, when comparing the automatically generated sentences, the 4-LSTM layer model tended to generate the sentences which are closer to the natural language than the 3-LSTM model. Although there were slight differences in the completeness of the generated sentences between the models, the sentence generation performance was quite satisfactory in any simulation conditions: they generated only legitimate Korean letters and the use of postposition and the conjugation of verbs were almost perfect in the sense of grammar. The results of this study are expected to be widely used for the processing of Korean language in the field of language processing and speech recognition, which are the basis of artificial intelligence systems.

Key Words : Language model, Recurrent neural network, Long short-term memory model, Sentence generation model

Received : April 3, 2017 Revised : June 15, 2017 Accepted : June 16, 2017
Publication Type : Regular Paper Corresponding Author : Yeojin Chung

저 자 소 개

안 성 만
현재 국민대학교 경영대학 경영학부교수로 재직하고 있다. 서울대학교 경영대학에서 학사를, George Mason University에서 정보기술학 박사학위를 취득하였다. Johns Hopkins University의 응용수학과에서 박사후과정을 보냈고 ㈜쌍용정보통신에서 근무하였다. 주요 관심분야는 통계적 방법론, 모형선택, 데이터마이닝, 뉴럴네트워크 등이다.

정 여 진
연세대학교에서 경제학 및 응용통계학 복수전공으로 학사를 취득하였으며, 동 대학원에서 응용통계학 석사학위를 취득한 후 Pennsylvania State University에서 통계학 박사학위를 취득하였다. 현재 국민대학교 경영대학 경영학부 조교수로 재직중이다. 주요 연구분야는 비모수분포추정, 모형기반 군집분석, hierarchical linear model, 일반화선형모형, 데이터마이닝 등이다.

이 재 준
현재 국민대학교 일반대학원 데이터사이언스학과에서 데이터사이언스전공 석사과정에 재학 중이며, 국민대학교 수학과에서 학사학위를 취득한 바 있다. 주요 관심 분야는 딥러닝(deeplearning)에 기반한 빅데이터 분석, 감성 분석이다.

양 지 헌
현재 국민대학교 일반대학원 데이터사이언스학과에서 데이터사이언스전공 박사과정에 재학 중이며,숭실대학교 물리학과에서 학사학위를 취득한 바 있다. 주요 관심 분야는 딥러닝(deeplearning)에 기반한 AI와 머신러닝이다. 전 VTW수석컨설턴트로 재직한 바 있으며 현재 글로벌텔레콤의 IOT분석팀장으로 일하고 있다.