논문 16-41-05-12 The Journal of Korean Institute of Communications and Information Sciences '16-05 Vol.41 No.05
http://dx.doi.org/10.7840/kics.2016.41.5.592

LSTM 언어모델 기반 한국어 문장 생성

김 양 훈◆, 황 용 근*, 강 태 관**, 정 교 민°

LSTM Language Model Based Korean Sentence Generation

Yang-hoon Kim◆, Yong-keun Hwang*, Tae-gwan Kang**, Kyo-min Jung°

요 약

순환신경망은 순차적이거나 길이가 가변적인 데이터에 적합한 딥러닝 모델이다. LSTM은 순환신경망에서 나타나는 기울기 소멸문제를 해결함으로써 시퀀스 구성 요소간의 장기의존성을 유지 할 수 있다. 본 논문에서는 LSTM에 기반한 언어모델을 구성하여, 불완전한 한국어 문장이 입력으로 주어졌을 때 뒤 이어 나올 단어들을 예측하여 완전한 문장을 생성할 수 있는 방법을 제안한다. 제안된 방법을 평가하기 위해 여러 한국어 말뭉치를 이용하여 모델을 학습한 다음, 한국어 문장의 불완전한 부분을 생성하는 실험을 진행하였다. 실험 결과, 제시된 언어모델이 자연스러운 한국어 문장을 생성해 낼 수 있음을 확인하였다. 또한 문장 최소 단위를 어절로 설정한 모델이 다른 모델보다 문장 생성에서 더 우수한 결과를 보임을 밝혔다.

Key Words : LSTM, Recurrent Neural Networks, Language Model, Sentence Generation

ABSTRACT

The recurrent neural network (RNN) is a deep learning model which is suitable to sequential or length-variable data. The Long Short-Term Memory (LSTM) mitigates the vanishing gradient problem of RNNs so that LSTM can maintain the long-term dependency among the constituents of the given input sequence. In this paper, we propose a LSTM based language model which can predict following words of a given incomplete sentence to generate a complete sentence. To evaluate our method, we trained our model using multiple Korean corpora then generated the incomplete part of Korean sentences. The result shows that our language model was able to generate the fluent Korean sentences. We also show that the word based model generated better sentences compared to the other settings.

Ⅰ. 서 론

최근 기계학습(Machine Learning) 연구 및 응용분야가 대폭 확대되는 가운데[1,2] 딥 러닝(Deep Learning)이 새롭게 주목받고 있다. 딥 러닝은 여러 은닉계층(Hidden Layer)으로 구성된 인공신경망(Artificial Neural Network)을 학습하고 이용하는 방법으로서, GPU(Graphics Processing Unit)와 같은 고성능 계산 하드웨어의 발달 및 풍부한 데이터의 축적과 동시에 여러 분야에서 기존 기계학습 기법을 뛰어

※ 본 연구는 2016년도 두뇌한국21플러스사업에 의하여 지원을 받아 수행되었습니다.
◆ First Author : Automation and Systems Research Institute (ASRI), Department of Electrical and Computer Engineering, Seoul National University, ad26kr@snu.ac.kr, 학생회원
° Corresponding Author : Automation and Systems Research Institute (ASRI), Department of Electrical and Computer Engineering, Seoul National University, kjung@snu.ac.kr, 정회원
* Department of Electrical and Computer Engineering, Seoul National University, wangcho2k@snu.ac.kr
** Department of Electrical and Computer Engineering, Seoul National University, zd9370@snu.ac.kr
논문번호 : KICS2015-12-388, Received December 10, 2015; Revised April 6, 2016; Accepted May 23, 2016

넘는 강력한 대안으로 부상하고 있다. 딥 러닝은 영상 인식[3]과 같이 기존 기계학습 분야에서 활발히 연구되고 있는 데이터 인식 분야는 물론이고, 헬스케어, 무선 센서 네트워크(Wireless Sensor Network)[4]등의 분야에도 최근 폭 넓게 적용되고 있다.

딥 러닝 기법 중 하나로 최근 연구가 활발하게 이루어지고 있는 순환신경망(Recurrent Neural Network; RNN)은 과거 시간 단계(Time Step)의 은닉상태(Hidden State)와 현 시간 단계의 시스템 입력값을 새로운 은닉 상태를 계산하는 데 사용하는 구조적 특징이 있다. 하지만 RNN의 기본적인 순환구조는 학습과정에서 연속된 입력값의 길이가 늘어남에 따라 축적되는 정보를 잃게 되는 기울기 소멸문제(Vanishing Gradient Problem[5]) 로 인해 응용적인 측면에서 뛰어난 성과를 보이지 못했으나, 학습 성능을 향상시키는 다양한 기법들이 개발되면서 다양한 분야에 널리 적용되기 시작했다. 특히 LSTM(Long-Short Term Memory)-RNN[6] 의 경우, RNN의 구조적 개선을 통해 학습 능력을 크게 향상시켜 연속적이고 길이가 긴 학습 데이터가 가지고 있는 순차적 의존성을 효과적으로 학습 할 뿐만 아니라 과거 시간 단계 정보의 중요도를 판별하여 얼마나 많은 과거 정보를 현재까지 끌어올 것인지, 혹은 현재 시간 단계의 정보를 얼마나 사용할 것인지 결정 할 수 있기 때문에 긴 시간 동안 나타나는 입력 데이터의 의존성을 기존의 RNN보다 더욱 효과적으로 학습하는 것이 가능하다. 그렇기 때문에 LSTM의 경우 길이가 긴 입력 데이터를 적용하는 작업에서 기존 RNN보다 좋은 성능을 보인다.

본 연구에서는 LSTM에 기반 한국어 언어 모델(Language Model)을 통해 불완전한 한국어 문장 혹은 단어가 주어졌을 때 이어지는 문장을 자동으로 생성할 수 있는 문장 생성 시스템을 구현한다. 언어 모델은 여러 개의 문서나 문장과 같은 해당 언어의 말뭉치(Corpus) 데이터를 학습하여, 주어진 단어의 배열이 해당 언어에 있어서 얼마나 인간의 언어와 유사한지를 표현할 수 있는 확률 모형이다. 이를테면, 적절하게 구현 및 학습된 한국어 언어 모델에서는 '밥을 먹다' 가 '먹다 밥을' 의 경우보다 높은 확률 값을 보일 것이다. LSTM에 기반한 언어모델은 주변 단어의 동시출현확률에 기반한 Unigram 언어모델 및 N-gram 언어모델보다는 문장에서 나타나는 단어 간의 순차적 의존관계를 보다 효과적으로 학습 할 수 있다. 이렇게 학습한 언어 모델을 바탕으로 문장의 일부가 주어졌을 때 문장의 나머지 단어들을 순차적으로 추론 및 생성하는 문장 생성 알고리즘을 구현하며, 본 논문에서는 실제 한국어 말뭉치를 이용한 언어모델 학습 및 문장 생성 실험을 통해 구현된 시스템이 말뭉치로부터 학습한 한국어의 특징에 맞는 문장을 생성하는 지를 살펴볼 것이다.

본 논문의 구성은 다음과 같다. 제2장에서는 순환신경망 및 순환신경망을 적용한 언어 모델 관련 연구들을 살펴볼 것이고, 3장에서는 순환신경망의 수학적 정의 및 훈련 방법에 대해 리뷰를 할 것이다. 4장에서는 본 연구에서 구현한 순환신경망 기반 언어 모델의 구조를 살펴볼 것이고, 5장에서는 실제 한국어 데이터를 활용하여 순환신경망 기반 언어모델을 사용한 문장 생성 실험을 진행한 결과를 설명한 뒤 마지막으로 6장에서 결론을 도출하려 한다.

Ⅱ. 관련 연구

은닉계층의 재귀적 연결(Recurrent Connectioin)구조로 인하여 RNN은 음성 인식[7], 필기체 인식[8], 제스처 인식[9] 과 같이 시간에 따라 연속성을 가지는 데이터 혹은 길이가 가변적인 데이터를 학습 및 표현하는 작업[10]에 적합하다. 텍스트 또한 연속된 문자와 단어의 집합으로 구성되어 있기 때문에 연속성을 가지는 데이터로 이해가 가능하며, 텍스트를 이해하거나 표현하는 순환신경망 알고리즘도 활발히 연구되고 있다.

RNN을 적용시킨 대표적인 연구 중 하나로는 언어 모델이 있다. 일반적으로 언어 모델은 주어진 단어의 배열에 대한 확률분포를 정의하는 것으로 말뭉치 데이터를 분석하여 언어 구조적으로 적합한 단어의 조합일수록 높은 확률 값을 가지도록 확률 모델을 정의하고 훈련시킨다. 기존 연구에서는 문서나 문장에 나오는 단어들을 n개씩 분절하고 등장 빈도를 분석하여 통계적 확률 모델로 사용하는 N-gram 모델이 주로 사용되었는데 여기에서는 단어열의 출현 빈도만이 반영될 뿐이고 인접한 단어들 간의 의존관계는 잘 표현하지 못하는 한계점을 가진다.

언어 모델은 단어를 추론하면서 문장을 생성하는 작업에도 사용될 수 있다. 실제로 통계기반 기계번역(Statistical Machine Translation; SMT)에서는 목적언어(번역문으로 작성될 언어)에 맞는 문장을 생성하기 위해 목적 언어의 말뭉치로 훈련된 언어 모델을 사용하며[11], 음성 인식의 경우에도 대상 언어에 맞는 언어 모델을 통해 음성 입력에 맞는 자연스러운 문장을 생성한다[12]. 따라서 이들 작업에서는 언어 모델의 성능이 전체 시스템의 성능에 큰 영향을 끼친다. 이와 관련된 주제로서 음성인식 연구 분야에서 RNN 기반

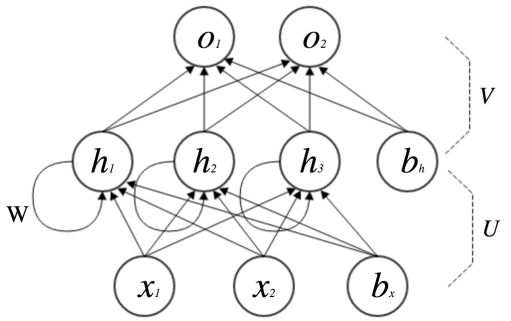

그림 1. RNN의 기본 구조
Fig. 1. Structure of vanilla RNN

언어모델이 기존 N-gram 모델 대비 인식 성능을 향상시키는 연구 결과[13] 가 발표된 바 있다.

한편 언어 모델을 단어 단위가 아닌 문자(Character) 단위로 고려하는 경우도 있는데, 실제로 [14] 에서는 글자 단위의 순환신경망 언어 모델을 정의하여 학습한 이후, 텍스트의 일부분만을 입력하여 나머지 텍스트를 생성하는 작업을 통해 성능을 입증하였다. 일반적으로 한 언어에서 문자의 개수가 단어의 개수보다 훨씬 적기 때문에 문자 단위로 언어 모델을 정의할 경우 단어 단위로 정의했을 때 보다 적은 개수의 변수로 순환신경망을 구성할 수 있는 특징이 있으나 한편으로는 같은 길이의 문장을 생성하는 데 있어 글자 단위로 생성할 경우 단어 단위로 생성할 때 보다 훨씬 긴 시간단계가 필요하기 때문에 단어 단위의 모델 대비 문장 생성에 대한 예측 정확도가 떨어지는 결과도 보인다[15].

Ⅲ. Recurrent Neural Networks

3.1 Traditional Recurrent Neural Networks

RNN 은 인공신경망의 일종으로 은닉계층의 유닛들이 방향성 사이클(Directed Cycle)을 포함하는 구조를 가진다[16]. 이러한 구조는 인공신경망으로 하여금 은닉계층에 저장되어 있는 과거의 정보들을 현재의 입력값과 결합하여 사용할 수 있게 함으로써 입력값 시퀀스에 대해 시퀀스 내부유닛간의 상호적인 정보를 어느 정도 유지 시킨 은닉변수 시퀀스로 비선형 변환을 시킬 수 있다. 그림 1 에서는 일반적인 RNN의 구조를 나타내고 있다.

그림 1의 작동 원리는 아래의 재귀식(Recurrence Equation) 으로 표현 할 수 있다:

$$h_t = g_1(x_t U + W h_{t-1} + b_x) \quad (1)$$

$$o_t = g_2(W h_t + b_h) \quad (2)$$

g1, g2 는 각각 Sigmoid, tanh 와 같은 비 선형 활성화 함수를 뜻하며, x_t 는 입력값, h_t 는 t 시간단계의 은닉변수(Hidden state), o_t 는 t 시간단계의 출력값, b는 바이어스(Bias), U, V, W 는 각각 x_t, h_t, h_{t-1} 의 가중치(Weight)를 뜻한다.

3.1.1 Forward Propagation

각각의 가중치가 모두 학습 되어있다는 가정 하에, 입력 시퀀스 X =(x_1, x_2, ... , x_T) 가 순차적으로 신경망에 입력되면 각각(1),(2) 식을 통해서 h_1, o_1, h_2, o_2,...,h_T, o_T 순으로 은닉변수 h_t 와 최종 출력값 o_t 가 출력된다.

3.1.2 Backward Propagation - BPTT (Back Propagation Through Time)

전통 인공신경망 학습에 적용되는 Back Propagation 알고리즘이 RNN의 재귀적 구조에 의해 BPTT[17] 으로 파생되었다. BPTT의 작동원리는 아래 그림 2에서 나타내고 있다.

BPTT 알고리즘은 가장 먼저 신경망의 각 유닛들을 입력 시퀀스의 시간단계 만큼 펼쳐서(Unfolding) 순환 구조를 순차적인 구조로 나타내며, 그 다음 일반적인 BP(Back Propagation)를 적용하여 각각의 가중치를 학습한다. 여기서 중요한 점은 그림 2에서도 알 수 있듯 각 유닛의 가중치는 시간단계에 따라 변하지 않는다.

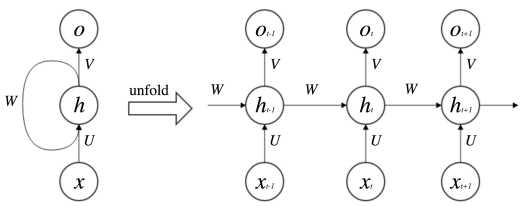

그림 2. BPTT에서의 레이어 펼침 과정
Fig. 2. Unfording process of BPTT

3.1.3 Vanishing Gradient Problem

입력 시퀀스의 길이가 늘어나게 되면 BPTT 알고리즘이 연쇄법칙(Chain Rule) 에 의해 계산된 기울기(Gradient)를 전파해야 하는 레이어의 수가 늘어나고 매 전파 시에 Sigmoid, tanh 와 같은 활성화 함수를 거쳐야 함에 있어 최종적으로 계산된 기울기가 극도

로 작아지는 기울기 소멸문제를 마주하게 된다. 기울기 소멸문제를 해결하기 위해 Non-Gradient 기반 방법으로 학습하는 SA(Simulated Annealing)[18], DEP(Discrete Error Propagation) 등 여러 가지 방법들이 제안되었고 그 중 LSTM 구조는 가장 효과적인 해결 방안을 제시했다.

3.2 LSTM

LSTM은 1997년 Hochreiter & Schmidhuber 에 의해 제안된 RNN 아키텍처이며 현재까지도 가장 주요적인 RNN 으로 자리 잡고 있다. LSTM 은 전통적인 RNN 구조에서 은닉계층의 유닛들을 LSTM 블록(Block) 으로 대치시킨 형태와 같다.

그림 4에서 나타내고 있는 LSTM 블록들은 기존의 은닉유닛(Hidden Unit) 들과 마찬가지로 재귀적 구조를 띄며, 그림 3에서 나타내고 있는 각각의 LSTM 블록 내부는 재귀적 구조를 가진 기억소자(Memory Cell)와 입력게이트(Input Gate), 잊기게이트(Forget Gate), 출력게이트(Output Gate) 3종류의 게이트유닛들로 이루어져 있다. LSTM은 전통적인 RNN과 마찬가지로 은닉변수를 거쳐 최종 출력값을 계산하지만, 은닉변수의 계산 과정에서 앞에 거론된 게이트유닛들을 적절하게 이용해서 정보의 흐름을 조절한다. 각각의 은닉변수의 유도과정은 다음과 같다: 가장 먼저 잊기게이트를 통해 기억소자에 저장되어 있는 기존의 소자변수(Cell State)를 얼마나 잊어버릴지 결정한다.

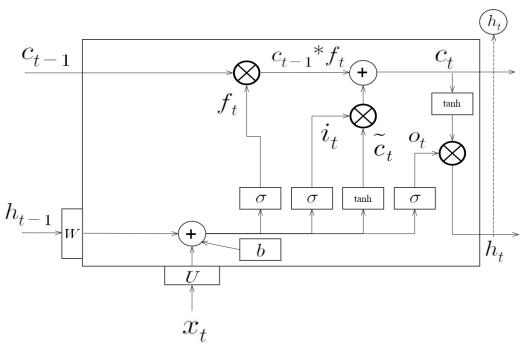

그림 3. LSTM 블록 구조
Fig. 3. Structure of LSTM block

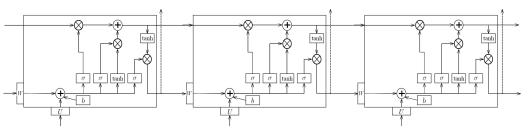

그림 4. LSTM 네트워크
Fig. 4. LSTM network

식(3)에서도 알 수 있듯 잊기게이트는 h_{t-1}, x_t 와 바이어스의 가중합(Weighted Sum)에 Sigmoid를 씌운 형태이며, 소자변수가 가지고 있는 각각의 정보에 대해 0 과 1 사이의 값을 계산한다. 여기서 1은 해당 위치의 정보를 전부 다 기억한다는 의미를 가지며, 0은 반대로 모두 잊어버리겠다는 의미를 가진다. 이렇게 계산된 값과 소자변수의 element-wise product는 현 시간단계의 계산에 사용될 과거의 소자변수를 나타낸다. 뒤이어 잊기게이트와 유사하게 입력게이트가 계산된다, 이름에서도 알 수 있듯 입력게이트는 얼마만큼의 새로운 정보를 소자변수로 가져갈지 결정하는 값이다. 여기서 말하는 새로운 정보도 마찬가지로 h_{t-1}, x_t 와 바이어스의 가중합을 통해 계산되며 tanh 활성화 함수를 통해(- 1, +1) 사이의 값으로 전환된다. 잊기게이트 와 입력게이트를 통해 조절된 과거의 정보와 새로운 정보는 식(6)과 같이 t 시간단계의 소자변수로 계산된다. 마지막으로 출력게이트는 필터링(Filtering)을 통해 계산된 소자변수의 유용한 정보를 최종 은닉변수로 출력하게 된다. LSTM 의 역방향 계산(Backward Pass)은 전통적인 RNN과 같이 BPTT를 통해 계산 된다.

$$f_t = \sigma(U^f x_t + W^f h_{t-1} + b^f) \tag{3}$$

$$i_t = \sigma(U^i x_t + W^i h_{t-1} + b^i) \tag{4}$$

$$\tilde{c_t} = \tanh(U^c x_t + W^c h_{t-1} + b^c) \tag{5}$$

$$c_t = f_t * c_{t-1} + i_t * \hat{c}_t \tag{6}$$

$$o_t = \sigma(U^o x_t + W^o h_{t-1} + b^o) \tag{7}$$

$$h_t = o_t * \tanh(c_t) \tag{8}$$

Ⅳ. 시스템 모델

본 논문에서는 입력값으로 불완전한 한국어 문장이 단어 단위로 주어졌을 때, LSTM을 통하여 뒤이어 나올 단어들을 예측하는 문제를 해결하려 한다. 입력값 w=(w_1, w_2, ... , w_T)가 입력계층(Input Layer)에 순차적으로 입력되었을 때 앞서 입력된 단어 배열을 통해 완전한 문장이 구성되기까지의 문맥(Context)를 학습하고 출력계층(Output Layer)에서 단어사전(Dictionary)의 모든 단어에 대해 매 단어가 불완전한

한국어 문장을 뒤 이을 단어 w_{T+1}로 나올 확률값을 출력한다.

$$P(x_{t+1}, x_t, \dots, x_2, x_1)$$
$$= P(x_{t+1}|x_1, x_2, \dots, x_{t-1}, x_t) \times P(x_1, x_2, \dots, x_{t-1}, x_t)$$
$$= P(x_{t+1}|x_1, x_2, \dots, x_{t-1}, x_t)$$
$$\times P(x_t|x_1, x_2, \dots, x_{t-2}, x_{t-1}) \times P(x_1, x_2, \dots, x_{t-2}, x_{t-1}) \quad (9)$$

최종적으로 계산된 확률값 중 가장 높은 확률값을 가진 단어 후보를 w_{T+1}로 출력하게 된다. 몇몇 높은 확률값을 가지는 후보들의 확률값이 큰 차이가 없는 등의 여러 상황을 적당히 고려하여 샘플링을 통해서 최종 단어 후보를 선정 할 수도 있다. 단어 예측에 사용된 LSTM 구조는 아래 그림 5와 같다.

입력계층과 출력계층의 유닛 수는 사전의 크기(Vocabulary Size)로 설정하여 각 단어가 순차적으로 입력 될 시 원-핫 인코딩(One-Hot Encoding)으로 표현되게 하였다. 입력계층을 뒤 잇는 임베딩계층(Embedding Layer)은 입력계층으로부터 전달받은 값을 특정 사이즈의 임베딩 공간(Embedding Space)로 맵핑하여 각각의 단어를 단어벡터(Word Vector)로 표현하게 된다. 이렇게 생성된 단어 임베딩(Word Embedding)은 원-핫 인코딩에 비해 단어의 의미적 요소와 단어 간 의 관계를 표현하는데 유용하다. 은닉계층은 총 3개의 동일한 차원의 계층으로 쌓여있으며, 다른 계층들과 달리 각각의 은닉유닛은 LSTM 블록으로 구성되어 있다. 또한 각각의 게이트의 연산에는 피폴 연결(Peephole Connection)을 더해주었다. 임베딩계층, 은닉계층의 사이즈는 휴리스틱한 방법으로 선정되었으며, 최종적으로 출력계층은 사전에 있는 모든 단어들에 대한 확률값을 출력한다.

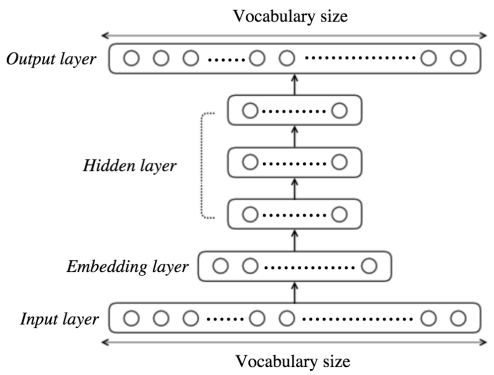

그림 5. 모델 구조
Fig. 5. Model structure

Input = "나는지금집에"
Preprocessing : "나", "는", "지금", "집", "에"
w_0 = "나", w_1 = "는", w_2 = "지금", w_3 = "집", w_4 = "에"
$\Rightarrow At\ t = 5,\quad P(w_5|$"나", "는", "지금", "집", "에")
$o_4 = \mathrm{argmax}\, P(w_5|$"나", "는", "지금", "집", "에") = "간다"
Input = "나는지금집에간다"
$\Rightarrow At\ t = 6,\quad P(w_6|$"나", "는", "지금", "집", "에", "간다")
$o_5 = \mathrm{argmax}\, P(w_6|$"나", "는", "지금", "집", "에", "간다") = "$< eos >$"
$If(o_t \equiv$ "$< eos >$") : *print* "나는지금집에간다"

그림 6. LSTM 기반 언어모델을 이용한 단어 생성 예제
Fig. 6. Example of sentence generation using LSTM language model

그림 6에서는 시스템의 전반적인 동작을 예제로 나타내고 있다. 입력 값으로 " 나는 지금 집에 " 라는 문장이 주어졌을 때 전처리를 통해 문장을 특정단위로 분할한다. 분할된 문장은 매 시간단계마다 순차적으로 시스템에 입력된다. 예제에서는 형태소 분석을 통한 전처리 과정 이후 x_0 : "나", x_1 : "는", x_2 : "지금", x_3 : "집", x_4 : "에" 와 같이 입력이 된다. 모델은 주어진 입력값 직후에 나타날 수 있는 단어 후보 중 가장 높은 확률을 가지는 후보를 t_4 시간단계의 출력값 o_4으로 출력하게 된다. 출력된 값은 다음 시간단계 t_5의 입력값으로 들어가게 되며 최종적으로 <eos>(End of Sentence)가 나올 때 까지 위의 과정을 반복하게 된다.

V. 실 험

5.1 실험 세팅

실험에서는 사용된 데이터는 성경 텍스트[19]와 뉴스 텍스트[20]이며 각각 문장 최소 단위를 어절, 형태소, 음절 단위로 분할하는 전처리 과정 거쳐 각각 최대

표 1. 모델 설정
Table 1. Model Settings

Data	Bible			News		
Sentences	22964			97123		
Preprocess	word	morpheme	character	word	morpheme	character
Vocabulary	28132	17129	1246	45085	19857	2203
Embedding Size	500/1000			1000		
Hidden Size	100					
Hidden stack	3					
mini-Batch	400					
Max sentence length	30	50	100	30	50	100

50,000, 20,000, 3000개의 최대 어휘 개수를 가지도록 설정 하였다. 그 중 형태소 분석은 Twitter 형태소 분석기[21]를 사용하였다. 출현빈도가 1인 문장 최소 단위 혹은 최대 어휘사이즈를 벗어난 문장 최소 단위들은 "**unknown**"으로 처리하였으며 "**unknown**"의 출현 빈도가 높은 상황에 대비하여 문장 생성 과정에서는 "**unknown**"을 제외 처리 했다. 실험에 사용된 각 모델의 설정은 다음과 같다:

제시된 LSTM 모델 및 문장 생성 시스템은 Python 언어 및 Theano 라이브러리[22,23]를 이용하여 작성하였으며, 모든 실험은 Intel Xeon CPU 2개(32 Cores) NVIDIA GTX TITAN X GPU 1개가 장착된 PC에서 진행하였다.

5.2 실험 결과

5.2.1 학습 시간

모델은 각 데이터 당 최대 60 epoch를 mini-batch를 이용하여 학습되었으며 수렴되기까지의 학습 시간은 표 2 에 나타나 있다. 실험 결과 학습 시간은 사전의 크기와 문장 당 학습하는 문장 최소 단위의 개수에 따라 좌우되는데, 두 가지 변수 모두 커질수록 학습에 필요한 연산량이 증가하기 때문에 학습 시간을 증가시킨다. 여기서 단어를 형태소 및 음절로 분할할 경우 사전 크기는 감소하지만 문장 당 문장 최소 단위의 개수는 증가한다. 실험 결과에서 음절 단위로 학습했을 때의 학습 시간이 가장 짧은데 이는 문장 최소 단위를 음절로 설정했을 때의 모델의 사전 크기가 나머지 두 모델에 비해 매우 적기 때문인 것으로 볼 수 있다. 한편 형태소 단위의 모델이 어절 단위의 모델보다 학습 시간이 약 10시간 정도 추가로 소요되고 있는데, 이는 감소한 사전 크기의 영향보다 증가한 문장 당 평균 형태소 개수가 문장 당 평균 어절 개수보다 많은 것이 더 큰 영향을 준 것으로 풀이된다.

표 2. 학습 시간
Table 2. Training Time

Data	Preprocess	Training time(hours)
Bible	word	7.5
	morpheme	6.1
	character	2.0
News	word	21.6
	morpheme	17.1
	character	5.4

표 3. 뉴스 데이터로 학습된 모델의 문장 생성 결과
Table 3. Sentence generation result using News data

Preprocess	Output Sentence
word	"정부는 전에도 이 같은 계획을 갖고 있는 것으로 알려졌다."
morpheme	"정부는 전에도 미국이 이라크에서 열린 우리당이 북한이 핵 프로그램을 위해 미국이 핵 프로그램을 비난했다."
character	"정부는 전에도 각각 8명의 상태가 목을 창출할 수 있는 서비스를 공개했다."

5.2.2 최대 확률 문장 생성 결과

아래 표 3에서는 입력 값으로 "정부는 전에도" 라는 부분 문장이 주어졌을 때의 출력 문장들을 나타내고 있다:

아래 표 4에서는 입력값으로 "이는 혈통으로나 육정으로나" 라는 부분 문장이 주어졌을 때의 출력 문장들을 나타내고 있다. 각 모델의 입력된 문장은 말뭉치의 주제를 고려하여 랜덤으로 입력되었다. 전반적으로 뉴스 데이터의 학습 결과가 성경 데이터보다 자연스러운 문장을 생성하였으며, 뉴스 데이터의 경우 성경 데이터의 약 4배 정도의 크기로 모델의 학습 결과에 큰 영향을 미친 것으로 생각된다. 음절 단위보다는 형태소 단위가, 형태소 단위 보다는 어절 단위를 채택한 모델이 뛰어난 성능을 보였으며, 이러한 결과는 문장 최소 단위 및 최대 어휘 사이즈의 영향을 받은 것으로 보인다. 한편 부족한 데이터는 모델 학습 과정에서 과적합(Over-fitting) 문제를 야기할 수 있는데, 상대적으로 학습 데이터가 적은 성경 데이터로 학습한 모델의 문장생성 결과에서 대다수가 비슷한 말을 반복하는 현상은 바로 이 과적합에 의한 것으로 보인다.

표 4. 성경 데이터로 학습된 모델의 문장 생성 결과
Table 4. Sentence generation result using Bible data

Preprocess	Output Sentence
word	"이는 혈통으로나 육정으로나 내가 나를 위하여 그 모든 말을 인하여 내 모든 말을 인하여 ..."
morpheme	"이는 혈통으로나 육정으로나 의 하나님께로 행하는것이 아니요 오직 하나님의 말씀을 알지 못함이니라. "
character	"이는 혈통으로나 육정으로나 사람이 그 아들 이스마엘과 아비야를 낳았고 그 아들 요나단이 그 아들 이스마엘과 아비야를 낳았고"

표 5. 샘플링을 통한 문장 생성 결과
Table 5. Sentence generation using sampling

Output Sentence
"정부는 전에도 중태 멸종위기 노동자들에게 회복될 것이라고 전했다 . <eos> "
"정부는 전에도 터키상공에서 이라크 정부의 협력을 꾸몄다고 결론 지었습니다 . <eos>"
"정부는 전에도 모든 가족을 논쟁에 분노하고 있다 . <eos>"
"정부는 전에도 미국에서 폭탄테러와 관련이 있다고 말했다 . <eos>"
"정부는 전에도 도시를 <eos>"
"정부는 전에도 발생한 폭탄 테러로 북핵 문건 을 <eos>"
"정부는 전에도 비상 공격 요구로 이명박 대통령을 지원할 수 있다고 주장했다 . <eos>"
"정부는 전에도 케냐 주민들이 동성연애자들을 대상으로 발표했다 . <eos>"

5.2.3 샘플링을 통한 문장 생성 결과

아래 표 5에서는 입력값으로 "정부는 전에도" 라는 부분 문장이 주어졌을 때 형태소를 문장 최소 단위로 사용하는 모델의 출력을 가장 큰 확률값이 아닌 각 확률의 크기로 표본 추출한 결과를 나타내고 있다.

샘플링을 통한 문장 생성 실험 결과 기존 최대 확률값을 가지는 값을 출력하는 경우에 비해 어법이나 문장 흐름 부분에서 부자연스러운 표현들의 출현 확률이 높았으나, 실험 결과에서도 확인할 수 있듯 생성된 문장은 주어진 부분 문장의 주제와 어느 정도 상통하는 어휘들로 이루어 진 것을 알 수 있다.

5.2.4 N-gram, vanilla RNN 과의 비교

아래 표 6에서는 LSTM 언어모델의 성능을 다른 언어모델과 비교하여 나열한 결과이다. 여기에서는 최대 50000개의 사전 크기를 가지는 어절 단위 뉴스 데이터를 사용하여 N-gram (3-gram), vanilla RNN 및 제시된 LSTM 기반 언어모델을 학습하여 평가한 결과를 나타내고 있다. 비교 실험에 사용된 N-Gram 언어모델은 NLTK 라이브러리[24]를 바탕으로 구현하였다. Vanilla RNN 기반 언어모델의 경우 제시된 LSTM 언어모델에서 LSTM 블록 대신 vanilla RNN 블록을 적용한 것이며, layer 수, 임베딩 크기 등의 모델 설정은 LSTM 기반 언어모델과 동일하다. N-gram 언어모델은 일반적으로 많이 사용되는 기초적인 언어모델이기 때문에 새로운 언어모델을 평가할 때 성능의 기준점이 될 수 있다. Vanilla RNN 언어모델은

표 6. N-gram, vanilla RNN, LSTM 기반 언어모델 비교
Table 6. Comparison between N-gram, vanilla RNN, LSTM based LM

Model	Perplexity	Sample sentences
N-gram	61	**미국 정부는 오늘** 아침 하락세로 개장했습니다.
		로스엔젤레스 남부의 한 작은 도시에서 총격전이 발생했다는 신고를 받고 출동한 경찰에 의해 수배 중이었다.
		우리는 그들의 도움이 필요하기 때문에 바로 착수할 수 있다고 말했다.
Vanilla RNN	64	**미국 정부는 오늘** 아침 처음으로 발표한 성명을 통해 이번 사건이 발생했다.
		로스엔젤레스 남부의 한 명의 미국 정부가 이끄는 기자회견에서 이번 사건에 대해 미국의 입장을 더 많은 조사를 하지 않을 것이라고 밝혔다
		우리는 그들의 경제 관련 문제를 해결할 수 있는 질문에 말했다.
LSTM	46	**미국 정부는 오늘** 초 오전 기자회견에서 이번 사건에 대해 더 많은 것을 발표했다.
		로스엔젤레스 남부의 한 종교단체 본부를 급습, 다른 세 명의 죽음에 이르게 한 20명의 여성을 체포했다.
		우리는 그들의 이런 모습을 본 적이 없다고 말했다.

LSTM 블록 구조가 언어 모델에서 실질적인 성능향상을 얼마나 가져올 수 있는지를 평가하기 위해 비교 실험하였다.

위 표의 perplexity는 테스트 셋 400 문장 각각에 대해 측정한 perplexity의 평균값이다. 문장 생성 결과는 굵은 글씨로 표시된 부분이 언어모델에 입력 값으로 주어진 내용이며, 이후 부분은 언어모델에 의해 생성된 단어들이다.

실험 결과는 크게 perplexity로 분석하는 정량적 성능 평가와 문장 생성 결과로 보는 정성적 성능 평가가 가능한데, 먼저 정량적 성능의 경우 제시된 LSTM 언어모델이 테스트 데이터에 대해 가장 낮은 평균 perplexity를 보였으며 뒤이어 N-gram 언어모델과 vanilla RNN 순으로 측정되었다. 이는 LSTM 언어모델이 주어진 테스트 문장에서 나타나는 단어의 확률 분포와 가장 유사하며, 즉 LSTM 언어모델의 학습 성능이 가장 뛰어난 것으로 해석이 가능하다. 한편 vanilla RNN 대비 LSTM 언어모델이 현저히 낮은 perplexity를 보였는데 이는 vanilla RNN 블록에서 LSTM 블록으로의 구조 개선이 언어 모델에 있어서는 성능 개선에 상당히 큰 영향을 끼친다고 볼 수 있다.

LSTM 기반 언어모델의 성능 우위는 문장 생성 결

과를 통한 정성적 평가에 있어서도 그대로 드러난다. 세 언어모델 모두 대체적으로 연속된 3-4단어 이내에서는 비교적 자연스러운 단어들을 생성해 내었는데, 문장 전체의 맥락을 살펴보면 LSTM 기반 언어모델이 vanilla RNN이나 N-gram 대비 보다 자연스러운 의미의 문장을 생성하였다. 세부적으로 목적어와 같은 문장 구성요소의 누락이나 단어 간 호응 관계 측면 등에서도 LSTM 언어모델이 생성한 문장이 좀 더 자연스러운 모습을 보이고 있다. 이는 LSTM이 vanilla RNN에 비해 단어 간의 long-term dependency를 보다 효과적으로 모델링할 수 있음을 보여주는 사례로 볼 수 있으며, 태생적으로 일정 개수의 단어만을 반영하는 N-gram 모델과의 비교에서도 우수성을 확인할 수 있다.

5.2.5 Embedding layer의 영향 분석

데이터를 서로 다른 기본단위로 분할하게 되면 모든 기본 단위를 포함하고 있는 사전의 크기가 달라지게 된다. 앞에서 언급한 내용과 같이 언어모델은 우선적으로 임베딩 계층을 통해 각각의 단어를 단어벡터로 재표현 하며, 재표현된 벡터들이 얼마나 해당 단어의 함축적인 의미를 잘 내포하고 있는지도 역시 언어모델의 최종적인 학습 결과에 영향을 미치게 된다. 아래 표 7에서는 각각 음절, 형태소, 음절을 기본 사전 단위로 사용한 성경데이터를 각각 1000-Embedding size, 500-Embedding size로 맵핑 시켰을 때 의 언어모델의 성능 분석 결과이다.

데이터를 각각 다른 문장최소단위로 분할할 경우 어절, 형태소, 음절 순서로 사전 크기가 작아진다. 결과에서도 알 수 있듯, 사전 크기가 비교적 큰 어절 단위 언어모델의 경우 임베딩을 적절하게 크게 선택하면 더 좋은 결과를 얻을 수 있으며, 반대로 음절 단위 언어모델의 경우 임베딩을 상대적으로 작게 선택했을 때의 perplexity 값이 작은 값을 기록했다. 위 실험 결과를 통해 사전 크기에 적합한 임베딩 크기 선정이 언어 모델의 성능에 중요한 역할을 하는 것을 확인할 수 있었다.

표 7. 임베딩 크기에 따른 perplexity 비교
Table 7. perplexity comparison between different embedding size

Perplexity	500 embedding	1000 embedding
어절	100	79
형태소	122	112
음절	111	141

Ⅵ. 결 론

본 논문에서는 LSTM 기반 언어모델을 통하여 문장의 일부분이 주어졌을 때 문장의 나머지 부분을 생성하여 문장을 완성시키는 시스템을 제안하였다. 또한 LSTM 구조가 기존 RNN과는 어떻게 다른지 밝혔고, 실험에서는 제시된 LSTM 기반 언어모델이 각 말뭉치의 주제, 말뭉치의 크기, 학습 문장 최소 단위가 모델 학습 성능 및 문장 생성 결과에 어떠한 영향을 미치는지 살펴보았다. 특히 학습 문장 최소 단위의 설정과 말뭉치의 크기가 언어모델의 성능에 중요한 것으로 판단되며, 실험 결과 약 9만 문장으로 구성된 뉴스 말뭉치에서 어절 단위로 학습 및 문장 생성을 했을 때 가장 좋은 문장 생성 결과를 얻을 수 있었다.

References

[1] S. H. Gil and G. H. Kim, “Vision-based vehicle detection and tracking using online learning,” *J. KICS*, vol. 39A, no. 1, pp. 1-11, 2014.

[2] J. H. Moon, et al., “Case study of big data-based agri-food recommendation system according to types of customers,” *J. KICS*, vol. 40, no. 5, pp. 903-913, 2015.

[3] O. Russakovskym, et al., “ImageNet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211-252, Dec. 2015.

[4] S. Kumar, et al., “Localization estimation using artificial intelligence technique in wireless sensor networks,” *J. KICS*, vol. 39C, no. 9, pp. 820-827, 2014.

[5] Y. Bengio, et al., “Leanring long-term dependencies with gradient descent is difficult,” *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157-166, 1994.

[6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[7] H. Sak, et al., “Fast and accurate recurrent neural network acoustic models for speech recognition,” in *Proc. INTERSPEECH*, Dresden, Germany, Sept. 2015.

[8] A. Graves, et al., “A novel connectionist

system for unconstrained handwriting recognition," *IEEE Trans. PAMI*, vol. 31 no. 5, pp. 855-868, May 2009.

[9] A. Grushin, et al., "Robust human action recognition via long short-term memory," in *IJCNN*, pp. 1-8, Dallas, United States, Aug. 2013.

[10] S. Shin, et al., "Image classification with recurrent neural networks using input replication," *J. KICS*, vol. 2015, no. 06, pp. 868-869, 2015.

[11] P. Koehn, et al., "Moses: Open source toolkit for statistical machine translation," in *ACL*, pp. 177-180, Prague, Czech, Jun. 2007.

[12] T. Mikolov, et al., "Extensions of recurrent neural network language model," in *ICASSP*, pp. 5528-5531, Prague, Czech, May 2011.

[13] T. Mikolov, "Statistical language models based on neural network," Ph.D. Dissertation, Brno University of Technology, 2012.

[14] I. Sutskever, et al., "Generating text with recurrent neural networks," in *ICML*, Bellevue, United States, Jun. 2011.

[15] T. Mikolov, et al., *Subword language modeling with neural networks*, preprint(http://www.fit.vutbr.cz/imikolov/rnnlm/char.pdf), 2012.

[16] Wkipedia, *Recurrent neural network*, Retrieved 3rd, Dec, 2015, https://en.wikipedia.org/wiki/Recurrent_neural_network.

[17] M. C. Mozer, *A focused backpropagation algorithm for temporal pattern recognition*, L. Erlbaum Associates Inc., pp. 137-169, 1995.

[18] S. Kirkpatrick, et al., "Optimization by simulated annealing," *Science*, pp. 671-680, 1983.

[19] Korean Bible Society, *Bible*, Retrieved 7th, Dec, 2015, http://www.bskorea.or.kr.

[20] jungyeul, *korean-parallel-corpora*(2014), Retrieved 22th, Oct, 2015, https://github.com/jungyeul/korean-parallel-corpora/tree/master/korean-english-v1.

[21] Twitter, *twitter-korean-text*, Retrieved 10th, Nov. 2015, https://github.com/twitter/twitter-korean-text.

[22] F. Bastien, et al., "Theano: new features and speed improvements," in *NIPS deep learning workshop*, Lake Tahoe, United States, Dec. 2012.

[23] J. Bergstra, et al., "Theano: A CPU and GPU math expression compiler," in *Proc. SciPy*, 2010.

[24] S. Bird, E. Klein, and, E. Loper *Natural Language Processing with Python*, O'Reilly Media Inc., Jun. 2009.

김 양 훈 (Yang-hoon Kim)

2014년 7월 : Tsinghua Univ. Department of Automation 졸업

2014년 9월~현재 : 서울대학교 전기정보공학부 석박사 통합과정

<관심분야> 머신러닝, 딥러닝, 자연어처리, 기계번역

황 용 근 (Yong-keun Hwang)

2014년 2월 : 한양대학교 전자정보시스템공학전공 졸업

2014년 3월~현재 : 서울대학교 전기정보공학부 석박사 통합과정

<관심분야> 머신러닝, 딥러닝, 자연어처리, 추천 시스템

강 태 관 (Tae-gwan Kang)

2016년 2월 : 서울대학교 전기정보공학부 졸업

2016년 3월~현재 : 서울대학교 전기정보공학부 석박사 통합과정

<관심분야> 머신러닝, 딥러닝, 음성처리

정 교 민 (Kyo-min Jung)

2003년 8월 : 서울대학교 수학과 졸업

2009년 6월 : Massachusetts Institute of Technology 수학과 박사

2009년~2013년 : 카이스트 전산학과 교수

2013년~현재 : 서울대학교 전기정보공학부 교수

<관심분야> 머신러닝, 딥러닝