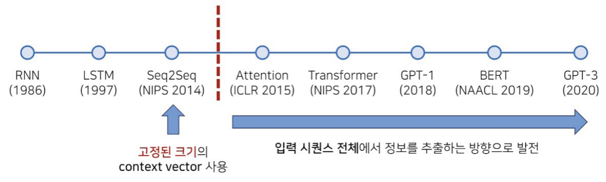


· 딥러닝 기반 기계번역의 돌파구 같은 역할 수행
· transformer (2017) 가 나오기 전까지 최고기술로 사용됨
LSTM을 활용한 효율적인 Seq2Seq 기계 번역 아키텍처를 제안함

Sequence to Sequence Learning with Neural Networks

= seq2seq (약어임)

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

다층 LSTM
여러 LSTM층 = * deep → 층이 여러개 ex) DNN : 은닉층이 여러개
통계적 기계 번역 (statistical machine translation)
단어 단위가 아닌 구문 (phrases, 단어들의 연속된 묶음)을 기본 번역 단위로 사용하는 SMT 방법

<DNN의 장점과 한계점>
장점 : 대규모 labeling된 train data set에서 good
단점 (한계점) : sequence to sequence (seq를 seq로 mapping)에 사용X

<제시하는 방법론>
1st) 다층 LSTM 사용
⇒ input sequence → 고정된 차원의 vector
mapping
2nd) 다층 LSTM 사용
⇒ 고정된 차원의 vector → target sequence
decoding

<결과>
(기존) PB-SMT : 33.3
↓
(new) LSTM : 34.8
↓
LSTM + rerank : 36.5

심층 신경망(DNNs)은 어려운 학습 작업에서 뛰어난 성능을 보여주는 강력한 모델입니다. DNNs는 대규모 라벨링된 학습 데이터셋이 있을 때 잘 작동하지만, 시퀀스를 시퀀스로 매핑하는 데는 사용할 수 없습니다. 이 논문에서는 시퀀스 구조에 대한 최소한의 가정만으로 시퀀스 학습을 위한 종단 간(end-to-end) 접근 방식을 제시합니다. 우리의 방법은 입력 시퀀스를 고정된 차원의 벡터로 매핑하는 다층 Long Short-Term Memory(LSTM)를 사용한 다음, 또 다른 깊은 LSTM을 사용하여 그 벡터에서 목표 시퀀스를 디코딩합니다. 우리의 주요 결과는 WMT'14 데이터셋의 영어-프랑스어 번역 작업에서 LSTM이 생성한 번역이 전체 테스트 세트에서 BLEU 점수 34.8을 달성했다는 것입니다. LSTM의 BLEU 점수는 어휘 외 단어에 대해 페널티를 받았습니다. 또한, LSTM은 긴 문장에서 어려움을 겪지 않았습니다. 비교를 위해, 구문 기반 SMT 시스템은 동일한 데이터셋에서 BLEU 점수 33.3을 달성했습니다. 우리는 LSTM을 사용하여 앞서 언급한 SMT 시스템이 생성한 1000개의 가설을 재순위화할 때, BLEU 점수가 36.5로 증가하여 이 작업의 이전 최고 결과에 가깝게 되었습니다. LSTM은 단어 순서에 민감하고 능동 및 수동 태세에 상대적으로 불변인 합리적인 구와 문장 표현도 학습했습니다. 마지막으로, 모든 소스 문장의 단어 순서를 반대로 바꾸면 LSTM의 성능이 크게 향상되었음을 발견했습니다. 이는 소스 문장과 목표 문장 사이에 많은 단기 의존성을 도입하여 최적화 문제를 더 쉽게 만들었기 때문입니다.

1 Introduction

DNN의 장점

Deep Neural Networks (DNNs) are extremely powerful machine learning models that achieve excellent performance on difficult problems such as speech recognition [13, 7] and visual object recognition [19, 6, 21, 20]. DNNs are powerful because they can perform arbitrary parallel computation for a modest number of steps. A surprising example of the power of DNNs is their ability to sort N N-bit numbers using only 2 hidden layers of quadratic size [27]. So, while neural networks are related to conventional statistical models, they learn an intricate computation. Furthermore, large DNNs can be trained with supervised backpropagation whenever the labeled training set has enough information to specify the network's parameters. Thus, if there exists a parameter setting of a large DNN that achieves good results (for example, because humans can solve the task very rapidly), supervised backpropagation will find these parameters and solve the problem.

Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a-priori. For example, speech recognition and machine translation are sequential problems. Likewise, question answering can also be seen as mapping a sequence of words representing the question to a

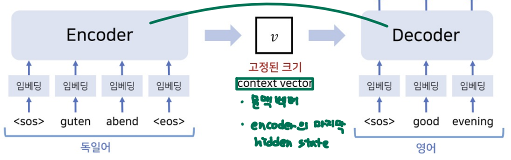
DNN의 한계점 +
mapping sequence to sequence의 필요성

Q. encoding, decoding 개념
A. encoding: sequence(ex: 문장) -> 고정된 크기의 벡터로 변환하는 과정
decoding: encoding으로 생성한 벡터 -> sequence(ex: 번역된 문장)으로 변환하는 과정

심층 신경망(DNNs)은 음성 인식[13, 7]과 시각 객체 인식[19, 6, 21, 20]과 같은 어려운 문제에서 뛰어난 성능을 달성하는 매우 강력한 기계 학습 모델입니다. DNNs는 임의의 병렬 계산을 적은 수의 단계로 수행할 수 있기 때문에 강력합니다. DNNs의 놀라운 예는 2개의 은닉층만을 사용하여 NN-비트 숫자를 정렬할 수 있는 능력입니다[27]. 따라서 신경망은 기존의 통계 모델과 관련이 있지만, 복잡한 계산을 학습합니다. 더욱이, 라벨링된 학습 세트가 네트워크의 매개변수를 지정할 충분한 정보를 가지고 있다면, 큰 DNNs는 지도 학습의 역전파를 통해 훈련될 수 있습니다. 따라서, 사람이 이 작업을 매우 빠르게 해결할 수 있기 때문에, 큰 DNN의 매개변수 설정이 좋은 결과를 달성할 수 있다면, 지도 역전파는 이러한 매개변수를 찾아 문제를 해결할 것입니다.

그들의 유연성과 강력함에도 불구하고, DNNs는 input과 target이 고정된 차원의 벡터로 합리적으로 인코딩될 수 있는 문제에만 적용될 수 있습니다. 이는 중요한 문제 중 많은 부분이 길이가 사전 알려지지 않은 시퀀스로 표현되는 것이 가장 좋기 때문에 중요한 한계입니다. 예를 들어, 음성 인식과 기계 번역은 순차적인 문제입니다. 마찬가지로, 질문 응답도 질문을 나타내는 단어 시퀀스를 답변을 나타내는 단어 시퀀스로 매핑하는 것으로 볼 수 있습니다. 따라서, 시퀀스를 시퀀스로 매핑하는 도메인 독립적인 방법이 유용하다는 것은 분명합니다.

< seq2seq model 구조 >



Q. domain - independent가 무슨 의미이지?

A. 도메인 독립적(Domain-independent)이라는 표현은 특정 도메인(특정 문제나 데이터 유형)에 한정되지 않고 다양한 도메인에 적용될 수 있음을 의미함. 즉, 어떤 특정 도메인에 맞추어져 있지 않기 때문에 다양한 분야에서 사용할 수 있는 방법이나 모델을 뜻함.

sequence of words representing the answer. It is therefore clear that a domain-independent method that learns to map sequences to sequences would be useful.

seq2seq model의 Idea와 구조

Sequences pose (제시하다) a challenge for DNNs because they (= DNNs) require that the dimensionality of the inputs and outputs is known and fixed. In this paper, we show that a straightforward (간단한) application (응용, 활용) of the Long Short-Term Memory (LSTM) architecture [16] can solve general sequence to sequence problems. The idea is to use one LSTM (LSTM encoder) to read the input sequence, one timestep at a time, to obtain large fixed-dimensional vector representation, and then to use another LSTM (LSTM decoder) to extract the output sequence from that vector (fig. 1). The second LSTM is essentially a recurrent neural network language model [28, 23, 30] except that it is conditioned on the input sequence. The LSTM's ability to successfully learn on data with long range temporal dependencies makes it a natural choice for this application due to the considerable time lag between the inputs and their corresponding outputs (fig. 1).

시퀀스는 DNNs에게 도전 과제를 제시하는데, 이는 입력과 출력의 차원이 알려져 있고 고정되어 있어야 하기 때문입니다. 이 논문에서 우리는 Long Short-Term Memory(LSTM) 아키텍처의 간단한 적용이 일반적인 시퀀스 투 시퀀스 문제를 해결할 수 있음을 보여줍니다. 아이디어는 하나의 LSTM을 사용하여 입력 시퀀스를 한 번에 하나의 타임스텝씩 읽어 큰 고정 차원의 벡터 표현을 얻고, 그런 다음 또 다른 LSTM을 사용하여 그 벡터에서 출력 시퀀스를 추출하는 것입니다(그림 1 참조). 두 번째 LSTM은 입력 시퀀스에 조건화된 순환 신경망 언어 모델과 본질적으로 동일합니다. LSTM이 장기 시간 의존성을 가진 데이터를 성공적으로 학습할 수 있는 능력 덕분에, 입력과 대응하는 출력 사이에 상당한 시간 차이가 있는 이 응용에 자연스럽게 적합합니다(그림 1 참조).

신경망을 사용하여 일반적인 seq2seq 학습문제를 해결하기 위한 다양한 접근 방식 (방법론)

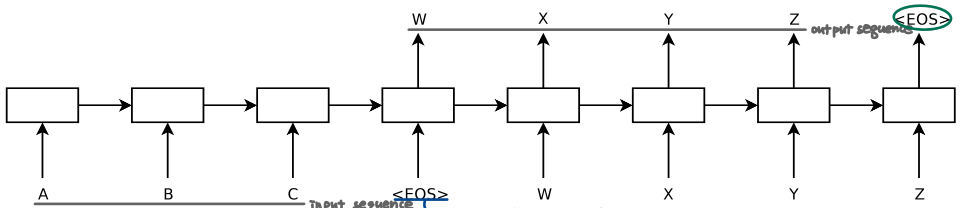
There have been a number of related attempts to address the general sequence to sequence learning problem with neural networks. Our approach is closely related to Kalchbrenner and Blunsom [18] who were the first to map the entire input sentence to vector, and is related to Cho et al. [5] although the latter was used only for rescoring hypotheses produced by a phrase-based system. Graves [10] introduced a novel (새로운) differentiable attention mechanism that allows neural networks to focus on different parts of their input, and an elegant variant of this idea was successfully applied to machine translation by Bahdanau et al. [2]. The Connectionist Sequence Classification is another popular technique for mapping sequences to sequences with neural networks, but it assumes a monotonic alignment between the inputs and the outputs [11].

신경망을 사용하여 일반적인 시퀀스 투 시퀀스 학습 문제를 해결하려는 여러 시도가 있었습니다. 우리의 접근 방식은 Kalchbrenner와 Blunsom[18]이 처음으로 전체 입력 문장을 벡터로 매핑한 것과 밀접하게 관련되어 있으며, Cho 등[5]의 연구와도 관련이 있지만 후자는 구문 기반 시스템이 생성한 가설을 재평가하는 데만 사용되었습니다. Graves[10]는 신경망이 입력의 다양한 부분에 집중할 수 있게 해주는 새로운 미분 가능한 attension 메커니즘을 도입했으며, 이 아이디어의 우아한 변형은 Bahdanau 등[2]에 의해 기계 번역에 성공적으로 적용되었습니다. Connectionist Sequence Classification은 또 다른 인기 있는 시퀀스 투 시퀀스를 매핑하는 기법이지만, 이는 입력과 출력 사이에 단조로운 정렬이 있어야 한다고 가정합니다[11].

* 초기 hidden state : 영벡터로 초기화함 (random으로 하지 x)

원래 LSTM: input sequence를 순차적으로 처리함
논문에서의 LSTM(LSTM encoder): input sequence를 역순으로 처리함 -> data에 많은 단기 의존성이 도입되어 최적화에 유리함

종료 시점: y_t = <EOS>



seq2seq model의 작동 방식

Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

그림 1: 우리의 모델은 입력 문장 "ABC"를 읽고 출력 문장 "WXYZ"를 생성합니다. 모델은 문장의 끝 토큰을 출력한 후 예측을 중지합니다. LSTM이 입력 문장을 역순으로 읽는다는 점에 주목하십시오. 이렇게 하면 데이터에 많은 단기 의존성이 도입되어 최적화 문제가 훨씬 쉬워집니다.

(fig1 기재된 것과는 다르게) C, B, A 순서로 처리됨

여기서의 seq2seq model → 아직 개선도 해야하고 최적화도 해야하지만 구문 기반 SMT보다 성능 good!
- encoder : 심층 LSTM 앙상블
- decoder : 좌-우 빔서치 decoder
  - Q. 얘도 LSTM decoder인가?
  - A. (X) 좌-우 빔서치 디코더는 빔서치 알고리즘을 사용하여 번역을 생성하는 디코더임. 빔서치(Beam Search)는 디코딩 과정에서 여러 후보 시퀀스를 동시에 탐색하여 최적의 번역을 찾는 방법임. 문장에서 "LSTM 디코더"라는 구체적인 언급은 없지만, 일반적인 seq2seq 모델에서는 LSTM 디코더를 사용하는 경우가 많음.

The main result of this work is the following. On the WMT'14 English to French translation task, we obtained a BLEU score of **34.81** by directly extracting translations from an ensemble of 5 deep LSTMs (with 384M parameters and 8,000 dimensional state each) (고정된 차원의 벡터 표현 생성) using a simple left-to-right beam-search decoder (번역 생성). This is by far the best result achieved by direct translation with large neural networks. For comparison, the BLEU score of an SMT baseline on this dataset is 33.30 [29]. The 34.81 BLEU score was achieved by an LSTM with a vocabulary of 80k words, so the score was penalized whenever the reference translation contained a word not covered by these 80k. This result shows that a relatively unoptimized small-vocabulary neural network architecture which has much room for improvement outperforms a phrase-based SMT system.

이 작업의 주요 결과는 다음과 같습니다. WMT'14 영어-프랑스어 번역 작업에서, 5개의 심층 LSTM 앙상블(총 3억 8,400만 개의 파라미터와 각각 8,000 차원의 상태)을 사용하여 단순한 좌-우 빔서치 디코더로 직접 번역을 추출하여 34.81의 BLEU 점수를 얻었습니다. 이는 큰 신경망을 사용한 직접 번역으로 달성된 최고의 결과입니다. 비교를 위해, 이 데이터셋에서 구문 기반 SMT 기준의 BLEU 점수는 33.30입니다 [29]. 34.81의 BLEU 점수는 80k 단어의 어휘를 가진 LSTM에 의해 달성되었으므로, 참조 번역에 이 80k에 포함되지 않은 단어가 있을 때마다 점수가 페널티를 받았습니다. 이 결과는 상대적으로 최적화되지 않은 소규모 어휘 신경망 아키텍처가 개선의 여지가 많이 있음에도 불구하고 구문 기반 SMT 시스템을 능가함을 보여줍니다.

Finally, we used the LSTM to rescore the publicly available 1000-best lists of the SMT baseline on the same task [29]. By doing so, we obtained a BLEU score of 36.5, which improves the baseline by 3.2 BLEU points and is close to the previous best published result on this task (which is 37.0 [9]).

마지막으로, 동일한 작업에서 SMT 기준의 공개된 1000-베스트 목록을 LSTM으로 재평가했습니다 [29]. 이를 통해 36.5의 BLEU 점수를 얻었으며, 이는 기준 점수를 3.2 BLEU 포인트 향상시키고 이 작업의 이전 최고 공개 결과(37.0 [9])에 가까운 점수입니다.

소스 문장의 단어를 뒤집는 trick (= LSTM encoder에서의 Input sequence 역순 처리 방법)
→ 긴 문장에서의 LSTM 성능 향상

Q. 원래 LSTM은 기존의 RNN의 장기의존성 문제를 해결한 모델인데, 그럼 이미 역순 처리를 하지 않아도 긴 문장에서의 성능은 좋았던 거 아닌가? (왜 역순 처리가 필요한가?)

A. 1. 추가적인 최적화
LSTM이 장기 의존성을 잘 처리할 수 있음에도 불구하고, 모든 문제에서 최적의 성능을 보장하는 것은 아님. 그러므로 실제로 긴 문장에서의 성능을 더욱 향상시키기 위해서는 추가적인 최적화가 필요할 수 있음. 특히, 번역과 같은 복잡한 시퀀스 투 시퀀스 작업에서는 입력 시퀀스와 출력 시퀀스 사이의 의존성이 매우 복잡할 수 있음.
2. 단기 의존성 증가
입력 시퀀스를 역순으로 처리하면, 디코더가 출력 시퀀스를 생성할 때 더 많은 단기 의존성을 활용할 수 있음. 단기 의존성이 증가하면, 모델이 더 쉽게 학습할 수 있으며, 최적화 문제를 간단하게 만들어 줌.

Surprisingly, the LSTM did not suffer on very long sentences, despite the recent experience of other researchers with related architectures [26]. We were able to do well on long sentences because we reversed the order of words in the source sentence but not the target sentences in the training and test set. By doing so, we introduced many short term dependencies that made the optimization problem much simpler (see sec. 2 and 3.3). As a result, SGD could learn LSTMs that had no trouble with long sentences. The simple trick of reversing the words in the source sentence is one of the key technical contributions of this work.

놀랍게도, LSTM은 관련 아키텍처로 최근 연구자들이 겪었던 경험에도 불구하고 매우 긴 문장에서 문제가 발생하지 않았습니다 [26]. 우리는 소스 문장의 단어 순서를 뒤집고 대상 문장은 뒤집지 않는 방법을 사용하여 긴 문장에서 잘 작동할 수 있었습니다. 이렇게 함으로써, 많은 단기 의존성을 도입하여 최적화 문제를 훨씬 간단하게 만들었습니다(2절 및 3.3절 참조). 그 결과, SGD는 긴 문장에서 문제없이 작동하는 LSTM을 학습할 수 있었습니다. 소스 문장의 단어를 뒤집는 간단한 트릭은 이 작업의 주요 기술적 기여 중 하나입니다.

LSTM의 특징과 번역 작업에서의 의미 표현 학습

A useful property of the LSTM is that it learns to map an input sentence of variable length into a fixed-dimensional vector representation. Given that translations tend to be paraphrases of the source sentences, the translation objective encourages the LSTM to find sentence representations that capture their meaning, as sentences with similar meanings are close to each other while different

LSTM의 유용한 특성 중 하나는 가변 길이의 입력 문장을 고정 차원의 벡터 표현으로 매핑하는 것을 학습한다는 점입니다. 번역이 원문 문장의 패러프레이즈(의역)인 경우가 많기 때문에, 번역 목표는 LSTM이 문장의 의미를 포착하는 문장 표현을 찾도록 장려합니다. 유사한 의미를 가진 문장들은 서로 가깝고 다른 의미를 가진 문장들은 멀리 떨어져 있게 됩니다. 질적 평가 결과, 우리 모델은 단어 순서를 인식하고 능동태와 수동태에 비교적 불변임을 보여줍니다.

<언어모델 (Language Model)>

· 문장(sequence)에 확률을 부여하는 model

· 언어모델을 통해 특정 상황에서의 적절한 문장 or 단어 예측 가능

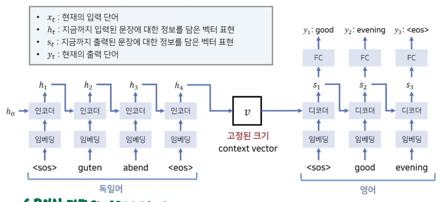
ex1) 기계 번역 · P(난 널 사랑해 | I love you) > P(난 널 싫어해 | I love you)

ex2) 다음 단어 예측 : P(먹었다 | 나는 밥을) > P(싸웠다 | 나는 밥을)

· 하나의 문장(W)은 여러 개의 단어(w)로 구성됨

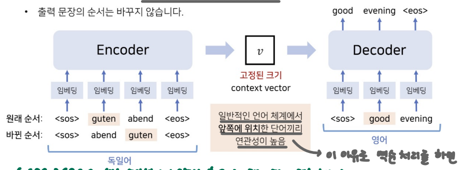
$P(W) = P(w_1, w_2, \ldots, w_n)$

ex) P(친구와 친하게 지낸다) = P(친구와, 친하게, 지낸다)



<RNN 기반의 seq2seq>

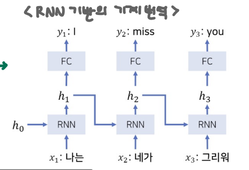
- 기본적인 RNN 대신에 LSTM을 활용했을 때 더 높은 정확도를 보입니다.
- 실제 학습 및 테스트 과정에서 **입력 문장의 순서를 거꾸로 했을 때 더 높은 정확도**를 보입니다.
  - 출력 문장의 순서는 바꾸지 않습니다.


Encoder
Decoder
v
고정된 크기
context vector
임베딩
원래 순서: <sos> guten abend <eos>
바뀐 순서: <sos> abend guten <eos>
독일어
<sos> good evening
good evening <eos>
영어
일반적인 언어 체계에서 앞쪽에 위치한 단어끼리 연관성이 높음

→ 이 이유로 역순 처리를 하면 짧은 단기 의존성 도입 가능

<seq2seq의 성능 개선 : LSTM 활용 + 입력 문장 역순 처리>

<RNN 기반의 기계번역>



Q. 능동태와 수동태에 비교적 불변이라는 게 무슨 의미?

A. Seq2Seq 모델이 능동태와 수동태와 같은 문법적 변형에도 불구하고 입력 문장의 의미를 일관되게 유지할 수 있음. 예를 들어, "The cat chased the mouse"와 "The mouse was chased by the cat"는 다른 구조를 가지고 있지만, 의미는 동일함. Seq2Seq 모델은 이러한 문장들을 유사한 벡터로 인코딩하여, 디코더가 이를 기반으로 적절한 번역을 생성할 수 있게 함. 이는 모델이 문장의 표면적 형태보다는 의미를 더 중요시하기 때문에 가능함.

sentences meanings will be far. A qualitative evaluation supports this claim, showing that our model is aware of word order and is fairly invariant to the active and passive voice.

2 The model

FFN:
- 가장 간단하고 기본적인 형태의 인공 신경망
- 정보가 입력층에서 시작하여 은닉층을 거쳐 출력층으로 일방향으로 흐르는 구조
- 복잡한 문제를 해결하기 위해 DNN, CNN, RNN 등의 변형이 개발됨

The Recurrent Neural Network (RNN) [31, 28] is a natural generalization of feedforward neural networks to sequences. Given a sequence of inputs $(x_1, \ldots, x_T)$, a standard RNN computes a sequence of outputs $(y_1, \ldots, y_T)$ by iterating the following equation:

순환 신경망(RNN)은 피드포워드 신경망을 시퀀스에 자연스럽게 일반화한 것입니다. 입력 시퀀스 (x1,...,xT)가 주어졌을 때, 표준 RNN은 다음 반복식을 통해 출력 시퀀스 (y1,...,yT)를 계산합니다.

* 일반적으로 RNN의 hidden state 구할 때 → tanh 활성화 함수를 사용함 (but 이 논문에서는 sigmoid 씀. 이유는 모름!)

· 입력값을 -1 ~ 1 사이로 변환
· 기울기 소실 문제 완화에 더 효과적임

보통 tanh를 쓰는데... ← sigmoid : 입력값을 0~1사이로 변환

input을 hidden layer로

이전 hidden layer에서 다음 hidden layer로 보내는 가중치

편향(b_h, b_y) 생략됨

$$\begin{aligned} h_t &= \mathrm{sigm}\left(W^{\mathrm{hx}} x_t + W^{\mathrm{hh}} h_{t-1}\right) \\ y_t &= W^{\mathrm{yh}} h_t \end{aligned}$$

입력 data 관련 가중치

hidden state 관련 가중치

RNN이 학습하는 parameter / 모든 시점의 state에서 이 parameter는 동일하게 적용됨

→ 입력 seq 길이 = 출력 seq 길이

Q. "입력과 출력 시퀀스 간의 정렬이 시간에 따라 미리 알려진 경우"라는 게 무슨 의미인가?

A. 입력 시퀀스의 각 요소가 출력 시퀀스의 각 요소와 시간 순서대로 일대일로 대응되는 경우를 의미함.
ex)
입력 시퀀스: 영어 문장
예: ["I", "am", "a", "student"]
출력 시퀀스: 프랑스어 번역
예: ["Je", "suis", "un", "étudiant"]
그러나 많은 자연어 처리 작업에서는 입력과 출력 시퀀스 간의 정렬이 명확하지 않은 경우가 많음. 예를 들어, 기계 번역에서 입력 문장과 출력 문장의 길이는 다를 수 있으며, 단어의 순서도 크게 다를 수 있음.
ex) 여기서 입력과 출력 시퀀스는 단어 단위로 일대일 대응되지 않으며, 문장의 길이와 구조가 크게 다를 수 있음.
입력 시퀀스: 영어 문장
예: ["The", "cat", "chased", "the", "mouse"]
출력 시퀀스: 프랑스어 번역
예: ["Le", "souris", "a", "été", "poursuivi", "par", "le", "chat"]

→ 이 경우에는 RNN 적용X

The RNN can easily map sequences to sequences whenever the alignment between the inputs the outputs is known ahead of time. However, it is not clear how to apply an RNN to problems whose input and the output sequences have different lengths with complicated and non-monotonic relationships.

RNN은 입력과 출력 시퀀스 간의 정렬이 시간에 따라 미리 알려진 경우 시퀀스를 시퀀스로 쉽게 매핑할 수 있습니다. 그러나 입력 시퀀스와 출력 시퀀스의 길이가 다르고 복잡하고 비단조적인 관계를 가지는 문제에 RNN을 적용하는 방법은 명확하지 않습니다.

The simplest strategy for general sequence learning is to map the input sequence to a fixed-sized vector using one RNN, and then to map the vector to the target sequence with another RNN (this approach has also been taken by Cho et al. [5]). While it could work in principle since the RNN is provided with all the relevant information, it would be difficult to train the RNNs due to the resulting long term dependencies (figure 1) [14, 4, 16, 15]. However, the Long Short-Term Memory (LSTM) [16] is known to learn problems with long range temporal dependencies, so an LSTM may succeed in this setting.

일반 시퀀스 학습을 위한 가장 단순한 전략은 입력 시퀀스를 고정된 크기의 벡터로 매핑한 다음, 또 다른 RNN을 사용하여 해당 벡터를 출력 시퀀스로 매핑하는 것입니다. 이 접근법은 Cho 등[5]에 의해 채택되었습니다. 이론적으로는 RNN이 모든 관련 정보를 제공받으면 작동할 수 있지만, 긴 시퀀스로 인한 긴 장기 의존성 문제 때문에 RNN을 훈련시키는 것은 어렵습니다. 그러나 장기 단기 기억(LSTM)은 긴 장기 의존성 문제를 잘 해결할 수 있으므로, 이 설정에서 LSTM이 성공할 가능성이 높습니다.

The goal of the LSTM is to estimate the conditional probability $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$ where $(x_1, \ldots, x_T)$ is an input sequence and $y_1, \ldots, y_{T'}$ is its corresponding output sequence whose length T' may differ from T. The LSTM computes this conditional probability by first obtaining the fixed-dimensional representation $\boxed{v}$ of the input sequence $(x_1, \ldots, x_T)$ given by the last hidden state of the LSTM, and then computing the probability of $y_1, \ldots, y_{T'}$ with a standard LSTM-LM formulation whose initial hidden state is set to the representation v of $x_1, \ldots, x_T$:

LSTM의 목표는 입력 시퀀스 (x1,...,xT)가 주어졌을 때, 출력 시퀀스 (y1,...,yT')의 조건부 확률 p(y1,...,yT'|x1,...,xT)를 추정하는 것입니다. 여기서 (y1,...,yT')는 대응하는 출력 시퀀스로, 길이 T'는 T와 다를 수 있습니다. LSTM은 입력 시퀀스 (x1,...,xT)의 고차원 표현 v를 얻은 후, 이 표현을 기반으로 표준 LSTM-LM 공식을 사용하여 y1,...,yT'의 확률을 계산합니다. 이때 초기 hidden state는 (x1,...,xT)의 표현 v로 설정됩니다.

* 연쇄법칙 (chain rule)

$P(w_1, w_2, \ldots, w_n) = P(w_1) P(w_2|w_1) P(w_3|w_1, w_2) \cdots P(w_n|w_1, \ldots, w_{n-1})$

$= \prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1})$

ex) P(친구와 친하게 지낸다)

= P(친구와) P(친하게 | 친구와) P(지낸다 | 친구와 친하게)

$x_1, \ldots, x_T$에 대한 정보를 담은 벡터 표현

encoder가 생성한 고정된 크기의 vector 표현

$$p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, \ldots, y_{t-1}) \tag{1}$$

입력 시퀀스 $(x_1, \ldots, x_T)$가 주어졌을 때 출력 시퀀스$(y_1, \ldots y_{T'})$의 조건부 확률

↳ 출력시퀀스가 생성될 확률

현재의 출력 단어

전체 시퀀스의 조건부 확률 = 개별 timestep의 확률을 곱한 것

In this equation, each $p(y_t|v, y_1, \ldots, y_{t-1})$ distribution is represented with a softmax over all the words in the vocabulary. We use the LSTM formulation from Graves [10]. Note that we require that each sentence ends with a special end-of-sentence symbol "<EOS>", which enables the model to define a distribution over sequences of all possible lengths. The overall scheme is outlined in figure 1, where the shown LSTM computes the representation of "A", "B", "C", "<EOS>" and then uses this representation to compute the probability of "W", "X", "Y", "Z", "<EOS>".

고정된 차원의 벡터 표현 계산

이 방정식에서 각 p(yt|v,y1,...,yt-1)분포는 어휘의 모든 단어에 대한 소프트맥스로 표현됩니다. 우리는 Graves[10]의 LSTM 공식을 사용합니다. 각 문장은 특별한 문장 끝 심볼 <EOS>로 끝나며, 이는 모델이 가능한 모든 길이의 시퀀스에 대해 분포를 정의할 수 있게 합니다. 전체 구조는 그림 1에 요약되어 있습니다. LSTM은 "A", "B", "C", "<EOS>"의 표현을 학습한 다음, 이 표현을 사용하여 "W", "X", "Y", "Z", "<EOS>"를 예측합니다.

Our actual models differ from the above description in three important ways. First, we used two different LSTMs: one for the input sequence and another for the output sequence, because doing so increases the number model parameters at negligible computational cost and makes it natural to train the LSTM on multiple language pairs simultaneously [18]. Second, we found that deep LSTMs significantly outperformed shallow LSTMs, so we chose an LSTM with four layers. Third, we found it extremely valuable to reverse the order of the words of the input sentence. So for example, instead of mapping the sentence a, b, c to the sentence α, β, γ, the LSTM is asked to map c, b, a to α, β, γ, where α, β, γ is the translation of a, b, c. This way, a is in close proximity to α, b is fairly close to β, and so on, a fact that makes it easy for SGD to "establish communication" between the input and the output. We found this simple data transformation to greatly improve the performance of the LSTM.

우리의 실제 모델은 위에서 설명한 기본 아이디어에서 세 가지 중요한 방식으로 다릅니다. 첫째, 입력 시퀀스를 위한 LSTM 하나와 출력 시퀀스를 위한 LSTM 하나를 사용하여, 학습할 매개변수의 수가 두 배로 증가하지만, 이는 중요하지 않으며 병렬화를 통해 여러 GPU에서 효율적으로 학습할 수 있습니다. 둘째, 우리는 다층 LSTM을 사용하여 깊은 LSTMs가 단일 언어 쌍에서 뛰어난 성능을 발휘함을 발견했습니다. 셋째, 소스 단어의 순서를 뒤집는 것이 매우 중요함을 발견했습니다. 이를 통해 SGD가 긴 문장에서의 성능을 향상시킬 수 있습니다.

3 Experiments

We applied our method to the WMT'14 English to French MT task in two ways. We used it to directly translate the input sentence without using a reference SMT system and we it to rescore the n-best lists of an SMT baseline. We report the accuracy of these translation methods, present sample translations, and visualize the resulting sentence representation.

3.1 Dataset details

We used the WMT'14 English to French dataset. We trained our models on a subset of 12M sentences consisting of 348M French words and 304M English words, which is a clean "selected" subset from [29]. We chose this translation task and this specific training set subset because of the public availability of a tokenized training and test set together with 1000-best lists from the baseline SMT [29].

As typical neural language models rely on a vector representation for each word, we used a fixed vocabulary for both languages. We used 160,000 of the most frequent words for the source language and 80,000 of the most frequent words for the target language. Every out-of-vocabulary word was replaced with a special "UNK" token.

3.2 Decoding and Rescoring

The core of our experiments involved training a large deep LSTM on many sentence pairs. We trained it by maximizing the log probability of a correct translation T given the source sentence S, so the training objective is

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

where $\mathcal{S}$ is the training set. Once training is complete, we produce translations by finding the most likely translation according to the LSTM:

$$\hat{T} = \arg\max_{T} p(T|S) \tag{2}$$

We search for the most likely translation using a simple left-to-right beam search decoder which maintains a small number B of partial hypotheses, where a partial hypothesis is a prefix of some translation. At each timestep we extend each partial hypothesis in the beam with every possible word in the vocabulary. This greatly increases the number of the hypotheses so we discard all but the B most likely hypotheses according to the model's log probability. As soon as the "<EOS>" symbol is appended to a hypothesis, it is removed from the beam and is added to the set of complete hypotheses. While this decoder is approximate, it is simple to implement. Interestingly, our system performs well even with a beam size of 1, and a beam of size 2 provides most of the benefits of beam search (Table 1).

We also used the LSTM to rescore the 1000-best lists produced by the baseline system [29]. To rescore an n-best list, we computed the log probability of every hypothesis with our LSTM and took an even average with their score and the LSTM's score.

3.3 Reversing the Source Sentences

While the LSTM is capable of solving problems with long term dependencies, we discovered that the LSTM learns much better when the source sentences are reversed (the target sentences are not reversed). By doing so, the LSTM's test perplexity dropped from 5.8 to 4.7, and the test BLEU scores of its decoded translations increased from 25.9 to 30.6.

While we do not have a complete explanation to this phenomenon, we believe that it is caused by the introduction of many short term dependencies to the dataset. Normally, when we concatenate a source sentence with a target sentence, each word in the source sentence is far from its corresponding word in the target sentence. As a result, the problem has a large "minimal time lag" [17]. By reversing the words in the source sentence, the average distance between corresponding words in the source and target language is unchanged. However, the first few words in the source language are now very close to the first few words in the target language, so the problem's minimal time lag is greatly reduced. Thus, backpropagation has an easier time "establishing communication" between the source sentence and the target sentence, which in turn results in substantially improved overall performance.

Initially, we believed that reversing the input sentences would only lead to more confident predictions in the early parts of the target sentence and to less confident predictions in the later parts. However, LSTMs trained on reversed source sentences did much better on long sentences than LSTMs trained on the raw source sentences (see sec. 3.7), which suggests that reversing the input sentences results in LSTMs with better memory utilization.

3.4 Training details

We found that the LSTM models are fairly easy to train. We used deep LSTMs with 4 layers, with 1000 cells at each layer and 1000 dimensional word embeddings, with an input vocabulary of 160,000 and an output vocabulary of 80,000. Thus the deep LSTM uses 8000 real numbers to represent a sentence. We found deep LSTMs to significantly outperform shallow LSTMs, where each additional layer reduced perplexity by nearly 10%, possibly due to their much larger hidden state. We used a naive softmax over 80,000 words at each output. The resulting LSTM has 384M parameters of which 64M are pure recurrent connections (32M for the "encoder" LSTM and 32M for the "decoder" LSTM). The complete training details are given below:

- We initialized all of the LSTM's parameters with the uniform distribution between -0.08 and 0.08
- We used stochastic gradient descent without momentum, with a fixed learning rate of 0.7. After 5 epochs, we begun halving the learning rate every half epoch. We trained our models for a total of 7.5 epochs.
- We used batches of 128 sequences for the gradient and divided it the size of the batch (namely, 128).
- Although LSTMs tend to not suffer from the vanishing gradient problem, they can have exploding gradients. Thus we enforced a hard constraint on the norm of the gradient [10, 25] by scaling it when its norm exceeded a threshold. For each training batch, we compute $s = \|g\|_2$, where g is the gradient divided by 128. If $s > 5$, we set $g = \frac{5g}{s}$.
- Different sentences have different lengths. Most sentences are short (e.g., length 20-30) but some sentences are long (e.g., length > 100), so a minibatch of 128 randomly chosen training sentences will have many short sentences and few long sentences, and as a result, much of the computation in the minibatch is wasted. To address this problem, we made sure that all sentences in a minibatch are roughly of the same length, yielding a 2x speedup.

3.5 Parallelization

A C++ implementation of deep LSTM with the configuration from the previous section on a single GPU processes a speed of approximately 1,700 words per second. This was too slow for our purposes, so we parallelized our model using an 8-GPU machine. Each layer of the LSTM was executed on a different GPU and communicated its activations to the next GPU / layer as soon as they were computed. Our models have 4 layers of LSTMs, each of which resides on a separate GPU. The remaining 4 GPUs were used to parallelize the softmax, so each GPU was responsible for multiplying by a 1000×20000 matrix. The resulting implementation achieved a speed of 6,300 (both English and French) words per second with a minibatch size of 128. Training took about a ten days with this implementation.

3.6 Experimental Results

We used the cased BLEU score [24] to evaluate the quality of our translations. We computed our BLEU scores using `multi-bleu.pl`[1] on the *tokenized* predictions and ground truth. This way of evaluating the BELU score is consistent with [5] and [2], and reproduces the 33.3 score of [29]. However, if we evaluate the best WMT'14 system [9] (whose predictions can be downloaded from `statmt.org\matrix`) in this manner, we get 37.0, which is greater than the 35.8 reported by `statmt.org\matrix`.

The results are presented in tables 1 and 2. Our best results are obtained with an ensemble of LSTMs that differ in their random initializations and in the random order of minibatches. While the decoded translations of the LSTM ensemble do not outperform the best WMT'14 system, it is the first time that a pure neural translation system outperforms a phrase-based SMT baseline on a large scale MT

[1]There several variants of the BLEU score, and each variant is defined with a perl script.

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
Best WMT'14 result [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	$\sim$45

Table 2: Methods that use neural networks together with an SMT system on the WMT'14 English to French test set (ntst14).

task by a sizeable margin, despite its inability to handle out-of-vocabulary words. The LSTM is within 0.5 BLEU points of the best WMT'14 result if it is used to rescore the 1000-best list of the baseline system.

3.7 Performance on long sentences

We were surprised to discover that the LSTM did well on long sentences, which is shown quantitatively in figure 3. Table 3 presents several examples of long sentences and their translations.

3.8 Model Analysis

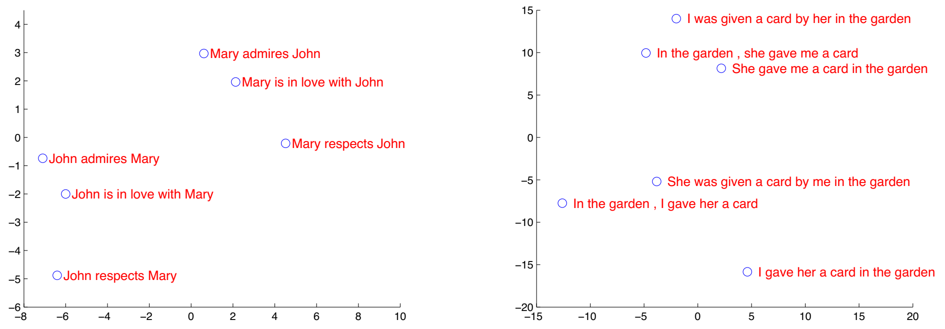


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

One of the attractive features of our model is its ability to turn a sequence of words into a vector of fixed dimensionality. Figure 2 visualizes some of the learned representations. The figure clearly shows that the representations are sensitive to the order of words, while being fairly insensitive to the

Type	Sentence
Our model	Ulrich UNK , membre du conseil d’ administration du constructeur automobile Audi , affirme qu’ il s’ agit d’ une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d’ administration afin qu’ ils ne soient pas utilisés comme appareils d’ écoute à distance .
Truth	Ulrich Hackenberg , membre du conseil d’ administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu’ ils ne puissent pas être utilisés comme appareils d’ écoute à distance , est une pratique courante depuis des années .
Our model	“ Les téléphones cellulaires , qui sont vraiment une question , non seulement parce qu’ ils pourraient potentiellement causer des interférences avec les appareils de navigation , mais nous savons , selon la FCC , qu’ ils pourraient interférer avec les tours de téléphone cellulaire lorsqu’ ils sont dans l’ air ” , dit UNK .
Truth	“ Les téléphones portables sont véritablement un problème , non seulement parce qu’ ils pourraient éventuellement créer des interférences avec les instruments de navigation , mais parce que nous savons , d’ après la FCC , qu’ ils pourraient perturber les antennes-relais de téléphonie mobile s’ ils sont utilisés à bord ” , a déclaré Rosenker .
Our model	Avec la crémation , il y a un “ sentiment de violence contre le corps d’ un être cher ” , qui sera “ réduit à une pile de cendres ” en très peu de temps au lieu d’ un processus de décomposition “ qui accompagnera les étapes du deuil ” .
Truth	Il y a , avec la crémation , “ une violence faite au corps aimé ” , qui va être “ réduit à un tas de cendres ” en très peu de temps , et non après un processus de décomposition , qui “ accompagnerait les phases du deuil ” .

Table 3: A few examples of long translations produced by the LSTM alongside the ground truth translations. The reader can verify that the translations are sensible using Google translate.

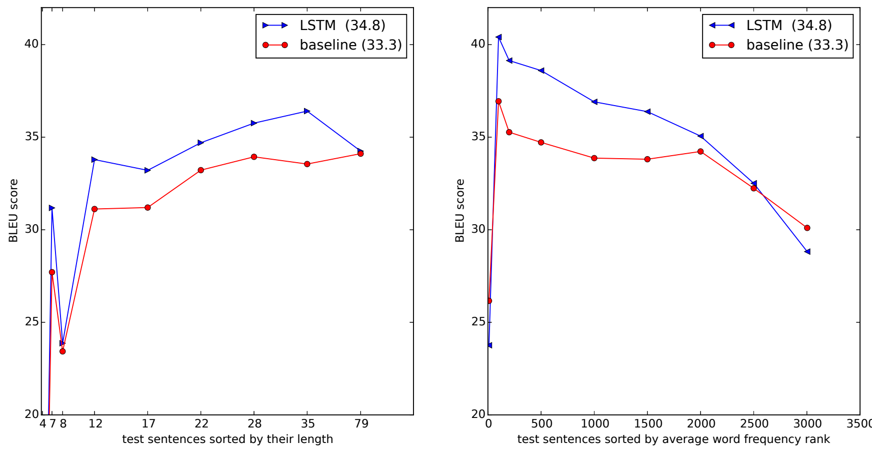


Figure 3: The left plot shows the performance of our system as a function of sentence length, where the x-axis corresponds to the test sentences sorted by their length and is marked by the actual sequence lengths. There is no degradation on sentences with less than 35 words, there is only a minor degradation on the longest sentences. The right plot shows the LSTM’s performance on sentences with progressively more rare words, where the x-axis corresponds to the test sentences sorted by their “average word frequency rank”.

replacement of an active voice with a passive voice. The two-dimensional projections are obtained using PCA.

4 Related work

There is a large body of work on applications of neural networks to machine translation. So far, the simplest and most effective way of applying an RNN-Language Model (RNNLM) [23] or a

Feedforward Neural Network Language Model (NNLM) [3] to an MT task is by rescoring the n-best lists of a strong MT baseline [22], which reliably improves translation quality.

More recently, researchers have begun to look into ways of including information about the source language into the NNLM. Examples of this work include Auli et al. [1], who combine an NNLM with a topic model of the input sentence, which improves rescoring performance. Devlin et al. [8] followed a similar approach, but they incorporated their NNLM into the decoder of an MT system and used the decoder's alignment information to provide the NNLM with the most useful words in the input sentence. Their approach was highly successful and it achieved large improvements over their baseline.

Our work is closely related to Kalchbrenner and Blunsom [18], who were the first to map the input sentence into a vector and then back to a sentence, although they map sentences to vectors using convolutional neural networks, which lose the ordering of the words. Similarly to this work, Cho et al. [5] used an LSTM-like RNN architecture to map sentences into vectors and back, although their primary focus was on integrating their neural network into an SMT system. Bahdanau et al. [2] also attempted direct translations with a neural network that used an attention mechanism to overcome the poor performance on long sentences experienced by Cho et al. [5] and achieved encouraging results. Likewise, Pouget-Abadie et al. [26] attempted to address the memory problem of Cho et al. [5] by translating pieces of the source sentence in way that produces smooth translations, which is similar to a phrase-based approach. We suspect that they could achieve similar improvements by simply training their networks on reversed source sentences.

End-to-end training is also the focus of Hermann et al. [12], whose model represents the inputs and outputs by feedforward networks, and map them to similar points in space. However, their approach cannot generate translations directly: to get a translation, they need to do a look up for closest vector in the pre-computed database of sentences, or to rescore a sentence.

5 Conclusion

★ 제한된 어휘를 가진 큰 심층 LSTM
vs
어휘가 무제한인 표준 SMT기반 시스템
⇒ 대규모 MT(기계번역) 작업에서
큰 심층 LSTM이 더 성능 good!

In this work, we showed that a large deep LSTM, that has a limited vocabulary and that makes almost no assumption about problem structure can outperform a standard SMT-based system whose vocabulary is unlimited on a large-scale MT task. The success of our simple LSTM-based approach on MT suggests that it should do well on many other sequence learning problems, provided they have enough training data.

이 연구에서 우리는 제한된 어휘를 가지고 문제 구조에 대해 거의 가정하지 않는 큰 심층 LSTM이 > 어휘가 무제한인 표준 SMT 기반 시스템을 대규모 기계 번역(MT) 작업에서 능가할 수 있음을 보여주었습니다. 우리의 간단한 LSTM 기반 접근 방식이 MT에서 성공을 거둔 것은 충분한 학습 데이터가 주어진다면 다른 많은 시퀀스 학습 문제에서도 잘 작동할 것임을 시사합니다.

We were surprised by the extent of the improvement obtained by reversing the words in the source sentences. We conclude that it is important to find a problem encoding that has the greatest number of short term dependencies, as they make the learning problem much simpler. In particular, while we were unable to train a standard RNN on the non-reversed translation problem (shown in fig. 1), we believe that a standard RNN should be easily trainable when the source sentences are reversed (although we did not verify it experimentally).

우리는 소스 문장의 단어를 역순으로 처리하여 얻은 개선의 정도에 놀랐습니다. 학습 문제를 훨씬 더 간단하게 만들기 위해 단기 의존성이 가장 많은 문제 인코딩을 찾는 것이 중요하다는 결론을 내렸습니다. 특히, 우리는 역순이 아닌 번역 문제에서는 표준 RNN을 훈련시킬 수 없었지만(그림 1에 나타나 있음), 소스 문장이 역순일 때는 표준 RNN도 쉽게 훈련시킬 수 있을 것이라고 믿습니다(비록 실험적으로 검증하지는 않았습니다).

We were also surprised by the ability of the LSTM to correctly translate very long sentences. We were initially convinced that the LSTM would fail on long sentences due to its limited memory, and other researchers reported poor performance on long sentences with a model similar to ours [5, 2, 26]. And yet, LSTMs trained on the reversed dataset had little difficulty translating long sentences.

또한 LSTM이 매우 긴 문장을 올바르게 번역할 수 있는 능력에도 놀랐습니다. 초기에는 LSTM이 한정된 메모리 때문에 긴 문장에서 실패할 것이라고 확신했으며, 다른 연구자들도 우리 모델과 유사한 모델로 긴 문장에서의 성능이 좋지 않다고 보고했습니다[5, 2, 26]. 그럼에도 불구하고, 역순 데이터셋으로 훈련된 LSTM은 긴 문장을 번역하는 데 거의 어려움이 없었습니다.

Most importantly, we demonstrated that a simple, straightforward and a relatively unoptimized approach can outperform an SMT system, so further work will likely lead to even greater translation accuracies. These results suggest that our approach will likely do well on other challenging sequence to sequence problems.

가장 중요한 점은 우리가 간단하고 직관적이며 비교적 최적화되지 않은 접근 방식이 SMT 시스템을 능가할 수 있음을 입증했다는 것입니다. 따라서 추가 연구는 번역 정확도를 더욱 향상시킬 가능성이 높습니다. 이러한 결과는 우리의 접근 방식이 다른 어려운 시퀀스 투 시퀀스 문제에서도 잘 작동할 것임을 시사합니다.

6 Acknowledgments

We thank Samy Bengio, Jeff Dean, Matthieu Devin, Geoffrey Hinton, Nal Kalchbrenner, Thang Luong, Wolfgang Macherey, Rajat Monga, Vincent Vanhoucke, Peng Xu, Wojciech Zaremba, and the Google Brain team for useful comments and discussions.

References

[1] M. Auli, M. Galley, C. Quirk, and G. Zweig. Joint language and translation modeling with recurrent neural networks. In *EMNLP*, 2013.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. In *Journal of Machine Learning Research*, pages 1137–1155, 2003.

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[5] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Arxiv preprint arXiv:1406.1078*, 2014.

[6] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR*, 2012.

[7] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing - Special Issue on Deep Learning for Speech and Language Processing*, 2012.

[8] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. Fast and robust neural network joint models for statistical machine translation. In *ACL*, 2014.

[9] Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. Edinburgh's phrase-based machine translation systems for wmt-14. In *WMT*, 2014.

[10] A. Graves. Generating sequences with recurrent neural networks. In *Arxiv preprint arXiv:1308.0850*, 2013.

[11] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006.

[12] K. M. Hermann and P. Blunsom. Multilingual distributed representations without word alignment. In *ICLR*, 2014.

[13] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 2012.

[14] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Master's thesis, Institut fur Informatik, Technische Universitat, Munchen*, 1991.

[15] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[17] S. Hochreiter and J. Schmidhuber. LSTM can solve hard long time lag problems. 1997.

[18] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *EMNLP*, 2013.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[20] Q.V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, and A.Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[22] T. Mikolov. *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.

[23] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[24] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[25] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

[26] J. Pouget-Abadie, D. Bahdanau, B. van Merrienboer, K. Cho, and Y. Bengio. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. *arXiv preprint arXiv:1409.1257*, 2014.

[27] A. Razborov. On small depth threshold circuits. In *Proc. 3rd Scandinavian Workshop on Algorithm Theory*, 1992.

[28] D. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[29] H. Schwenk. University le mans. `http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/`, 2014. [Online; accessed 03-September-2014].

[30] M. Sundermeyer, R. Schluter, and H. Ney. LSTM neural networks for language modeling. In *INTERSPEECH*, 2010.

[31] P. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of IEEE*, 1990.