**Sequence to Sequence Learning with Neural Networks**

Ilya Sutskever, Oriol Vinyals, Quoc V. Le

**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.심층 신경망(DNN)은 어려운 학습 작업에서 우수한 성능을 달성한 강력한 모델입니다. DNN은 레이블이 지정된 큰 훈련 세트를 사용할 수 있을 때마다 잘 작동하지만 시퀀스를 시퀀스에 매핑하는 데 사용할 수는 없습니다. 이 논문에서 우리는 시퀀스 구조에 대한 최소한의 가정을 하는 시퀀스 학습에 대한 일반적인 종단 간 접근 방식을 제시합니다. 우리의 방법은 다층 LSTM(Long Short-Term Memory)을 사용하여 입력 시퀀스를 고정 차원의 벡터에 매핑한 다음 다른 깊은 LSTM을 사용하여 벡터에서 대상 시퀀스를 디코딩합니다. 주요 결과는 WMT'14 데이터 세트의 영어에서 프랑스어로의 번역 작업에서 **LSTM으로** 생성된 번역이 전체 테스트 세트에서 BLEU(번역 품질을 측정하기 위한 정량적 지수로 기계가 번역한 문장과 정답 문장간의 정확도를 비교하여 측정\_높을수록 좋음) 점수 **34.8을 달성**했다는 것입니다. 여기서 LSTM의 BLEU 점수는 어휘 외의 단어에 대해 벌점을 받았습니다. 추가적으로 LSTM은 긴 문장에 어려움이 없었습니다. 비교를 위해 **구문 기반 SMT**(통계기반 기계번역\_Statistifcal Machine Translation) 시스템은 동일한 데이터 세트에서 **BLEU 점수 33.3**을 달성했습니다. 앞서 언급한 **SMT 시스템에서 생성된 1000개 가설의 순위를 재지정하기 위해 LSTM을 사용했을 때 BLEU 점수는 36.5로 증가**했으며 이는 이 작업에 대한 이전 최고의 결과에 가깝습니다. LSTM은 또한 어순에 민감하고 능동태와 수동태에 상대적으로 변하지 않는 합리적인 구와 문장 표현을 학습했습니다. 마지막으로, 우리는 모든 소스 문장(목표 문장[프랑스어]은 제외)에서 **단어[영어]의 순서를 반대로 하면 LSTM의 성능이 현저하게 향상된다는 것을 발견했습니다**. 그 이유는 역순으로 뒤집는 것이 문장 내 많은 단기의존성을 source와 target sentence사이에 도입해서 최적화 문제를 쉽게 만들어 줬기 때문입니다.

**1. Introduction**

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.

심층 신경망(DNN)은 음성 인식 및 시각적 개체 인식과 같은 어려운 문제에서 우수한 성능을 달성하는 매우 강력한 기계 학습 모델입니다. DNN은 적당한 수의 단계에 대해 임의의 병렬 계산을 수행할 수 있기 때문에 강력합니다. DNN의 능력에 대한 놀라운 예는 2차 크기의 hidden 레이어 2개만 사용하여 N개의 N비트 숫자를 정렬하는 기능입니다[27].*<quadratic size의 hidden layer의 의미를 잘 모르겠음. 크게 중요하지 않으므로 그냥 넘어감. 대충 2차원의 plane형태의 vector2층을 쌓아 만든 DNN(shallowNN과 유사)은 2-bit숫자(0~3)중 2개, 3-Bit숫자(0~7) 중 3개, 8-Bit(0-255) 중 8개를 정렬할 수 있는 능력이 있다는 말 같다>*

따라서 신경망은 기존의 통계 모델들과 관련이 있으면서도 더 복잡한 계산을 배웁니다. 또한 레이블링이 된 훈련 세트에 네트워크의 매개변수(parameter)를 지정하기에 충분한 정보가 있을 때마다 대형 DNN을 지도학습 방식의 역전파(supervised backpropagation)로 훈련할 수 있습니다. 따라서 좋은 결과를 얻는 대규모 DNN의 매개변수(parameter) 설정이 있는 경우(예: 인간이 작업을 매우 빠르게 해결할 수 있기 때문에), 지도학습 방식의 역전파는 이러한 매개변수(parameter)를 찾아 문제를 해결합니다.

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 1 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.

유연성과 능력에도 불구하고 DNN은 input과 output이 고정 차원의 벡터들이기 때문에 올바르게 인코딩될 수 있는 문제에만 적용될 수 있습니다. 많은 중요한 problems에서 길이가 사전에 알려지지 않은 시퀀스로 가장 잘 표현되기 때문에 이는 큰 한계점입니다. 예를 들어, 음성 인식과 기계 번역은 순차적인(sequential) 문제입니다. 마찬가지로 질의응답은 질문을 나타내는 단어 시퀀스를 답변을 나타내는 하나의 단어 시퀀스에 매핑하는 것으로 볼 수도 있습니다. 따라서 시퀀스를 시퀀스에 매핑하는 방법을 배우는 도메인-독립적인 방법이 유용할 것이 분명합니다.*<여기서 도메인-독립적인 방법이란 정해진 boundary에서 작동하는 method를 말하는 것 같다. 즉 학습에 사용된 sequence들로 구성된 dataset만을 가지고 작동하는 QnA정도를 의미한다고 볼 수 있다. 따라서 완성된 model은 dataset에 없는 sequence는 생성할 수 없다. 뭐 비슷한 형태나 패턴이라면 생성할 수 있음을 포함하는 말 일수도 있다.>*

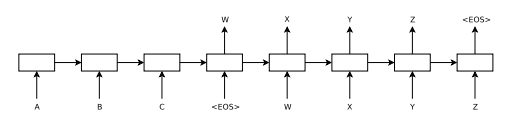
Sequences pose a challenge for DNNs because they require that the dimensionality of the inputs and outputs is known and fixed. In this paper, we show that a straightforward application of the Long Short-Term Memory (LSTM) architecture [16] can solve general sequence to sequence problems. The idea is to use one LSTM to read the input sequence, one timestep at a time, to obtain large fixeddimensional vector representation, and then to use another LSTM 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).

input과 output의 차원을 사전에 알려진 고정된 차원이어야하기 때문에 DNN으로 Sequence를 처리하는 것은 도전적인 문제이다. 이 논문에서 우리는 LSTM 아키텍처[16]의 복잡하지 않은 적용이 일반적인 sequence to sequence 문제를 해결할 수 있음을 보여줍니다.

아이디어는 하나의 LSTM을 사용하여 input시퀀스를 한 번에 한 타임스텝씩 읽고 큰 고정 차원 벡터 표현을 얻은 다음, 다른 LSTM을 사용하여 해당 벡터에서 output시퀀스를 추출하는 것입니다(그림 1). 두 번째 LSTM은 input시퀀스에 따라 조건이 지정된다는 점을 제외하고 본질적으로 순환 신경망 언어 모델입니다. 장기 의존성을 가진 데이터에 대해 성공적으로 학습할 수 있는 LSTM의 기능은 input과 해당 ouput사이의 상당한 시간 지연으로 인해 이 애플리케이션에 자연스러운 선택이 됩니다(그림 1).

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 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].

신경망의 일반적인 sequence to sequence 학습 문제를 해결하기 위한 많은 관련 시도가 있었습니다. 우리의 접근 방식은 전체 input문장을 벡터에 매핑한 최초의 Kalchbrenner and Blunsom과 밀접한 관련이 있으며, 비록 구절 기반 시스템에 의해 생선된 가설을 rescoring하는데만 사용되었지만 Cho et al.의 방법과도 관련이 있다. Graves는 neural networks가 input의 특정 부분에 집중할 수 있도록 하는 참신한 differentiable<미분가능인지 구분가능인지 모르겠음> attention mechnism을 도입했으며 이 아이디어의 우아한 변형이 Bahdanau et al.에 의해 기계 번역에 성공적으로 적용되었습니다. [2]. 연결주의 sequence classification는 neural networks을 사용하여 sequence 와 sequence를 매핑하는 또 다른 인기 있는 기술이지만 이는 input과 output사이의 단조로운 정렬을 가정합니다.



.

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: 우리 모델은 입력 문장 "ABC"를 읽고 "WXYZ"를 출력 문장으로 생성합니다. Model은 end-of-sentence를 출력한 후 예측을 중지합니다. LSTM은 입력 문장을 반대로 읽습니다. 그렇게 하면 최적화 문제를 훨씬 쉽게 만드는 데이터에 많은 단기 종속성이 도입되기 때문입니다.

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) using a simple left-to-right beamsearch decoder. 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 영어에서 프랑스어로 번역 작업에서 간단한 왼쪽에서 오른쪽으로 Beam Search(가장 확률이 높은 단어를 추출하는 Greedy Decoding의 단점을 어느정도 극복하기 위한 방법- 임의의 수 n를 지정한 다음 매 시퀀스마다 누적확률이 높은 상위 n개의 단어만 선택하는 것)를 수행하는 decoder를 사용하여 5개의 심층 LSTM(각각 3억8천4백만개 parameter 및 8,000차원 state)의 앙상블에서 번역문을 직역해서 BLEU 점수 34.81을 얻었습니다. 이것은 대규모 신경망을 사용한 직접 번역으로 달성한 단연 최고의 결과입니다. 비교를 위해 이 데이터 세트에 대한 SMT 기준선의 BLEU 점수는 33.30입니다. 34.81 BLEU 점수는 8만개 단어의 어휘를 사용한 LSTM에 의해 달성했습니다. 따라서 참조 번역에 이러한 8만개 단어 외의 단어가 포함될 때마다 점수가 감점되었습니다. 이 결과는 구문 기반 SMT 시스템보다 성능 향상의 여지가 많은 상대적으로 최적화되지 않은 작은 규모의 어휘에 대한 신경망 구조가 성능이 우수함을 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, 우리는 같은 2014 WMT English to French 번역 작업에 대해 오픈 돼있는 SMT 기반의 1000개 best 모델을 재채점 하기 위해 LSTM을 사용했습니다. 그렇게 함으로써 우리는 BLEU 점수 36.5점을 얻었는데 이것은 BLEU 점수를 3.2점까지 향상 시킨 것이고 이 번역 작업의 이전에 발표된 최상의 결과에 가까운 것이다 (최고점 37.0 [9]).

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 최근의 다른 연구자들의 LSTM과 관련된 구조에 대한 연구에서 경험한 한계에도 불구하고 아주 긴 문장에도 어려움을 겪지 않았습니다. training set, test set에서 source 문장에 있는 단어는 역순으로, target 문장에서는 순차로 배치했기 때문에 긴 문장에서도 잘 작동할 수 있었던 것입니다. 그렇게 함으로써 우리는 최적화 문제를 훨씬 더 단순하게 만드는 많은 단기 의존성을 도입했습니다(섹션 2 및 3.3 참조). 그 결과로, 긴 문장에서도 문제가 없는 LSTM들을 SGD방법으로 학습시킬 수 있었습니다. source sentence에서 단어를 역순으로 바꾸는 간단한 트릭은 이 논문의 주요한 기술적 기여 중의 하나입니다.

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 2 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.

LSTM의 유용한 특징은 다양한 길이를 가진 input sentence를 항상 고정된 차원수의 vector 표현으로 매핑하는 법을 학습한다는 것입니다. 번역은 source sentence들을 의역하는 경향이 있다는 점을 고려하면 (=번역은 문장을 그대로 직역하기 보다 원문이 말하고자 하는 바를 번역문에 오해 없이 담으려고 하는 경향이 있다는 점을 고려하면), LSTM 또한 번역을 하면서 다른 의미의 문장들은 서로 멀고 의미가 유사한 문장들은 서로 가까울수 있게끔 문장의 의미를 담아낼 수 있는 어떤 표현 (주로 vector를 뜻함)을 찾습니다. 정성적 평가는 우리 model이 단어의 어순을 이해하고 있고, 능동태와 수동태에 대해서도 상당히 불변한 결과를 보여줌으로써 이 주장을 뒷받침합니다.

**2. The model**

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

Recurrent Neural Network (RNN) [31, 28] 은 sequence들에 대한 feedforward neural networks의 자연스런 일반화이다. inputs sequence  가 주어질때, 표준 RNN은 다음의 방정식을 반복하면서 output sequence 를 계산합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

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 input과 output 사이 정렬이 미리 알려질 때마다 sequence들을 sequence들로 쉽게 매핑 할 수 있습니다. 그러나, RNN을 input과 output sequence들이 복잡하고 비단조적인 관계 하에 서로 다른 길이를 가진 경우의 문제들에 어떻게 적용하지는 불분명합니다.

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.

일반적인 sequence 학습을 위한 가장 단순한 전략은 1개 RNN을 사용하여 input sequence를 고정 사이즈의 vector로 매핑한 다음, 그 vector를 또다른 RNN을 이용해서 target sequence로 매핑하는 것이다(이런 접근법은 Cho et al. [5]에서도 사용됨)  
이런 전략은 모든 관련 정보가 RNN에 제공되기 때문에 이론적으로는 올바르게 작동하지만, 장기 의존성 때문에 RNN을 학습하는 것이 어려울 수 있다.  
그러나 LSTM은 장기적 의존성 문제 또한 학습할 수 있다고 알려져있다. 따라서 LSTM은 이러한 전략을 성공적으로 수행할 수 있을 것이다.

The goal of the LSTM is to estimate the conditional probability where is an input sequence and is its corresponding output sequence whose length T’ may differ from T. The LSTM computes this conditional probability by first obtaining the fixeddimensional representation v of the input sequence given by the last hidden state of the LSTM, and then computing the probability of with a standard LSTM-LM formulation whose initial hidden state is set to the representation of :

LSTM의 목표는  input sequence 와 그에 대응하는 길이가 T와 길이가 다를 수 있는 T′인 output sequence 에 대해조건부 확률 를 추정하는 것입니다. LSTM은 먼저 LSTM의 마지막 hidden state에 의해 주어진 input sequence 의 고정 차원 표현 v를 얻은 다음, 초기 hidden state가 input sequence 의 표현 v로 주어진 표준 LSTM-LM 공식으로 의 확률을 계산하여 이러한 조건부 확률을 계산합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

In this equation, each 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>”.

텍스트, 신문, 서류, 영수증이(가) 표시된 사진

자동 생성된 설명이 방정식에서 각각의 확률분포 는 어휘의 모든 단어에 대한 softmax로 표현됩니다. 우리는 Graves[10]의 LSTM 공식을 사용합니다각 문장의 끝은 모델이 다양한 길이의 sequence들에 대한 분포를 정의하는 것을 가능하게 해주는 하는 특별한 문장 끝 부호 “<EOS>”로 끝나야 합니다. 이 기호를 사용하면 모델이 가능한 모든 길이의 시퀀스에 대한 분포를 정의할 수 있습니다. 이 모든 체계는 (그림 1)에 요약되어 있습니다. 여기서 표시된 LSTM은 "A", "B", "C", "<EOS>"의 표현을 계산한 다음 이 표현을 사용하여 "W", "X", "Y", "Z", "<EOS>"의 확률을 계산.

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 세 가지 중요한 면에서 위의 설명과 다릅니다.

첫째, 우리는 **서로 다른 두 가지 LSTM을 사용**했습니다. 하나는 **input sequence용이고 다른 하나는 output sequence용**입니다. 그렇게 하는 것이 학습해야할 **model parameter의 수는 거의 증가시키지 않으면서도 LSTM이 다양한 언어쌍을 동시에 학습하는 것이 가능해지기 때문**입니다.[18] 예를 들자면, English 언어 타입의 Input Sequence를 Representation Vector로 바꾸는 Encoder(LSTM 모델)에 해당 Representation Vector를 특정 언어 타입(French, Korean)의 Output Sequence로 바꾸는 Decoder들을 쌍으로 묶을 수 있다는 말인 것 같다…

English->French, English->Korean 이런식으로

둘째, **깊은 LSTM이 얕은 LSTM보다 성능이 훨씬 뛰어나서 4개의 레이어가 있는 LSTM을 선택**했습니다.

셋째, **입력 문장의 단어 순서를 반대로 하는 것이 매우 가치 있음을 발견**했습니다. 따라서 예를 들어 문장 a, b, c를 문장 α, β, γ에 매핑하는 대신 LSTM에 c, b, a를 α, β, γ로 매핑하도록 요청합니다. 여기서 α, β, γ는 a, b, c의 번역입니다. 이렇게 a는 α에 아주 가깝고, b는 β에 아주 가까워지므로 (Encoder에서는 역순 input sequence를 넣기 때문에 a가 마지막으로 들어가게 되며 Decoder에서는 α가 가장 먼저 나오니까 a와 α를 가장 가까이 위치하도록 만든셈.), SGD가 input과 output 사이의 연관 관계를 연결하여 계산하는 것을 쉽게 만들어줍니다. (Gradient의 Backpropagation에 있어 α의 Gradient를 받아서 가능한 빨리 a에게 전달시킬 수 있다는 의미인 듯)  
우리는 이러한 단순 데이터 변환으로 LSTM의 성능을 극대화 시킬 수 있음을 알았습니다.

**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.

우리는 우리의 방법을 WMT2014 English to French 기계 번역 작업에 2가지 형태로 적용시켰습니다. 참조 SMT 시스템을 사용하지 않고 input sentence를 직역하는 형태와 SMT 시스템을 적용하여 나온 결과의 Top n을 재채점하는 형태가 그것입니다. 우리는 실험에서 그런 형태의 번역의 정확성과, 번역 결과 샘플, sentence represntation 결과를 시각화해 보이도록 합니다.

**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.우리는 WMT 2014 English to French 데이터셋을 사용했다. 우리의 모델은 3억 4천 8백만 프랑스 단어와 3억 4백만 영단어로 구성한 1200만개의 문장 집합들을 학습했으며 학습에 사용한 문장들은 [29] 로부터 깔끔하게 선택한 문장들이다.  
우리가 이 번역 작업과 특정한 학습 데이터셋을 선택한 이유는 데이터셋이 SMT 기준 best 1000 항목에 올라와 있는 토큰화 되어있는 train, test 공용 데이터셋이기 때문이다 [29].  
전형적인 뉴럴 언어처리 모델은 각 단어에 대한 representation vector에 의존하므로 우리는 양측 언어에 대해 고정적인 단어 수의 어휘록을 사용했다. 또한 source 언어에서 가장 자주 쓰이는 16만개의 단어, target 언어에서 가장 자주 쓰이는 8만개의 단어를 사용했다. 어휘록 외의 모든 단어는 특수 토큰인 “UNK”(Unknown)로 치환했다.

**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

우리 실험의 핵심 중 하나는 거대하고 깊은 LSTM으로 수많은 문장들을 학습하는 것이다.  
우리는 source sentence가 S일때, 올바른 번역 T가 일어날 확률의 로그 값을 최대화하는 방법 (Maximum Likelihood Estimation 최대가능도추정법)으로 모델을 학습시킨다.



where S is the training set. Once training is complete, we produce translations by finding the most likely translation according to the LSTM:여기서 S는 훈련 세트입니다. 교육이 완료되면 LSTM에 따라 가장 가능성이 높은 번역을 찾아 번역합니다.



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).

우리는 일부 번역의 부분적인 추측이 접두사일때 적은 수 B개의 부분 추측을 유지하는 단순한 left-to-right beam search decoder 이용해서 가장 적절한 번역을 찾는다. B개의 Beam을 가지고 탐색하는 Beam Search를 의미하는 듯하다.    
timestep 마다 우리는 beam 안의 부분 추측을 어휘록에 있는 가능한 모든 단어들로 확장한다. 이것은 추측의 수를 크게 증가시키며 우리는 모델의 log 확률값에 따라 가장 적합한 추측 B개를 제외하고 나머지를 버린다.  
추측 결과에 “<EOS>” 심볼이 나타나면 beam으로부터 제거되고 완성된 추측 집합에 추가된다.  
이 decoder는 거의 정확한 값을 계산해내는데 구현도 간단하다. 흥미롭게도, 우리의 시스템은 beam size가 1일때에도 잘 작동하고 beam size가 2일때 beam search의 대부분의 장점을 잘 발휘해낸다 (표 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.

또한 우리는 기준 시스템에 의해 생성된 1000개 베스트 항목[29]을 재채점 하기위해 LSTM을 사용했다. 우리는 n개 베스트 항목을 재채점하기 위해서 LSTM으로 모든 추측 결과의 log 확률을 계산하고 원래 점수와 LSTM 점수의 평균값을 사용했다.

**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.우리는 장기 의존성 문제를 해결할 수 있는 능력이 있는 LSTM이 source sentences를 거꾸로 뒤집었을 때 더 잘 학습 할 수 있다는 점을 발견했다 (target sentences는 뒤집지 않음).  
그렇게 함으로써 LSTM의 테스트 perplexity(혼란도)[4](https://seungwubaek.github.io/deep-learning/natural-language-processing/seq2seq_learning_with_neural_nets/#fn:perplexity)를 5.8에서 4.7로 감소시킬 수 있었고 번역문의 test BLEU 점수를 25.9에서 30.6으로 향상시킬 수 있었다.

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.

이러한 현상을 명확하게 설명할 순 없지만 우린 그 현상이 dataset에 많은 단기 의존성을 도입한 것에 의해 일어났다고 믿는다.  
보통, source sentence를 target sentence와 결합할 때, source sentence를 구성하는 각 단어는 그와 대응하는 target sentence의 단어로부터 멀리 떨어져있다. 그 결과로, 큰 “minimal time lag” 문제[17]를 낳는다. LSTM으로 입력된 source sentence의 초반의 단어들은 timestep이 증가하면서 점차 그 영향이 흐려진다는 것. 따라서 source sentence를 역순으로 입력하면 source sentence의 초반의 단어의 영향이 희미해지지 않으며 최종 Hidden State에 미치는 영향이 증가할 것이다. 그리고 그만큼 역전파 시, 초반 단어들에 대한 학습 효과를 향상시킨다.  
source sentence에 있는 단어들의 순서를 뒤집어도 source와 target 언어 간 대응하는 단어 사이의 평균적인 거리는 변하지 않는다. 그러나 source sentence의 처음의 몇개 단어들은 target 언어에 있는 처음의 몇개 단어들과 아주 가깝게 위치하게 되고 “minimal time lag” 문제를 상당히 감소시킨다. 초반의 단어에 있어서는 정보손실을 막을 수 있고 또한 초반 단어에 대한 target 값이 가까워 잘 예측할 수 있다고 했을 때, 반대로 원래 문장의 후반에 위치한 단어의 경우에는 맨 앞으로 위치하게 될 것이고 그렇다면 이 단어에 대한 정보는 손실이 커지고 또한 예측해야하는 값과의 거리도 늘어나 영향력이 많이 줄어들 것으로 보인다. 이 점에 대한 언급은 없어 궁금증이 생기게 되었다.  
그러므로 역전파 동안 source sentence와 target sentence 간 연결 고리를 더 쉽게 만들고 결과적으로 전체 성능을 크게 향상 시킨다.

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 4 trained on the raw source sentences (see sec. 3.7), which suggests that reversing the input sentences results in LSTMs with better memory utilization.초기에는 input sentences의 단어 순서를 역순으로 뒤집는 것이 target sentence의 앞부분에 대해서만 예측의 신뢰성을 좀 더 향상시키고 뒷부분에 대해서는 오히려 하락시킬 것이라 생각했다.  
그러나 역순의 source sentences를 학습한 LSTM들은 원형의 source sentences를 학습한 LSTM들 보다 긴 문장에 대해서 더 좋은 성능을 발휘했다. ([섹션 3.7](https://seungwubaek.github.io/deep-learning/natural-language-processing/seq2seq_learning_with_neural_nets/#37-performance-on-long-sentences) 참조)  
이것은 input sentences의 단어를 역순으로 뒤집는 것이 LSTM이 메모리를 더 효과적으로 이용하도록 만든다는 것을 의미한다.

**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:LSTM을 학습시키는 것은 꽤 쉬웠다. 우리는 각 layer가 1000개 cell로 이루어진 layer 4개를 쌓은 deep LSTM을 사용했고 16만개의 input 어휘와 8만개의 output 어휘를 1000차원의 word embedding으로 변환하여 사용했다.  
또한 깊은 LSTM이 얕은 LSTM의 성능을 압도하는 것을 발견했는데 LSTM layer가 추가될때 마다 perplexity가 거의 10% 씩 감소하였다. 이것은 deep LSTM의 hidden state가 더 크기 때문인 것으로 예상된다. 그리고 각 output에서 8만개 단어를 대상으로 하는 softmax를 적용했다. 최종 LSTM은 3억 8000만개의 parameter들을 가지며 그 중 6400만개는 순수 recurrent connection을 위한 인자들이다. (또한 그 중 3200만개는 encoder, 나머지 3200만개는 decoder에 있다). 전체 train 과정에 대한 상세는 아래와 같다

• We initialized all of the LSTM’s parameters with the uniform distribution between -0.08 and 0.08• 우리는 LSTM parameter들을 -0.08~0.08 사이 값을 갖는 균일분포를 따르는 임의의 값으로 초기화 했다.

• 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.• 학습 초반엔 Momentum 없는 Stochastic Gradient Descent (모멘텀 없는 랜덤 경사하강법. 성능이 좋지 않은 Naive SGD를 사용했다는 의미 같다.)에 learning rate 값을 0.7로 고정해놓고 학습을 진행했다.  
5 epoch을 학습한 후 부터는 0.5 epoch 마다 learning rate를 절반으로 줄여가면서 모델이 처음의 5 epoch을 포함하여 총 7.5 epoch을 학습할때까지 학습을 진행했다

• We used batches of 128 sequences for the gradient and divided it the size of the batch (namely, 128).

• gradient를 얻기 위해 128 sequence의 배치들을 사용했고 gradient를 batch size로 나누었다 (즉, 128로 나눔).

• 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 , where g is the gradient divided by 128. If we set .

• LSTM에서는 vanishing gradient 문제가 잘 발생하지 않지만 exploding gradient 문제가 발생한다. 그래서 우리는 gradient의 norm(vector의 크기로 해석)이 threshold를 초과할때 그것을 scaling 하는 식으로 norm에 강한 제약을 걸었다[10, 25].   
각 training batch에서 g를 gradient를 128로 나눈 값이라고 할 때, 를 계산한다. 그리고 s>5 이면,  로 설정하여 gradient를 scaling 했다.

• 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.• 서로 다른 문장은 서로 다른 길이를 가진다. 대부분의 문장은 짧지만 (예를들면, 20-30 정도의 길이) 어떤 문장은 길기 때문에 (예들들어, 100를 초과하는 길이), 랜덤하게 선택된 128의 traing sentence들의 minibatch는 많은 짧은 문장과 적은 수의 긴 문장을 갖게 된다. 그리고 그 결과로, minibatch의 연산 대부분이 낭비된다.  
이 문제를 해결하기 위해 우리는 minibatch 내의 모든 sentences들이 거의 비슷한 길이를 갖도록 했고 연산 속도가 2배로 향상됐다.

**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.이전 섹션에서 설명한 설정들에 따라 C+로 구현한 deep LSTM은 single GPU에서 초당 약 1700개의 단어를 처리한다. 이것은 우리의 목적을 위해서는 느린 속도였고 모델 연산을 위해 8개 GPU machine으로 병렬처리를 수행했다.  
LSTM의 각 layer는 서로 다른 GPU에서 실행되고 계산이 완료된 즉시 activation 값을 다음의 GPU(또는 layer)로 전달하여 통신한다. 우리의 모델은 4개의 LSTM layer로 구성돼 있는데 각 layer는 서로 분리된 GPU에 할당된다. 남은 4개의 GPU들은 softmax 연산을 병렬화 하는데 사용되었다. softmax를 위해 각 GPU는 1000 x 20000 사이즈의 matrix의 곱셈 연산을 맡는다.  
최종 구현체는 데이터 셋을 학습 반복(training iteration) 단위 사이즈인 mini batch size 128로 분할 학습(SGD without momentum)하여 초당 6300개의 영단어와 프랑스 단어를 처리하는 속도를 냈다. 학습이 완료되기까지는 10일 정도가 소요됐다.

**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.pl1 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.

우리는 BLEU 점수[24]를 사용하여 번역 품질을 평가했습니다. 우리는 토큰화된 예측과 정답에 대해 multi-bleu.pl1을 사용하여 BLEU 점수를 계산했습니다. 이러한 BELU 점수 평가 방식은 [5], [2]와 일치하며, [29]의 33.3점을 재현한다.   
그러나 이 방법으로 [9]의 최신 시스템(예측값은 <statmt.org/matrix>에서 다운로드 가능)을 평가했을 때 우린 37.0 점을 얻었으며, 이것은 <statmt.org/matrix>에 보고된 35.8 점 보다 높다.

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 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.

table 1과 2에 그 결과를 작성하였다. 각 LSTM layer 마다 서로 다른 초기화 과정을 거치고 minibatch를 랜덤하게 섞어 학습한 LSTM 앙상블로부터 best result를 얻었다.  
LSTM 앙상블의 번역 결과가 세계 최고 신기록을 달성하진 못했지만 등록된 어휘 단어 외의 단어를 처리할 수 없음에도 불구하고 순수 neural 번역 시스템으로써 대규모 기계 번역 작업에서 구(phrase) 기반의 SMT 시스템의 성능을 상당한 차이로 최초로 넘었다.  
LSTM은 기준 시스템의 최고 1000개 항목을 재채점 함으로써 이전의 최고 점수에 0.5 BLEU 점 뒤쳐져있다.

테이블이(가) 표시된 사진

자동 생성된 설명

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.표 1: WMT'14 영어에서 프랑스어로 테스트 세트(ntst14)에 대한 LSTM의 성능. 빔 크기가 2인 5개의 LSTM 앙상블은 빔 크기가 12인 1개(단일) LSTM보다 저렴합니다.

테이블이(가) 표시된 사진

자동 생성된 설명

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

표 2: WMT'14 영어에서 프랑스어로 테스트 세트(ntst14)에서 SMT 시스템과 함께 신경망을 사용하는 방법.

**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.우리는 LSTM이 그림 3에 정량적으로 표시된 긴 문장에서 잘 작동한다는 사실에 놀랐습니다. 표 3은 긴 문장과 그 번역의 몇 가지 예를 보여줍니다.

**3.8 Model Analysis**

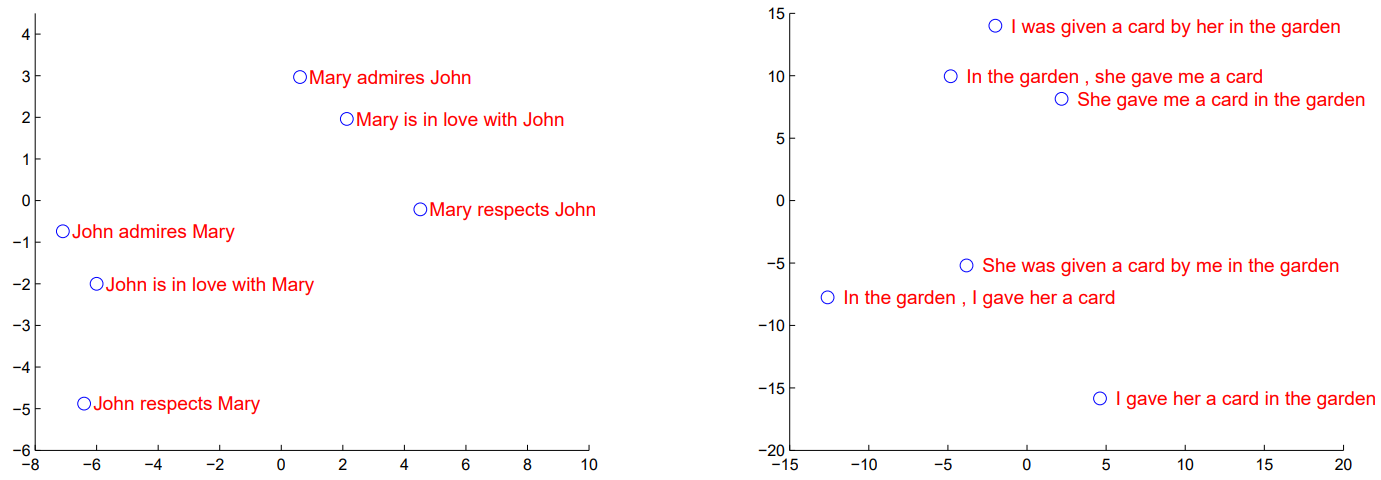


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words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

그림 2: 그림은 그림의 문구를 처리한 후 얻은 LSTM hidden state의 2차원 PCA 투영을 보여줍니다. 구는 의미별로 클러스터링되어 있으며, 이러한 예에서는 이것은 단어 순서의 기능이며, 이는 bag-of-word 모델로는 포착하기 어렵습니다. 두 클러스터의 내부 구조가 비슷합니다.

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

우리 model의 매력적인 기능 중 하나는 단어의 sequence를 고정 차원수의 vector로 변환하는 능력이다. [Figure 2](https://seungwubaek.github.io/deep-learning/natural-language-processing/seq2seq_learning_with_neural_nets/#figure-2)는 학습된 몇개의 representation들을 시각화하여 보여준다.  
그림은 representation들이 수동태, 능동태의 차이에는 매우 둔감한 반면 단어의 순서에는 민감하다는 것을 명확하게 보여준다. 이 그림의 2차원 영사는 PCA를 통해 생성됐다.

텍스트이(가) 표시된 사진

자동 생성된 설명

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.

표 3: 실측 번역(정답)과 함께 LSTM에 의해 생성된 장문의 번역의 몇 가지 예시이다. Google translate를 사용한 번역이 말이 된다는 것을 독자가 평가할 수 있다.

텍스트, 하늘, 지도이(가) 표시된 사진

자동 생성된 설명

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”.

그림 3: 왼쪽의 plot은 문장 길이의 기능으로써 우리 시스템의 성능을 보여준다. 이때 x축은 길이 순으로 정렬된 테스트 문장들에 해당하며 실제 sequence 길이로 표시된다. 단어수가 35개 미만의 문장은 성능 저하가 없고 그보다 많은 단어수의 문장에서 약간의 저하가 있다. 오른쪽 plot은 좀더 희귀한 단어를 가진 문장들에서 LSTM의 성능을 보여준다. 이때 x축은 평균 단어 빈도 순위에 따라 정렬된 테스트 문장들에 해당한다.

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

Neural Network를 기계 번역에 적용하는 많은 작업들이 있다. 지금까지, RNN-Language Model(RNNLM)[23] 또는 Feedforward Neural Network Language Model(NNLM)[3]을 기계 번역 작업에 적용하는 가장 단순하고 효과적인 방법은 강력한 기계번역 기준 [22]의 n개 베스트 항목을 재채점하는 방법이이었고 그런 방법은 확실하게 번역의 질을 향상시켰다.

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.

좀 더 최근엔, 학자들은 source language의 정보를 NNLM(신경망 모델)로 포함시키는 방법을 찾기 시작했다. 그 예로 Auli et al. [1] 이 있는데 여기서는 NNLM을 input sentence의 topic model과 결합시켰고(sentence의 주제에 대한 representation을 뽑는 모델(topic model)을 NNLM과 결합시켰다는 것), 재채점 성능을 향상시킬 수 있었다.   
Devlin et al. [8]은 비슷하게 접근하면서 NNLM을 기계 번역 시스템의 decoder에 통합했고 input sentence에서 가장 유용한 단어를 NNLM에 제공하기 위해 decoder의 정렬 정보를 사용했다. 이 방법은 매우 성공적이었고 그들의 비교 기준으로 부터 아주 큰 향상을 이루었다.

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.우리의 방법은 Kalchbrenner and Blunsom [18]과 밀접한 관련이있다. 그들의 방법은 처음으로 input sentence를 vector로 매핑하고 다시 sentence에 매핑했다. 하지만 그들 방법에서는 convolution neural networks를 사용해서 문장을 vector로 매핑 했고 단어 간의 순서 정보를 상실했다.  
이것과 유사하게, Cho et al [5]은 비록 주요 관심은 그들의 neural network를 SMT 시스템에 통합시키는 것이었지만, sentences를 vectors로 매핑하고 다시 거꾸로 매핑하는데에 LSTM-like RNN 구조를 사용했었다.  
Bahdanau et al. [2]는 또한 Cho et al. [5]에서 경험한 긴 문장들에대한 저성능을 극복하기 위해 attention mechanism을 사용한 neural network로 직접 번역을 시도했고 고무할 만한 결과를 달성했다.  
비슷하게, Pouget-Abadie et al. [26]은, 구(phrase) 기반 접근법과 유사하게, 부드러운 번역을 생성하는 방식으로 source sentence의 조각들을 번역해서 Cho et al. [5]의 메모리 문제의 해결하려고 시도했다.  
그러나 우리는 그들의 이러한 방법 대신 단순히 network에 역순의 source sentences를 학습시키는 것으로 비슷한 수준의 향상을 달성할 수 있을 것이라 생각한다.

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.종단 간 훈련은 또한 Hermann et al. [12]의 관심사였는데 그들의 모델은 feedforward nerworks에 의해 input과 output을 표현하고 그것들을 공간 안의 비슷한 점에 매핑한다. 그러나, 그들의 접근법은 번역을 직접 생성 할 수 없다: 번역 결과를 얻거나 벡터를 문장으로 복원하기 위해서는, 미리 계산된 문장 벡터 database에서 가장 가까운 벡터를 검색하는 작업이 필요하다.

**5. Conclusion**

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 non MT suggests that it should do well on many other sequence learning problems, provided they have enough training data.

이 작업에서 우리는 어휘가 제한적이고 문제 구조에 대한 가정을 거의 하지 않는 대규모 deep LSTM이 대규모 기계번역 작업에서 어휘가 무제한인 표준 SMT 기반 시스템보다 성능이 우수함을 보여주었습니다. 기계 번역에 대한 간단한 LSTM 기반 접근 방식의 성공은 만약 훈련 데이터가 충분하다면 다른 많은 시퀀스 학습 문제에서도 잘 수행될 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 source sentences의 단어를 역순으로 배치하는 것으로 얻은 향상의 정도에 놀랐다. 우리는 단기 종속성이 학습 문제를 더 쉽게 만들어주기 때문에 가장 많은 단기 종속성을 갖는 encoding 문제를 찾는 것이 중요하다고 결론을 내린다.  
특히, 역순이 아닌 번역 문제에서 standard RNN을 학습할 수 없었지만([fig. 1](https://seungwubaek.github.io/deep-learning/natural-language-processing/seq2seq_learning_with_neural_nets/#figure-1)에서 보임), standard RNN도 source sentence가 역순으로 된 경우 쉽게 학습 할 수 있을 것이라 믿는다. (실험적으로 증명해보진 못했다)

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.

우리는 또한 매우 긴 문장을 정확하게 번역하는 LSTM의 능력에 놀랐습니다. 우리는 처음에 LSTM이 제한된 메모리로 인해 긴 문장에서 실패할 것이라고 확신했고 다른 연구자들은 우리와 유사한 모델을 사용하여 긴 문장에서 성능이 좋지 않다고 보고했습니다. 그러나 역 데이터셋으로 훈련된 LSTM은 긴 문장을 번역하는 데 어려움이 거의 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 것은 단순하고 간단하며 상대적으로 최적화되지 않은 접근 방식이 SMT 시스템보다 성능이 우수하므로 추가 작업으로 번역 정확도가 훨씬 더 높아질 수 있다는 점입니다. 이러한 결과는 우리의 접근 방식이 다른 도전적인 sequence to sequence 문제에서도 좋은 결과를 낼 것임을 시사합니다.

**6 Acknowledgements**

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.

유용한 논평과 토론을 해주신 Samy Bengio, Jeff Dean, Matthieu Devin, Geoffrey Hinton, Nal Kalchbrenner, Thang Luong, Wolfgang Macherey, Rajat Monga, Vincent Vanhoucke, Peng Xu, Wojciech Zaremba 및 Google Brain 팀에 감사드립니다.