공학 논문 작성법

유하영

23. 12. 26

공학 논문

보통 학술대회에서 발표한 내용으로
자신의 연구내용을 업그레이드 하여
학술지에 제출

	학술대회 논문 (conference)		학술지 논문 (journal)
목적	**학자들이 연구하는 것을 발표하는 논문** 연구동향파악 다른 연구자들과 공유 및 교류	목적	**엄격한 심사를 통과하여 게재되는 논문** (완성된 논문) 연구 결과를 학계에 정식으로 기록하는 정기적으로 출판되는 간행물
형식	구두세션 , 포스터 세션 등을 통한 **저자의 논문 발표**	형식	발표없이 **저널에 게재만**
절차	초록제출 -> 수락 -> 학회발표	절차	논문 제출 -> 심사 (6 개월 ~1 년) -> 게재

공학 논문 의 구성

제목 (Title)

초록 (Abstract)

↓

서론 (Introduction)

↓

연구 배경 및 방법 (Materials & Methods)

연구 결과 (Results)

↓

결론 (Conclusions)

↓

참고문헌 (references)

Abstract (초록)

목적 : 본 논문에 대한 핵심적인 개요를 전달하고 ,
독자로 하여금 논문 전체를 읽을 것인지 여부를
결정하게 함

용도 : 독자는 Abstract 를 읽음으로써 본 논문의 주제와 연구 내용을
이해한 후 논문을 더 자세히 읽을 것인지 결정

분량 : 보통 150~250 단어 이하의 짧은 문단

구조 : 연구목표 (Objective)
연구방법 (Methods)
결과 (Result)
결론 (Conclusion)

순서로 간략히 기술

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Introduction (서론)

목적 : 독자에게 본 논문의 연구 분야에 대한 이해를 돕기 위해

연구의 배경정보를 보다 **구체적으로** 설명하는 섹션

용도 : 독자에게 논문의 주제 영역에 대한 이해를 도움

분량 : 일반적으로 1-2 page

구조 : 연구의 배경

문제 제기

(선행연구 살짝 제시)

연구의 중요성 및 필요성

연구의 목표 제시 . 연구의 진행방향 간략하게 제시

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

3 Model Architecture

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

Materials & Methods (연구 배경 및 방법)

목적 : 연구를 수행한 방법과 절차를 상세하게 설명

용도 : 연구의 신뢰성을 높이고,
다른 연구자가 연구를 재현할 수 있도록 함

분량 : –

구조 : 배경 이론 (Background) – 선행연구, 이론에 대한 설명

과정과 방법 (Methods) – 실험 과정과 방법을 구체적으로 기술

~~(비슷한 분야를 연구하는 연구자가 읽고 그대로 재현이 가능한 수준으로 기술)~~
과학적 재현성도로 완 해도 됨 !!

* 선행연구는 내 연구와 아주 유사한 경우에만 한 섹션을 만들어서 언급한다.

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

3 Model Architecture

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

Materials & Methods (연구 배경 및 방법)

목적 : 연구를 수행한 방법과 절차를 상세하게 설명

용도 : 연구의 신뢰성을 높이고,
다른 연구자가 연구를 재현할 수 있도록 함

분량 : –

구조 : 배경 이론 (Background) – 선행연구, 이론에 대한 설명

problem formulation – 수식, 구조도 제시

과정과 방법 (Methods) – 실험 과정과 방법을 구체적으로 기술

(비슷한 분야를 연구하는 연구자가 읽고 그대로 재현이 가능한 수준으로 기술)

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{\text{model}}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model}. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by $\sqrt{d_{\text{model}}}$.

Results (연구 결과)

목적 : 수행한 연구로부터 얻은 결과를 객관적으로 제시

용도 : 연구 결과를 통해 연구 가설을 검증

연구 목표 달성 여부를 보여줌

분량 : –

구조 : 실험 결과 – 실험 데이터, 그래프, 표, 이미지 등을 이용해

결과를 명확하게 제시

고찰 (Discussion) – 실험결과의 해석 제시

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

6 Results

6.1 Machine Translation

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.0, outperforming all of the previously published single models, at less than $1/4$ the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate $P_{drop} = 0.1$, instead of 0.3.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty $\alpha = 0.6$ [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [38].

Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU [5].

6.2 Model Variations

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the

[5] We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

실험결과 작성

중요한 결과는 도표로 만들거나 그림을 통해 확인할 수 있게 한다.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

Conclusions (결론)

목적 : 연구 결과의 의미를 해석하고 , 연구의 중요성 및 영향을 강조

용도 : 연구 결과의 해석 , 연구의 한계점 , 향후 연구방향 제시

분량 : 일반적으로 1 page 이내

구조 : 연구결과의 요약 , 결과의 해석 ,
연구의 한계 및 향후 연구에 대한 제안

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

increased the maximum output length to input length + 300. We used a beam size of 21 and $\alpha = 0.3$ for both WSJ only and the semi-supervised setting.

Our results in Table 4 show that despite the lack of task-specific tuning our model performs surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar [8].

In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the Berkeley-Parser [29] even when training only on the WSJ training set of 40K sentences.

7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

The code we used to train and evaluate our models is available at `https://github.com/tensorflow/tensor2tensor`.

Acknowledgements We are grateful to Nal Kalchbrenner and Stephan Gouws for their fruitful comments, corrections and inspiration.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

서론 (Introduction)

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez*† University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Pol illia.polosu

초록 (Abstract)

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

3 Model Architecture

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

연구 배경 및 방법 (Materials & Methods)

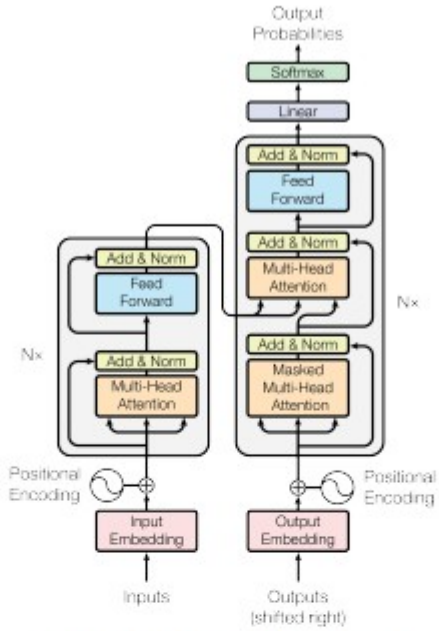

Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

3.1 Encoder and Decoder Stacks

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum

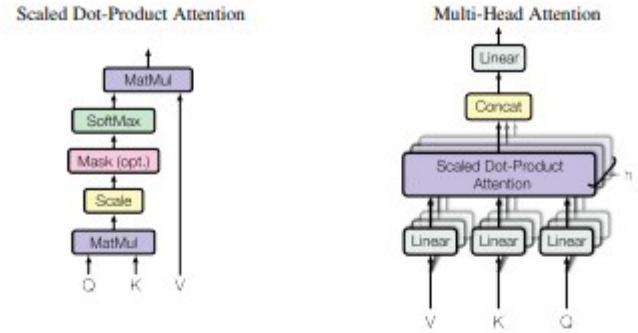

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

3.2.2 Multi-Head Attention

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{\text{model}}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model}. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by $\sqrt{d_{\text{model}}}$.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

In this work, we use sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations $(x_1, ..., x_n)$ to another sequence of equal length $(z_1, ..., z_n)$, with $x_i, z_i \in \mathbb{R}^d$, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence

[4] To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k.

length n is smaller than the representation dimensionality d, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece [38] and byte-pair [31] representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position. This would increase the maximum path length to $O(n/r)$. We plan to investigate this approach further in future work.

A single convolutional layer with kernel width $k < n$ does not connect all pairs of input and output positions. Doing so requires a stack of $O(n/k)$ convolutional layers in the case of contiguous kernels, or $O(log_k(n))$ in the case of dilated convolutions [18], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of k. Separable convolutions [6], however, decrease the complexity considerably, to $O(k \cdot n \cdot d + n \cdot d^2)$. Even with $k = n$, however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

5 Training

This section describes the training regime for our models.

5.1 Training Data and Batching

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared source-target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [38]. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

5.2 Hardware and Schedule

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

5.3 Optimizer

We used the Adam optimizer [20] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \quad (3)$$

This corresponds to increasing the learning rate linearly for the first $warmup\_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup\_steps = 4000$.

5.4 Regularization

We employ three types of regularization during training:

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

6 Results

6.1 Machine Translation

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.0, outperforming all of the previously published single models, at less than $1/4$ the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate $P_{drop} = 0.1$, instead of 0.3.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty $\alpha = 0.6$ [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [38].

Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU [5].

6.2 Model Variations

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

[5] We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

In Table 3 rows (B), we observe that reducing the attention key size d_k hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [9], and observe nearly identical results to the base model.

6.3 English Constituency Parsing

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37].

We trained a 4-layer transformer with $d_{model} = 1024$ on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters English-to-German base translation model. During inference, we

연구 결과
(Results)

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

increased the maximum output length to input length + 300. We used a beam size of 21 and $\alpha = 0.3$ for both WSJ only and the semi-supervised setting.

Our results in Table 4 show that despite the lack of task-specific tuning our model performs surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar [8].

In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the Berkeley-Parser [29] even when training only on the WSJ training set of 40K sentences.

7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

The code we used to train and evaluate our models is available at `https://github.com/tensorflow/tensor2tensor`.

결론 (Conclusions)

Acknowledgements We are grateful to Nal Kalchbrenner and comments, corrections and inspiration.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

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

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841. ACL, August 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

참고문헌 (references)

Abstract (초록)

LSTM과 Bi-LSTM을 이용한 개인 성향분석

유하영, 문성욱, 김재현*, 조영완*
서경대학교
bluebarry37@naver.com, uky6202@naver.com,
statsr@skuniv.ac.kr*, ywcho@skuniv.ac.kr*

Personality Analysis using LSTM and Bi-LSTM

Ryu Ha Yeong, Moon Seong Uk, Kim Jae Hyun*, Cho Young Wan*
Seokyeong Univ

요 약

본 논문은 딥러닝 기법(LSTM과 Bi-LSTM)을 이용해 MBTI의 E와 I 성격 유형을 나양한 구조로 학습하고, 그에 대한 성능 평가 및 성격 유형을 분류하는 모델을 제안한다. 학습 데이터의 사용 범위와 사전학습된 임베딩 벡터의 사용에 따른 영향, 두 가지 측면에서 5가지의 학습 환경으로 모델을 훈련하였다. 실험 결과 모든 모델이 약 90% 이상의 학습 정확도를 보였으며, 실제 성격 예측 단계에서는 Bi-LSTM이 LSTM에 비해 높은 예측률을 보였다. Bi-LSTM 모델 중에서도 사전학습된 임베딩 벡터와 전체 단어학습을 결합한 Bi-LSTM 모델이 가장 많은 단어를 예측하였다.

연구배경

(Background)

연구목표 (Objective)

연구방법 (Methods)

결과 (Result)

결론 (Conclusion)

Introduction (서론)

I. 서 론

사람의 성향이나 심리를 이해하기 위해서는 의학적, 언어적 분석 등이 존재한다. 그 중, 언어적 심리분석 분야에서는 자연어 처리를 기반으로 한 인공지능 기법이 새롭게 주목받고 있다. 과거에는 인공지능의 문장 분석 능력이 긍정 및 부정 분류와 같은 한정된 문제만 해결 가능했다. 하지만 인공지능이 발전함에 따라 고도화된 언어모델을 통해 사람의 심리와 성향까지 탐색할 수 있게 되었다.

최근, 새로운 성격 유형 분석 도구인 MBTI가 등장하여 인간의 심리 이해에 큰 지표를 제시하고 있다. MBTI(Myers-Briggs Type Indicator)는 사용자의 4가지의 선호 경향을 기반으로 16개의 성격 유형으로 분류하는 도구이다. 전통적인 MBTI 검사 방식은 사용자가 객관식 질문에 응답하도록 설계되어 있다. 이러한 방식의 검사는 사용자의 응답 범위를 제한하며, 인간의 복잡한 심리를 묘사하기 어렵다는 한계점이 존재한다. 다시 말해, 사용자는 '그렇다'와 '그렇지 않다'라는 단순한 선택만 할 수 있어, 검사 결과의 정확성에 의문을 제기할 수 있다. 만약 사용자가 주관식으로 자신의 생각과 감정을 표현할 수 있다면, MBTI의 성격 유형을 더욱 정밀하게 파악할 수 있을 것이다. 이러한 문제점을 극복하기 위하여, 텍스트 데이터를 활용하여 MBTI 성격 유형을 분석하고 예측하는 모델을 제안한다.

본 논문은 MBTI 성격 유형 중 'EI preference (Extraversion or Introversion, 이하 유형 E/유형 I)'을 중심으로 분석을 수행한다.[1] E 유형과 I 유형을 구분하는 방법으로는 binary-class를 적용하였다. 실험에 사용할 데이터는 GPT-4와 GPT-2를 혼용해 생성하고, 자연어 처리에서의 데이터 증강 기법을 이용해 데이터의 수를 늘렸다.

제작된 데이터 셋을 기반으로 각 유형의 특징을 분석하기 위해 자연어처리의 기본적인 모델인 LSTM 및 Bi-LSTM을 활용했다.

모델 학습에는 사전학습된 임베딩 벡터를 사용한 것과 이를 사용하지 않은 경우와의 차이를 비교하고, 학습 데이터 선택 범위를 적용하여 그에 따른 성능 변화를 분석하였다. 마지막으로 개발된 모델을 바탕으로 실제 텍스트 데이터에 대한 예측을 진행해보고, 이를 바탕으로 적합한 성격 유형을 판단하는 모델을 제시한다.

연구 주제의 배경

연구 문제

연구의 중요성 및 필요성

연구의 목표 및 가설제시

Materials & Methods (연구 배경 및 방법)

배경 이론 (Background)

II. LSTM을 이용한 MBTI 유형 분석

2.1 선행 연구

국내 인터넷 커뮤니티 사이트의 게시글을 활용한 선행 연구에서는 LSTM 신경망으로 약 33,000개의 데이터를 학습하였고, 그 결과 예측 정확도는 20.29%로 나타났다.[2] 선행 연구는 크롤링을 사용하여 데이터를 수집하였지만, 본 논문에서는 GPT를 활용해 새로운 학습 데이터를 생성한다.

타 연구에서는 MBTI 성격 유형 분류를 위한 방식을 제안한다.[3] 16개의 multi-class 성격 유형을 개별적으로 분류하는 16-class classifier이고, 두 번째는 4가지의 선호 경향을 binary-class로 분류한 뒤 조합하는 4 binary classifier이다. 전자의 방식은 23%의 Test Accuracy를 보였고, 후자의 방식에서는 38%로 전자의 방식보다 높은 정확도를 보였다. 이러한 결과는 4 binary classifier 방식이 MBTI의 특징을 더 정확하게 반영하고 있음을 알 수 있으며, 본 논문에서도 이와 같은 방식을 활용한다.

2.2 LSTM

LSTM(Long Short-Term Memory)은 시퀀스 데이터에서 장기 의존성 문제를 해결하기 위해 고안된 RNN의 한 형태이다.[4] LSTM은 과거의 정보를 통해 현재의 컨텍스트를 이해하는데 유용하며, 이 특징은 Vanilla RNN에 비해 텍스트와 같은 시퀀스 데이터 처리에 효율적이다.

그러나 입력 시퀀스가 길어질수록 초기 정보를 잃어버리는 장기 의존성 문제를 해결하기 위해 역방향으로도 정보를 처리하는 Bi-LSTM(Bidirectional LSTM)을 도입하였다.[5] 이 구조는 과거와 미래의 정보를 동시에 활용하여 정확도를 향상시킨다.

본 논문은 LSTM과 Bi-LSTM 두 모델에 대한 실험을 진행하고 성능을 비교한다.

III. 분류 모델 비교 실험

3.1 데이터 수집

본 연구는 16개의 성격 유형 중 E 유형과 I 유형을 중점적으로 분류하는 것을 주목표로 설정하였다. 분류 방식은 이전 연구와 마찬가지로 binary-class를 사용하였다. 성격 유형 E에 해당하면 1, 유형 I면 0으로 labeling 하였다.

성격 유형의 분류를 위해 각 유형 사용자의 특성을 정확하게 파악해야 한다. 이에 두 유형에 대한 응답 데이터를 학습용으로 구성하였다. 학습 데이터 생성에 앞서, 주어진 질문을 바탕으로 응답 데이터를 구성할 것이기 때문에 질문 데이터 셋을 먼저 생성했다. 이를 위해 16 Personalities와 같은 대중적인 검사에서 사용되는 질문들을 수집하고 분석한 뒤, 총 20개의 질문을 새롭게 제작하였다.[6]

기초 응답은 각 질문에 대해 대략 30개씩 작성했다. 이후 GPT-4를 이용해 E와 I 유형의 특징이 분명하게 드러나는 약 70개의 모범답안을 생성하였다. 앞서 제작한 응답을 바탕으로 KoGPT-2 모델에 기초 응답과 모범답안을 결합한 데이터 셋으로 fine-tuning한 후, 응답 데이터의 다양성을 확장하여 최종적으로 총 1,375개의 질문-응답 데이터 셋을 만들었다. KoGPT-2는 GPT-4만을 이용하여 데이터를 생성한 방법보다 비용적 면에서 합리적이고 실험목적에 맞게 수정 가능하다는 장점이 있기에, 이와 같은 방법을 적용하였다. 데이터 셋 중 1,000개는 Training set에 이용하였고, 나머지 375개는 Validation set으로 사용하였다.

과적합을 방지하기 위하여 1,000개의 Training set에 대하여 Data Augmentation을 진행하였다.[7] 본 논문은 한국어에 적합하다고 판단되는 RS(Random Swap)와 RD(Random Deletion)를 사용했다. 이를 통해 한 응답 데이터당 3개의 문장으로 확장하여 총 3,000개의 문장을 생성하고 중복된 문장을 제거하여 2,965개의 증강된 데이터를 학습에 사용하였다. 성능 테스트를 위해서 사람들이 자유롭게 작성한 38개의 응답을 수집하여 활용하였다.

구축된 데이터는 모두 한국어 정보처리 패키지 Konlpy의 Mecab을 이용하여 형태소 분석을 진행하였다.[8]

3.2 모델 제안

본 연구에서는 더 적합한 딥러닝 모델을 제안
다양한 접근법을 시도하였다.

첫 번째로, 임베딩 레이어의 선택에서 기본
된 FastText 임베딩 중 어느 것이 더 효과적
FastText 임베딩은 3만 개의 한국어 단어로 사
(FastText-KO)를 활용한다.[9]

두 번째로, 학습 데이터 선택 범위에 따른 성능 변화를 관찰하였다. 문법적 역할을 하는 조사나 어미와 같은 형태소는 제거하고, 주요 정보를 담고 있는 명사, 동사, 형용사, 그리고 어간을 추출하여 사용한 extracted word와 추출하지 않은 전체 문장 데이터인 whole word에 따른 성

따라서, 이 연구에서는
과 Bi-LSTM을 사용하여
(표 1)에는 각 모델의 조합

였다. 이후 본 논문에서는 5가지 조합 모델을 (표 1)의 model name에 정의되어있는 이름으로 명칭 한다.

(표 1) 분류 모델별 학습 방식 비교

Classification model	Use of FastText	Training Data Scope	Model name
LSTM	X	extracted word	Simple-LSTM
LSTM	O	extracted word	Fa-LSTM
Bi-LSTM	X	whole word	Simple-BLSTM
Bi-LSTM	O	extracted word	Fa-extr-BLSTM
Bi-LSTM	O	whole word	Fa-whl-BLSTM

3.3 성격 유형 정의

제안한 5개 모델에 대해 학습을 진행한 후, 최종적으로 실제 입력한 응답을 바탕으로 사용자의 MBTI를 잘 예측하는지 실험하였다. 사용자가 주어진 질문에 대답하면, Konlpy의 mecab을 이용해 응답을 형태소 단위로 분해하며 전처리 과정을 거친다. 전처리된 데이터는 모델에 입력해 예측값을 출력한다. 예측값이 0.5 이상이면 사용자의 성격 유형을 E 유형으로 판단하고, 0.5 미만이면 I 유형으로 판단한다. 이와 함께 어느 정도로 해당 성격 유형에 가까운지 출력한다. 만일 예측값이 0.1564라면, 이는 사용자의 MBTI E/I 유형이 84.36% 정도로 I 유형임을 의미한다.

LSTM 사용이유에 대한 설명이 부족.

과정과 방법 (Methods)

Results (연구 결과)

고찰 (Discussion)

Ⅳ. 실험결과

본 연구의 목표는 MBTI의 E와 I 유형을 분류하는 것이다. 사전학습된 임베딩 벡터의 사용 여부를 기준으로 FastText 임베딩 벡터와 결합한 Fa-LSTM, Fa-BLSTM 모델을 사용하였다. 추가로 Bi-LSTM에서는 학습 데이터에서 extracted word을 사용해 학습한 Fa-extr-BLSTM과 whole word을 사용한 Fa-whl-BLSTM에 따른 성능 비교를 진행했다.

그 외에 과적합을 방지하기 위한 전략을 적용했다. 초기 10 epoch 동안에는 Adam 최적화 함수의 학습률을 0.0001로 낮추어 안정적인 학습이 이루어질 수 있도록 하였다. 이후의 epoch에서는 기본 학습률로 학습을 진행하였다.

또한, 모델에는 드롭아웃, 배치 정규화, 그리고 Leaky ReLU 활성화 함수를 사용하였다. 학습 과정 중에 Validation 데이터의 loss 값이 증가하는 경우, Early Stopping을 활용하여 학습을 중단해주었다. 이러한 접근법들을 통해 모델의 성능을 최적화해 나갔다. (표 2)에서는 5개의 각 모델에 대한 정확도를 나타낸다.

(표 2) 분류 모델별 성능 비교

Model name	Train	Validation	Test
Simple-LSTM	95.10%	93.60%	92.11%
Fa-LSTM	97.50%	93.30%	86.84%
Simple-BLSTM	**98.00%**	94.40%	89.29%
Fa-extr-BLSTM	95.50%	93.60%	**94.74%**
Fa-whl-BLSTM	**98.00%**	95.30%	**94.74%**

제시된 모델들은 Train 및 Validation 과정에서 90% 이상의 정확성을 보였다. 이러한 높은 성능의 원인은 두 가지로 볼 수 있다. 첫째로, E와 I 유형을 구분하는 학습 데이터 셋은 주로 유사한 문장구조를 가지기 때문이다. 대부분의 질문이 성격 유형을 구분하는 특징적인 질문들이기 때문에, 답변도 비슷한 패턴을 보이는 경향이 있다. 둘째로, MBTI의 각 유형은 서로가 상반되는 유형이다. '외향'과 '내향'이라는 명확한 특징은 모델 학습에서 강력한 분류 기준을 제공한다. 따라서 이러한 특징을 기반으로 모델은 각 유형을 효과적으로 구분할 수 있다.

그러나 단순히 정확성만으로 모델의 성능을 판단하는 것은 한계가 있다. 이에 모델들에 대한 세부적인 능력을 검증하기 위해 단어와 문장에 대한 성격 유형 예측을 추가로 시행하였다.

(표 3) 성격 유형 분류를 위한 대표적인 단어와 문장 예시

Classification Criteria	Word/Sentence	Label	case Identifier
1. Clear Distinction Word	혼자	I	**case 1.1**
	친구	E	**case 1.2**
2. Neutral Word	사람	Neutral	**case 2**
3. Classification Indicator Sentence	친한 친구와만 노는 편이다.	I	**case 3.1**
	많은 친구와만 노는 편이다.	E	**case 3.2**
4. User-defined Sentence	주목받은 후 즉시는 살짝 부끄럽지만, 나를 주목받는 것을 좋아합니다.	E	**case 4.1**
	예기치 못한 상황이면 살짝 당황스러움..	I	**case 4.2**

(표 3)에서는 모델의 성능 평가를 위한 다양한 텍스트 데이터를 제시한다. case 1은 E 유형과 I 유형을 명확히 구분하는 단어를 기반으로 한다. case 2는 E와 I 유형 사이에 있는 중립적인 단어를 선정하였다. 예를 들어 '사람'이라는 단어는 문맥에 따라 다르게 해석될 수 있다. '사람들을 만나는 것을 즐긴다.'라는 문장은 E 유형을 나타내지만, '사람들을 만나는 것을 꺼린다.'는 I 유형을 더 잘 반영하기 때문이다. case 3은 case 1에 따른 단어를 문장으로 확장하여 제시한다. case 4에서는 실제 사용자가 작성한 원본 문장을 사용했다. 여기에는 '당황스러움..'과 같은 정제되지 않은 문장구조도 포함된다. (표 3)에서 제시된 단어와 문장을 통해 각 모델의 MBTI E/I 유형 예측 능력을 평가하였다.

(표 4) 모델별 성격 유형 예측 성능 분석 - LSTM

case	y	model: Simple-LSTM Pred	Simple-LSTM $\hat{y}$	Fa-LSTM Pred	Fa-LSTM $\hat{y}$
case 1.1	I	80.47%	**I**	95.67%	**I**
case 1.2	E	78.10%	**E**	97.18%	**E**
case 2	(N)	**52.85%**	I	94.65%	E
case 3.1	I	61.89%	E	97.66%	**I**
case 3.2	E	83.60%	I	74.04%	**E**
case 4.1	E	97.79%	I	75.82%	I
case 4.2	I	84.09%	**I**	99.85%	**I**

(표 4)에서 1행의 Pred는 case에 대한 모델의 예측값을 의미하고, y는 각 Case의 실제 MBTI이며 $\hat{y}$는 해당 Case를 학습된 모델에 적용했을 때의 예측 MBTI 결과를 표시한다. 2열의 (N)은 Neutral word를 의미한다. LSTM 기반의 모델 중 Simple-LSTM은 제시된 7개의 예시 중 4개를 올바르게 예측했고, Fa-LSTM은 5개를 정확하게 예측하였다. 이를 보아 사전학습된 임베딩 벡터를 사용한 Fa-LSTM이 Simple-LSTM에 비해 E/I 성격 유형을 잘 판단한다는 것을 알 수 있다.

(표 5) 모델별 성격 유형 예측 성능 분석 - Bi-LSTM

case	y	model: Simple-BLSTM Pred	Simple-BLSTM $\hat{y}$	Fa-extr-BLSTM Pred	Fa-extr-BLSTM $\hat{y}$	Fa-whl-BLSTM Pred	Fa-whl-BLSTM $\hat{y}$
case 1.1	I	73.15%	**I**	91.37%	**I**	80.59%	**I**
case 1.2	E	93.99%	**E**	99.33%	**E**	90.54%	**E**
case 2	(N)	**70.57%**	E	97.31%	E	**61.94%**	E
case 3.1	I	72.75%	**I**	66.29%	**I**	58.92%	**I**
case 3.2	E	99.87%	**E**	98.79%	**E**	98.56%	**E**
case 4.1	E	93.76%	I	70.19%	**E**	91.92%	**E**
case 4.2	I	99.87%	I	98.83%	**I**	68.57%	**I**

(표 5)에 따르면 Bi-LSTM 기반 모델들은 LSTM보다 성능이 우수한 것을 확인할 수 있다. 세 모델 모두 대부분의 case를 정확하게 예측하였다. 하지만 비교적 예측이 어려운 case 4에 대해서는 Simple-BLSTM이 2개 중 1개만 정답을 맞혔고, 사전 학습된 임베딩 벡터를 이용한 Fa-ext-BLSTM와 Fa-whl-BLSTM은 모두 정확하게 예측했다.

case 2의 중립단어를 예측할 때는 예측값이 50%에 가까울수록 잘 예측하는 것이다. 위에서 정의한 '사람'이란 용어는 E, I 유형 모두 사용할 수 있는 단어이기 때문이다. case 2에 대해서 Simple-BLSTM과 Fa-whl-BLSTM 모델은 중립단어에 대해 거의 50%에 가까운 예측 성능을 보였으나, Fa-ext-BLSTM은 97.31% 정도로 E 유형에 가까운 예측을 하였다. 이는 extracted word에 대해서만 학습한 Fa-ext-BLSTM 모델은 중립단어나 모호한 단어와 같이 문맥에 의해 좌우되는 단어 잘 반영하지 못한다는 것을 알 수 있다.

종합적으로 Fa-whl-BLSTM 모델은 모든 단어에 대해 정답을 맞히며 좋은 예측 성능을 보여주었다. 이를 통해 네 가지 주요한 결론을 도출하였다. 첫째, 이전 연구 [2]는 크롤링을 통한 데이터 수집 방법을 사용했으나, 본 연구에서는 GPT를 활용해 양질의 데이터를 생성해 학습하였다. 데이터가 양이 적었음에도 유의미한 결과가 나타난 것으로 보아, 데이터의 질이 양보다 중요하다는 것을 시사할 수 있다. 둘째, 학습 시 사전에 대량으로 학습된 임베딩 벡터를 가중치로 사용하는 것이 유리하다. 셋째, 기존의 연구대로 LSTM보다는 Bi-LSTM을 사용했을 때 예측 능력이 향상된다.[10] 마지막으로 Bi-LSTM은 whole word로 학습하는 것이 extracted word로 학습하는 것보다 문맥 정보를 더욱 잘 파악하는 데 있어 효과적이었다.

Conclusions（결론）

결과 요약

한계
후속연구
제안

V. 결론

본 논문은 MBTI의 E/I preference를 대상으로 E와 I 유형 분류에 LSTM과 Bi-LSTM 기법을 적용하여 분석하였다. 각 유형의 데이터는 GPT를 이용해 고품질의 데이터를 생성하였고, 자연어 처리에서의 증강 기법을 이용해 데이터의 양을 늘려주었다. 학습 과정에서는 내부 파라미터의 조정을 통해 성능을 조절하였고, 한국어로 사전학습된 임베딩 벡터를 사용해 수집한 데이터가 한국어의 다양한 의미적·구문적 정보를 학습할 수 있도록 하였다. 그 뒤 기존 모델과 개선한 모델까지 총 5가지 모델을 제안하여 다양한 환경에서 학습을 시도하였다. 그 결과, 약 1,000개의 데이터만을 수집했음에도 불구하고 대부분의 모델이 90%이상의 정확성을 보였으며, (표 3)의 실제 성격 유형도 대부분의 case에 대하여 잘 예측했다. 결과적으로 제안된 5개의 모델 중, FastText를 활용해 whole word에 대해서 학습한 Fa-whl-BLSTM 모델이 특히 좋은 예측 성능을 보였다.

하지만 해당 실험에서도 일부 한계가 존재한다. LSTM 기반의 모델은 학습 데이터에 없는 새로운 단어인 OOV(Out Of Vocabulary)에 대해서는 예측하지 못한다. 만일, 사용자가 학습되지 않은 단어를 포함하여 응답을 작성한다면, 해당 단어들은 모두 OOV 처리가 되어 예측 성능에 영향을 줄 수 있다. 이를 극복하기 위해 더 발전된 모델인 Transformer와 단어를 더 작은 단위로 나누어서 학습하는 Subword Tokenization 기법을 이용하여 위의 문제를 해결하고, 복잡한 단어 간의 관계를 파악할 수 있다. 후속 연구 주제로 앞서 언급한 문제점을 보완하고, 더욱 정교한 MBTI 유형 예측 모델을 개발하고자 한다.

참 고 문 헌

[1] Isabel Briggs Myers, Peter B. Myers, 『Gifts Differing: Understanding Personality Type』, CPP, 2010.

[2] 김정민, 박지민, 이로운, 조서원, 신재형, "딥러닝 기반의 MBTI 성격유형 분류 연구", 한국통신학회 하계종합학술대회 논문집, p. 1,740-1,741, 2022.

[3] Cui, Brandon and Calvin Qi., "Survey Analysis of Machine Learning Methods for Natural Language Processing for MBTI Personality Type Prediction.", 2017.

[4] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, p. 1735-1780, 1997.

[5] Graves, Alex, and Jürgen Schmidhuber., "Framewise phoneme classification with bidirectional LSTM and other neural network architectures.", Neural networks 18, p. 602-610, 2005.

[6] NERIS Analytics Limited, "16-personalities", https://www.16personalities.com/ko, 2023.

[7] Wei, Jason and Kai Zou., "EDA: Easy Data

감사합니다