Improving Language Understanding by Generative Pre-Training

Tags	Empty
Created	4월 16, 2019 7:14 오후
Updated	4월 22, 2019 6:48 오후
Add a Property	

Add a comment...

0. Paper

language_understanding_paper.pdf 1940.0KB

1. Introduction

Tranning Procedure

1. Use a language modeling objective on the unlabeled data to learn the initial parameters of a neural network model.
2. Adapt these parameters to a target task using the corresponding supervised objective.

Use Transformer

1. for handling long-term dependencies in text, compared to alternatives like recurrent networks
2. resulting in robust transfer performance across diverse tasks

During Transfer

we utilize task-specific input adaptations derived from traversal-style approaches which process structured text input as a single contiguous sequence of tokens

2. Related Work

Semi-supervised learning for NLP

1. Transfer word-level information, whereas we aim to capture higher-level semantics.
2. Phrase-level or sentence-level embeddings

Unsupervised pre-training

1. Unsupervised pre-training is a special case of semi-supervised learning where the goal is to find a good initialization point instead of modifying the supervised learning objective.

Auxiliary training objectives

1. Adding auxiliary unsupervised training objectives is an alternative form of semi-supervised learning

3. Framework

3.1. Unsupervised pre-training

1. use a standard language modeling objective to maximize the following likelihood:

$$\mathcal{U} = \{u_1, \ldots, u_n\}$$

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

- k: the size of the context window
- P: the conditional probability is modeled using a neural network with parameters Θ
- Θ: parameters are trained using stochastic gradient descent

1. use multi-layer Transformer decoder

$$U = (u_{-k}, \ldots, u_{-1})$$

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

- n: the number of layers
- We: the token embedding matrix
- Wp: the position embedding matrix

3.2. Supervised fine-tuning

- x1,...,xm: the input tokens
- y: the label
- h: the final transformer block's activation
- w: the output layer parameters

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y)$$

the following objective to maximize:

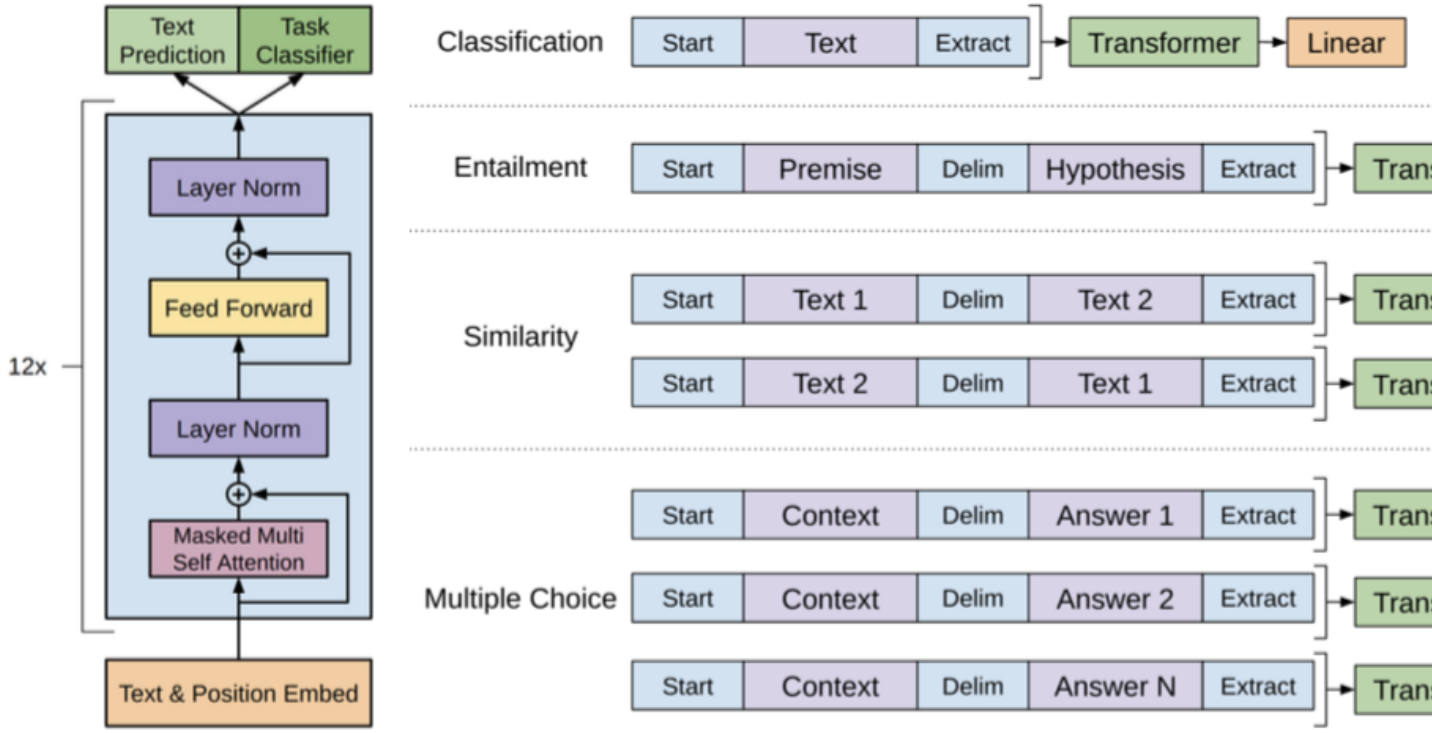
$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$$

Specifically, we optimize the following objective (with weight λ):

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

3.3. Task-specific input transformations

- use a traversal-style approach
- convert structured inputs into an ordered sequence that our pre-trained model can process
- input transformations allow to avoid making extensive changes to the architecture across tasks



Textual entailment

For entailment tasks, we concatenate the premise p and hypothesis h token sequences, with a delimiter token ($) in between

Similarity

For similarity tasks, we modify the input sequence to contain both possible sentence orderings and process each independently to produce two sequence representations h which are added element-wise before being fed into the linear output layer.

Question Answering and Commonsense Reasoning

For these tasks, we are given a context document z, a question q, and a set of possible answers {ak}. We concatenate the document context and question with each possible answer, adding a delimiter token in between to get [z;q;$;ak].

4. Experiments

4.1 Setup

Unsupervised pre-training

- the BooksCorpus dataset for training the language model.
- An alternative dataset the 1B Word Benchmark, which is used by a similar approach, ELMo
- Our language model achieves a very low token level perplexity of 18.4 on this corpus.

Model specifications

- 12-layer decoder-only transformer with masked self-attention heads (768 dimensional states and 12 attention heads)
- For position-wise feed-forward networks, we used 3072 dimensional inner states
- Adam optimization scheme with a max learning rate of 2.5e-4. The learning rate was increased linearly from zero over the first 2000 updates and annealed to 0 using a cosine schedule.
- train for 100 epochs on minibatches of 64 randomly sampled, contiguous sequences of 512 tokens
- layernorm is used extensively throughout the model, a simple weight initialization of N (0, 0.02) was sufficient.
- We used a bytepair encoding (BPE) vocabulary with 40,000 merges and residual, embedding, and attention dropouts with a rate of 0.1 for regularization.
- We also employed a modified version of L2 regularization proposed in, with w = 0.01 on all non bias or gain weights
- the activation function, we used the Gaussian Error Linear Unit (GELU)
- We used learned position embeddings instead of the sinusoidal version proposed in the original work
- We use the ftfy library2 to clean the raw text in BooksCorpus, standardize some punctuation and whitespace, and use the spaCy tokenizer.

Fine-tuning details

- We add dropout to the classifier with a rate of 0.1.
- For most tasks, we use a learning rate of 6.25e-5 and a batchsize of 32.

- Our model finetunes quickly and 3 epochs of training was sufficient for most cases.

- We use a linear learning rate decay schedule with warmup over 0.2% of training. λ was set to 0.5.

4.2. Supervised fine-tuning

Task	Datasets
Natural language inference	SNLI [5], MultiNLI [66], Question NLI [64], RTE [4], SciTail [25]
Question Answering	RACE [30], Story Cloze [40]
Sentence similarity	MSR Paraphrase Corpus [14], Quora Question Pairs [9], STS Benchmark [6]
Classification	Stanford Sentiment Treebank-2 [54], CoLA [65]

Natural Language Inference

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

- On RTE, one of the smaller datasets we evaluate on (2490 examples), we achieve an accuracy of 56%, which is below the 61.7% reported by a multi-task biLSTM model. Given the strong performance of our approach on larger NLI datasets.

Question answering and commonsense reasoning

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

Semantic Similarity / Classfiction

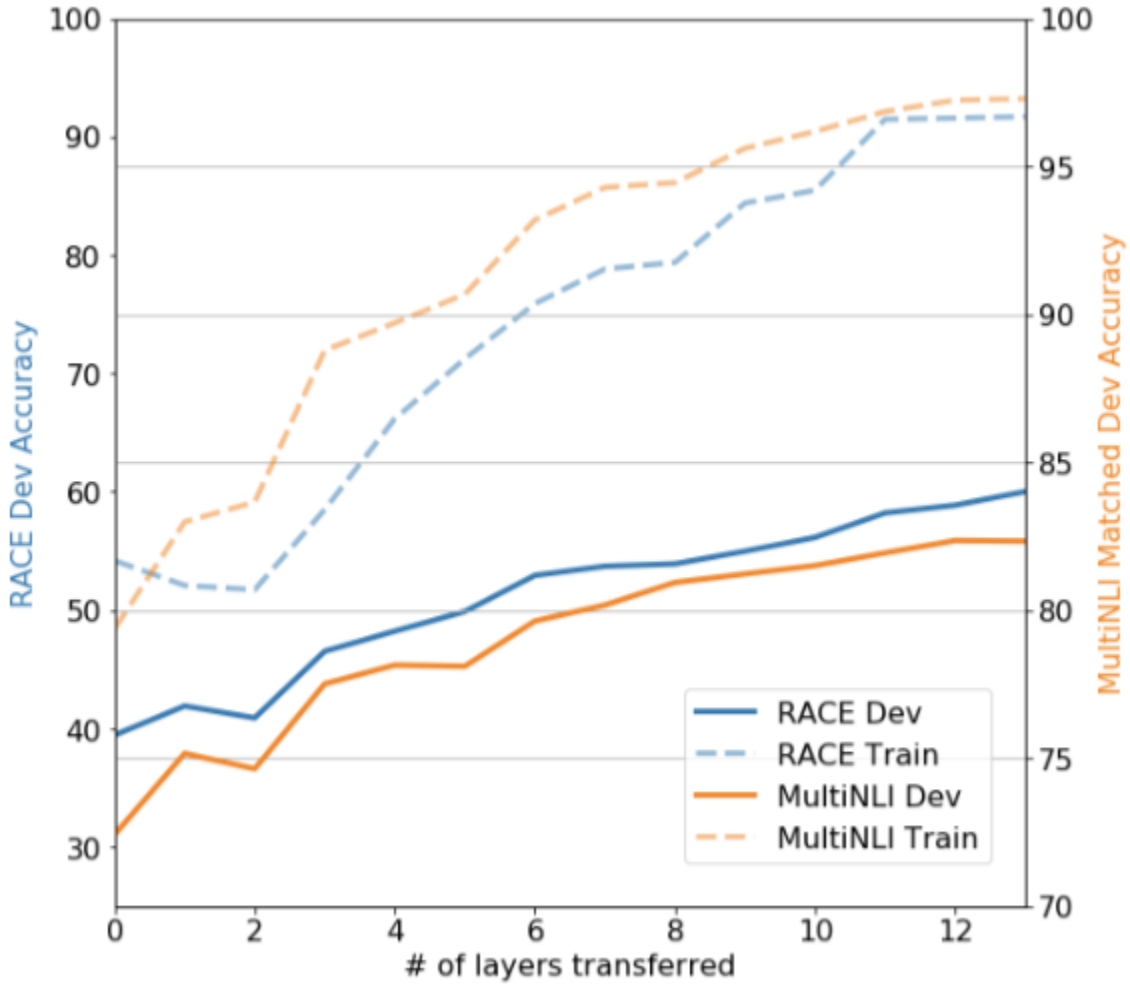
Method	Classification		Semantic Similarity			GLUE
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3	68.9
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**	**72.8**

Result

We achieves new state-of-the-art results in 9 out of the 12 datasets we evaluate on, outperforming ensembles in many cases. Our results also indicate that our approach works well across datasets of different sizes, from smaller datasets such as STS-B (≈5.7k training examples) – to the largest one – SNLI (≈550k training examples)

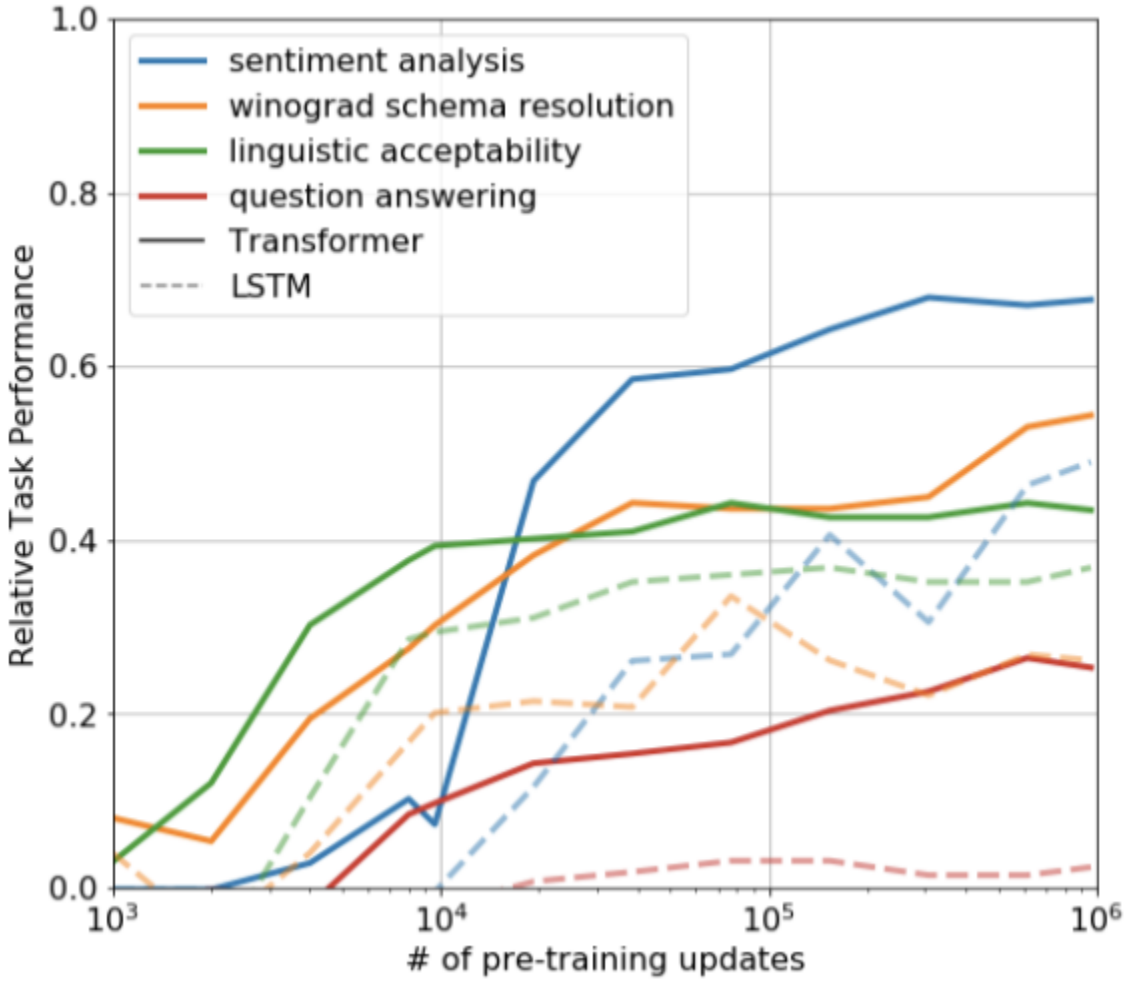
5. Analysis

Impact of number of layers transferred



- We observe the standard result that transferring embeddings improves performance and that each transformer layer provides further benefits up to 9% for full transfer on MultiNLI.
- This indicates that each layer in the pre-trained model contains useful functionality for solving target tasks.

Zero-shot Behaviors



- We observe the performance of these heuristics is stable and steadily increases

Ablation studies

Method	Avg. Score	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	MNLI (acc)
Transformer w/ aux LM (full)	74.7	45.4	91.3	82.3	82.0	**70.3**	**81.8**
Transformer w/o pre-training	59.9	18.9	84.0	79.4	30.9	65.5	75.7
Transformer w/o aux LM	**75.0**	**47.9**	**92.0**	**84.9**	**83.2**	69.8	81.1
LSTM w/ aux LM	69.1	30.3	90.5	83.2	71.8	68.1	73.7

- We observe that the auxiliary objective helps on the NLI tasks and QQP.
- Overall, the trend suggests that larger datasets benefit from the auxiliary objective but smaller datasets do not

- We observe a 5.6 average score drop when using the LSTM instead of the Transformer

- We observe that the lack of pre-training hurts performance across all the tasks, resulting in a 14.8% decrease compared to our full model.

6. Source Code

GIT: https://github.com/openai/finetune-transformer-lm

Data:
https://docs.google.com/spreadsheets/d/1FkdPMd7ZEw_Z38AsFSTzgXeiJoLdLyXY_0B_0JIJIbw/export?format=csv
https://docs.google.com/spreadsheets/d/11tfmMQeifqP-Elh74gi2NELp0rx9JMMjnQ_oyGKqCEg/export?format=csv

train.py

main

traing 시작 점

transform_roc

<start>본문<delimiter>질문1<end>, <start>본문<delimiter>질문2<end> 형태로 변경

mgpu_train

muti gpu를 지원하는 학습 모델 생성 (GPU 당 1개의 모델 생성)

mgpu_predict

muti gpu를 지원하는 예측 모델 생성 (GPU 당 1개의 모델 생성)

model

모델 생성

block

transformer 블락 생성

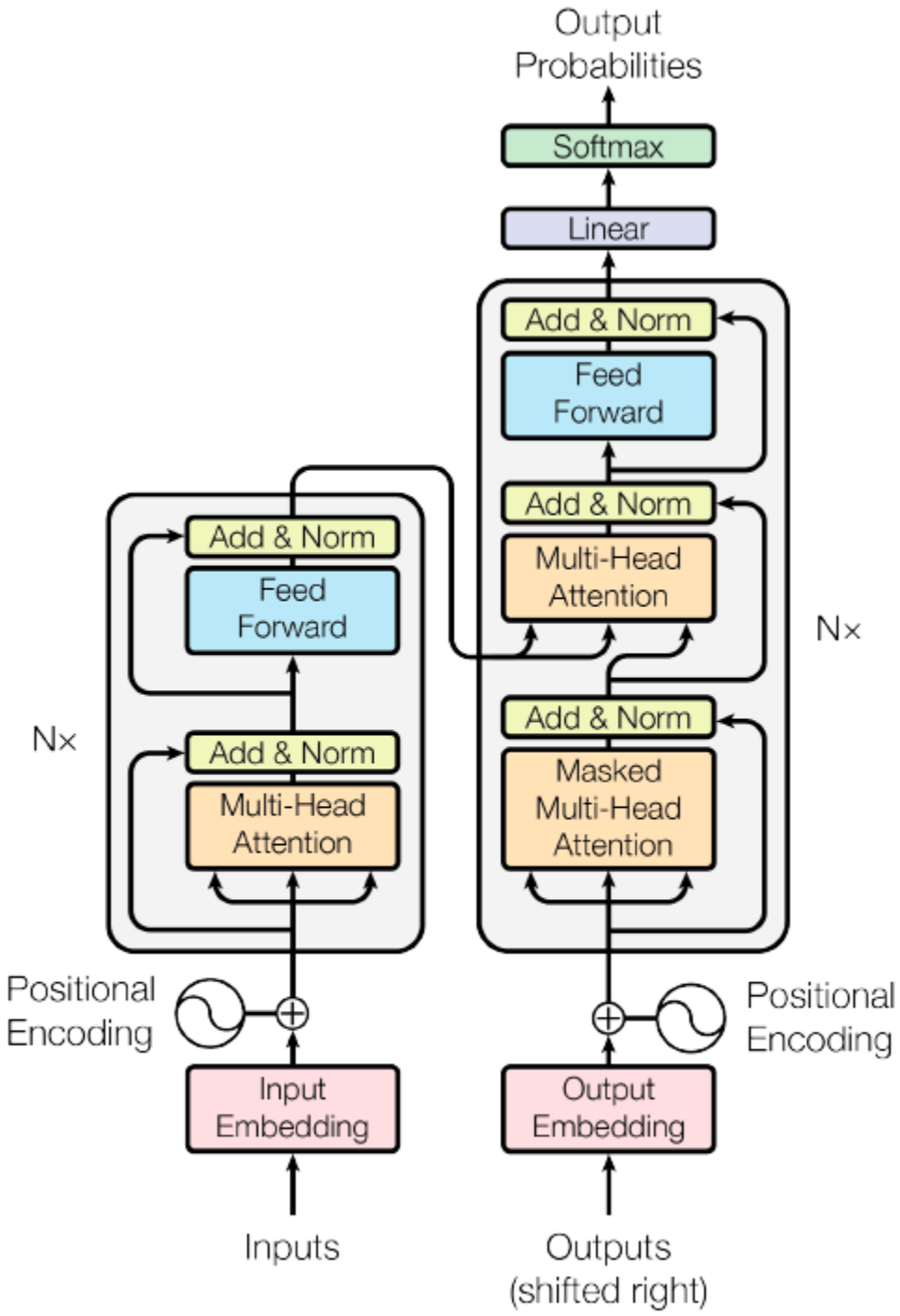
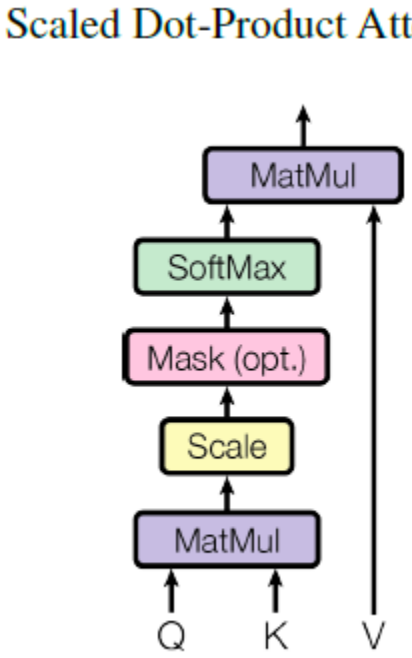


Figure 1: The Transformer - model architecture.

attn

self attention 블락 생성



dataset.py

데이터 파일 로딩

text_util.py

TextEnocder

- bpe

  byte pair encoding : BERT의 wordpeace 와는 다른 방식 임 / 내용이 복잡하여 디버깅으로 설명

- encode

  BPE를 이용하여 text encoding

utils.py

유틸.. 프로그램