Generative pre-training of transformer networks

Grégory Châtel

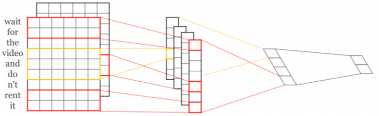
Disaitek
Intel Software Innovator

@rodgzilla
github.com/rodgzilla

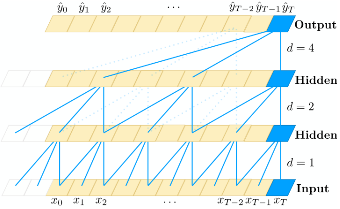
11/17/2018

Traditional architectures for NLP

CNN



Dilated CNN



RNN

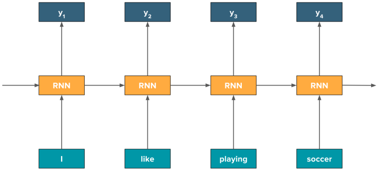


Image from https://techblog.gumgum.com/articles/deep-learning-for-natural-language-processing-part-2-rnns and

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Attention mechanisms

Concept

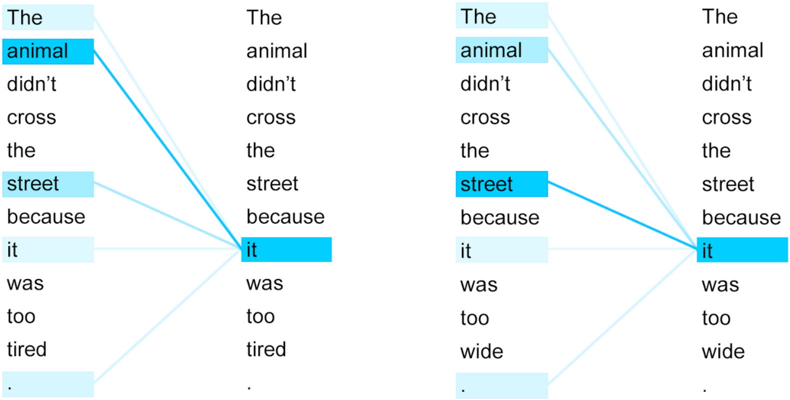


Image from https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Attention mechanisms

Scaled Dot-Product Attention

Input sentence	elle	alla	à	la	plage
Key	subject	verb	filler	filler	location
Value	*she*	*go, past tense*	-	-	*beach*

Output sentence	she	went	to	the	**?????**
Query	subject	verb	filler	filler	location

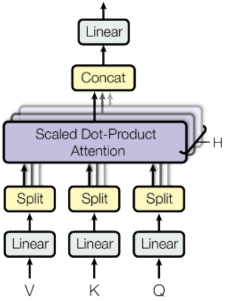
To compute the next word in the translation, the attention mechanism creates a vector using the source sentence and what has been generated so far.

Q, K and V are respectively the query, key and value vectors.

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

Attention mechanisms

Multi-Head Attention

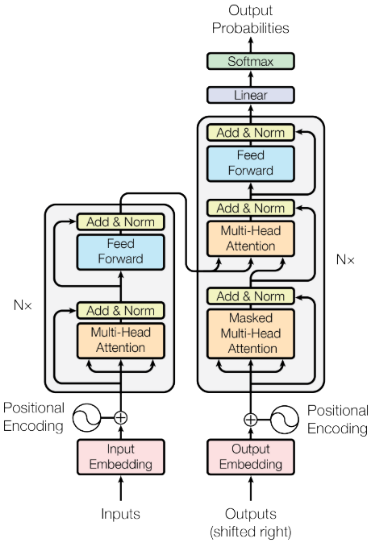


$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)$$
$$\text{where} \quad \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where the projections W_i^Q, W_i^K and W_i^V are parameter matrices.

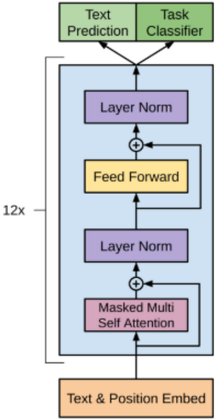
Transformer network

Original transformer



Transformer network

OpenAI multi-layer decoder



W_e is the token embedding matrix

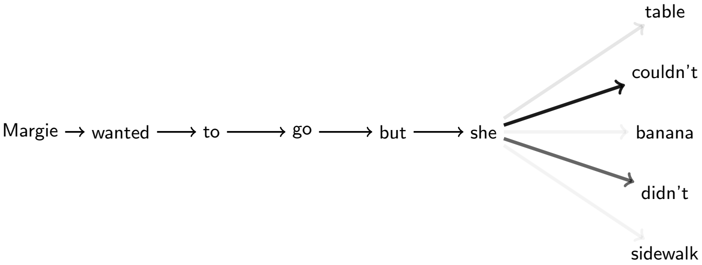
W_p is the position embedding matrix

$$h_0 = UW_e + W_p$$
$$h_l = \text{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$

The *Text Prediction* and *Task classifier* heads take h_n as input.

Unsupervised pre-training task

Language modeling



$$P(u) = \mathrm{softmax}(h_n W_e^T)$$

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

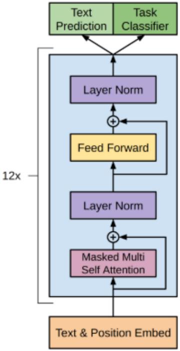
Dataset BooksCorpus (7000 books, $\sim$ 800M words, $\sim$ 5GB of text),

Duration 1 month,

Hardware 8 GPUs.

Supervised fine-tuning

Multitask learning



$$P(u) = \text{softmax}(h_n W_e^T)$$

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

Language modeling loss

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

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

Classification loss

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

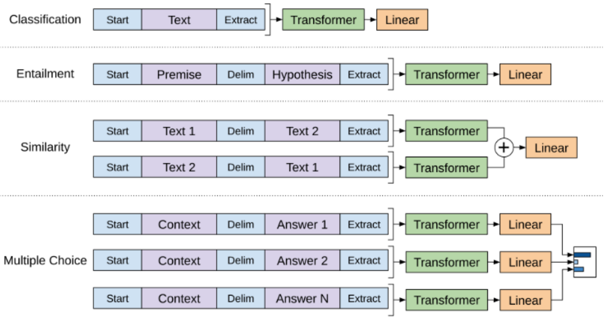
Final loss

Results on standard datasets

New state of the art on the following tasks:

- Textual Entailment
  - SNLI 89.3 → 89.9
  - MNLI Matched 80.6 → 82.1
  - MNLI Mismatched 80.1 → 81.4
  - SciTail 83.3 → 88.3
  - QNLI 82.3 → 88.1
- Semantic Similarity
  - STS-B 81.0 → 82.0
  - QQP 66.1 → 70.3
- Reading Comprehension
  - RACE 53.3 → 59.0
- Commonsense Reasoning
  - ROCStories 77.6 → 86.5
  - COPA 71.2 → 78.6
- Linguistic Acceptability
  - CoLA 35.0 → 45.4
- Multi-Task Benchmark
  - GLUE 68.9 → 72.8

Input formatting



Two possible input shape:

- (batch_idx, token_idx, 2)
- (batch_idx, sequence_idx, token_idx, 2)

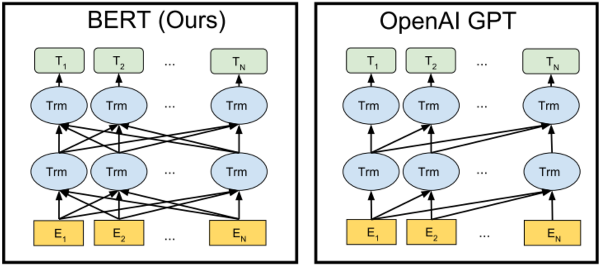
The 2 is there to select either the token embedding or its corresponding position embedding.

Input formatting

```
def transform_imdb(X, encoder, max_len, n_vocab, n_special,
                   n_ctx):
    n_batch   = len(X)
    xmb       = np.zeros((n_batch, n_ctx, 2), dtype = np.int32)
    mmb       = np.zeros((n_batch, n_ctx), dtype = np.float32)
    start     = encoder['_start_']
    clf_token = encoder['_classify_']
    for i, x in enumerate(X):
        x_with_tokens    = [start] + x[:max_len] + [clf_token]
        l_x              = len(x_with_tokens)
        xmb[i, :l_x, 0]  = x_with_tokens
        mmb[i, :l_x]     = 1
    pos_emb_start = n_vocab + n_special
    xmb[:, :, 1]  = np.arange(
        pos_emb_start,
        pos_emb_start + n_ctx
    )

    return xmb, mmb
```

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding



BERT is an improvement on the GPT. The main differences are:

- Bidirectional training,
- Different pre-training tasks (masked language model and next sentence prediction),
- Trained on a much bigger corpus (BookCorpus (800M words) + **Wikipedia (2500M words)**),
- 3 $\times$ as many parameters for the large version,
- Pre-trained model for 102 languages.

BERT produces 11 new states of the art.

References

- Tensorflow GPT: https://github.com/openai/finetune-transformer-lm
- Tensorflow BERT: https://github.com/google-research/bert
- PyTorch GPT: https://github.com/huggingface/pytorch-openai-transformer-lm
- PyTorch BERT: https://github.com/huggingface/pytorch-pretrained-BERT
- IMDB movie review classification: https://github.com/rodgzilla/pytorch-openai-transformer-lm/tree/movie_reviews_classification
- Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv preprint arXiv:1803.01271 (2018).
- Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. 2017.
- Radford, Alec, et al. "Improving language understanding by generative pre-training." URL Article pdf link Blog post (2018).
- Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805 (2018).