Transfer learning with Transformer networks

Grégory Châtel

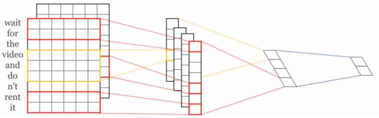
Disaitek
Intel Software Innovator

@rodgzilla
github.com/rodgzilla

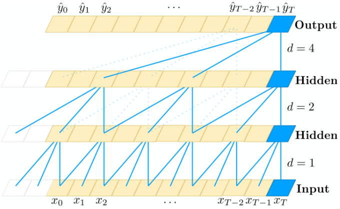
11/28/2018

Traditional architectures for NLP

CNN



Dilated CNN



RNN

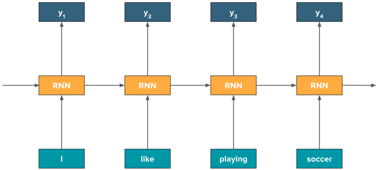
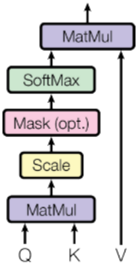


Image from https://techblog.gumgum.com/articles/deep-learning-for-natural-language-processing-part-2-rnns and

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

Attention mechanisms

Scaled Dot-Product Attention

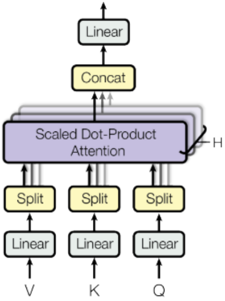


Q is the query vector, K is the key vector and V value vector.

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

Attention mechanisms

Multi-Head Attention

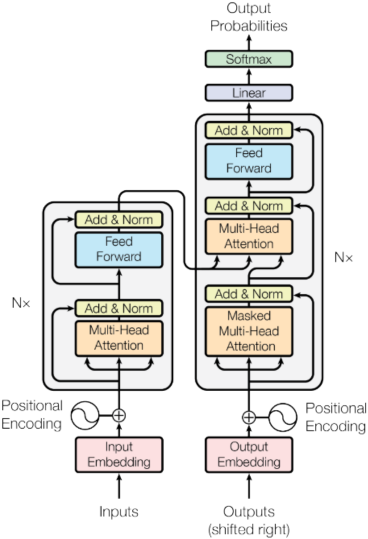


$$\begin{aligned}\mathrm{MultiHead}(Q, K, V) &= \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h) \\ \text{where} \quad \mathrm{head}_i &= \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

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

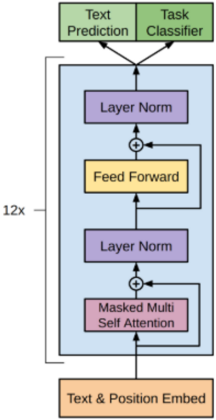
Transformer network

Original transformer



Transformer network

OpenAI multi-layer decoder



W_e is the token embedding matrix

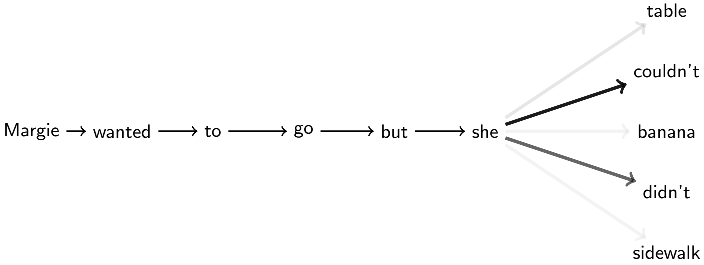
W_p is the position embedding matrix

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

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

Unsupervised pre-training task

Language modeling



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

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

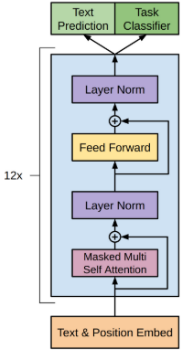
Dataset BooksCorpus (7000 books, $\sim$ 5GB of text),

Duration 1 month,

Hardware 8 GPUs.

Supervised fine-tuning

Multitask learning



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

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

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

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

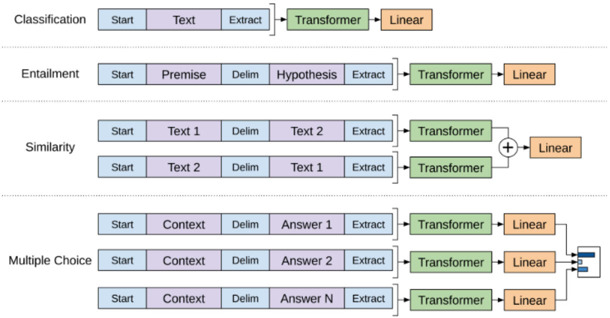
Final loss

Results on standard datasets

New state of the art on the following tasks:

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

```python
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
```

References

- 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).