

Transformers 2: Decoder, Limitations

Ling 575j: Deep Learning for NLP

C.M. Downey

Spring 2023

Announcements

- HW4 ref (/ sym-links) available
- Why a character-level language model for HW5?
 - Primarily: *compute* efficiency. For SST data:
 - Size of char vocab: ~70. Size of word vocab: ~13000.
 - Softmax layer sums over the whole vocab (for denominator); becomes very expensive!
 - NB: will talk today about “modern” approaches to solving this problem
 - Secondarily: very impressive! Still can work quite well (e.g. char LSTM LMs from those slides).
 - Third: may learn interesting phenomena below the word level (e.g. morpheme discovery).
 - NB: *hard* problem so models may not work as well as word-level, esp. small ones (e.g. HW5).
- See Ajda Gokcen’s [Treehouse talk](#) from last year on text granularity in NLP models

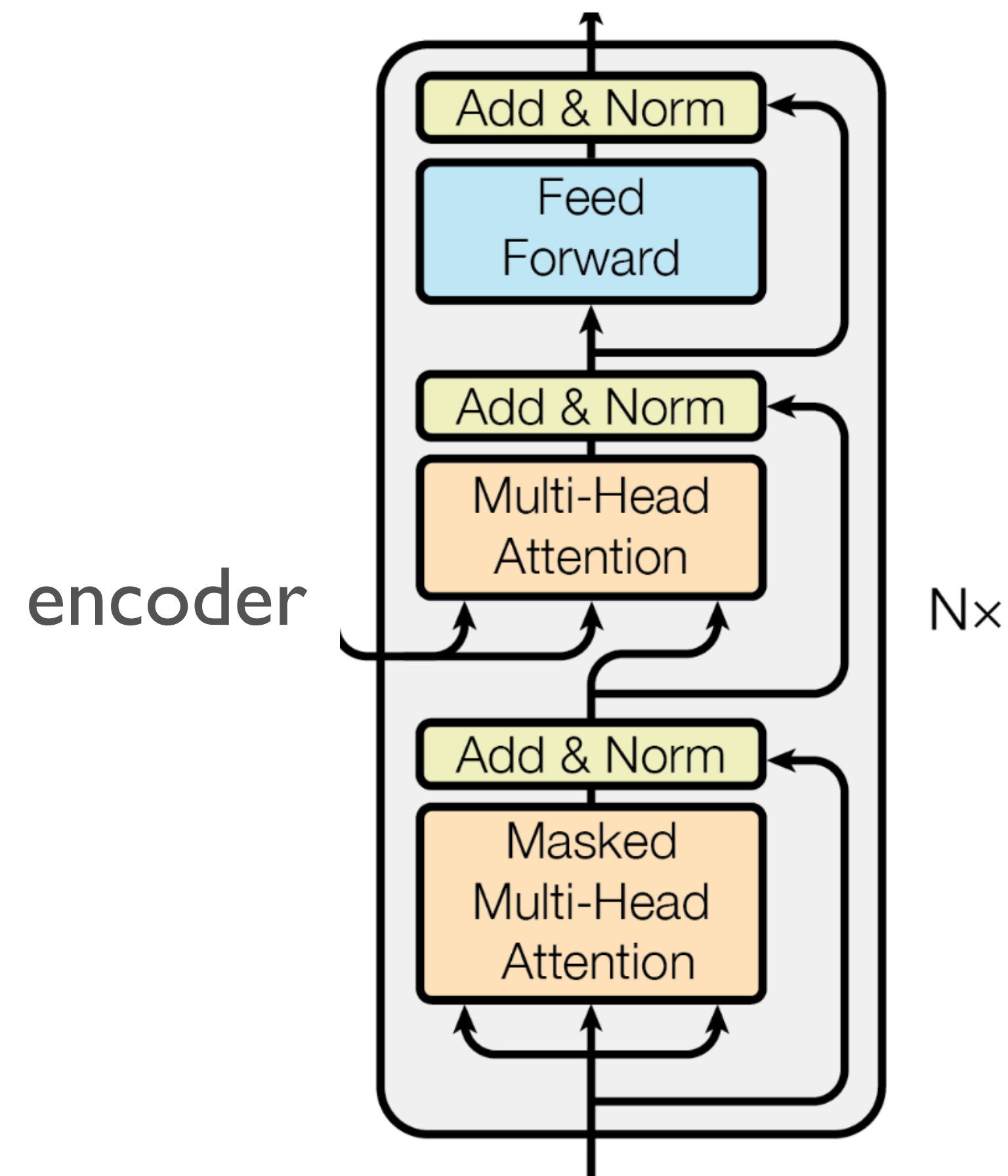
Today's Plan

- Wrap up Transformer Encoder
- Transformer Decoder
 - Attention Masks
- Limitations
 - Quadratic attention
 - Sequential generation
- Subword Tokenization

Transformer Decoder

Decoder Block

- Like the encoder, the decoder is many *blocks* stacked vertically
- Two slightly different ingredients:
 - *Masked* self-attention
 - Cross (encoder-decoder) attention



Masked Self-Attention

Masked Self-Attention

- Recall from seq2seq:
 - Decoder a kind of *conditional* language model
 - Predicts next tokens in output sequence, *given* the encoder representations
 - (Can also be used on its own as an unconditional LM; more later)
- Problem: self-attention “looks to the future”
 - Decoders should only be able to pay attention to *previous* positions

Masking Out the Future

- Key idea:
 - Use a “mask” to block out certain attention scores
- On the left:
 - Tokens in the rows (as queries) can *not* pay attention to the tokens in the columns (values) that are shaded in

| | <S> | Ceci | n' | est | pas | une | pipe |
|------|-----|------|----|-----|-----|-----|------|
| <S> | | | | | | | |
| Ceci | | | | | | | |
| n' | | | | | | | |
| est | | | | | | | |
| pas | | | | | | | |
| une | | | | | | | |
| pipe | | | | | | | |

Masking Out the Future

QK^T : total attention scores

$$\text{mask}_{ij} = \begin{cases} -\infty & j > i \\ 0 & \text{otherwise} \end{cases}$$

$$\text{MaskedAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{mask}\right) V$$

| | <S> | Ceci | n' | est | pas | une | pipe |
|------|-----|------|------|------|------|------|------|
| <S> | 0 | -inf | -inf | -inf | -inf | -inf | -inf |
| Ceci | 0 | 0 | -inf | -inf | -inf | -inf | -inf |
| n' | 0 | 0 | 0 | -inf | -inf | -inf | -inf |
| est | 0 | 0 | 0 | 0 | -inf | -inf | -inf |
| pas | 0 | 0 | 0 | 0 | 0 | -inf | -inf |
| une | 0 | 0 | 0 | 0 | 0 | 0 | -inf |
| pipe | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Masked Self-Attention

- In a nutshell:
 - Compute “raw” attention scores as before
 - Add a mask to “zero out” the future positions in a sequence
- As in the encoder:
 - This is one attention *head*, several used for multi-headed attention
 - Q, K, V are generated by applying learned matrices for each head

Cross-Attention

Cross-Attention

- Recall the original application of attention: allowing a decoder to attend to *all* of an encoder's representations, instead of just the final one
- How can we apply this form in Transformer-land?
 - What are the queries, keys, and values?

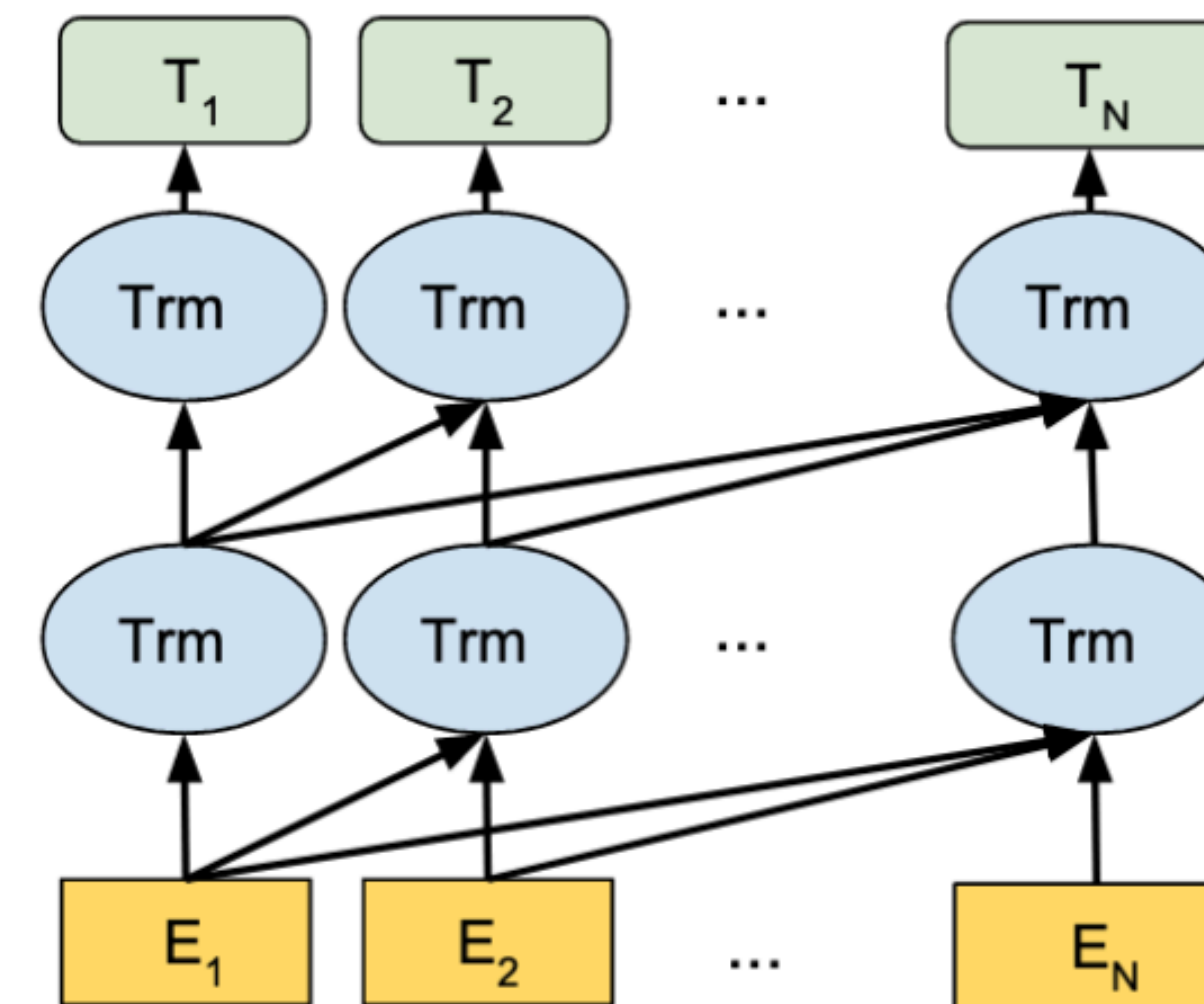
Cross-Attention

- Queries: decoder representations X
- Keys and values: top-layer encoder representations Z
- Learned weight matrices W_q, W_k, W_v as before

$$\text{CrossAttention} = \text{Attention} \left(XW_q, ZW_k, ZW_v \right)$$

Transformer Decoders

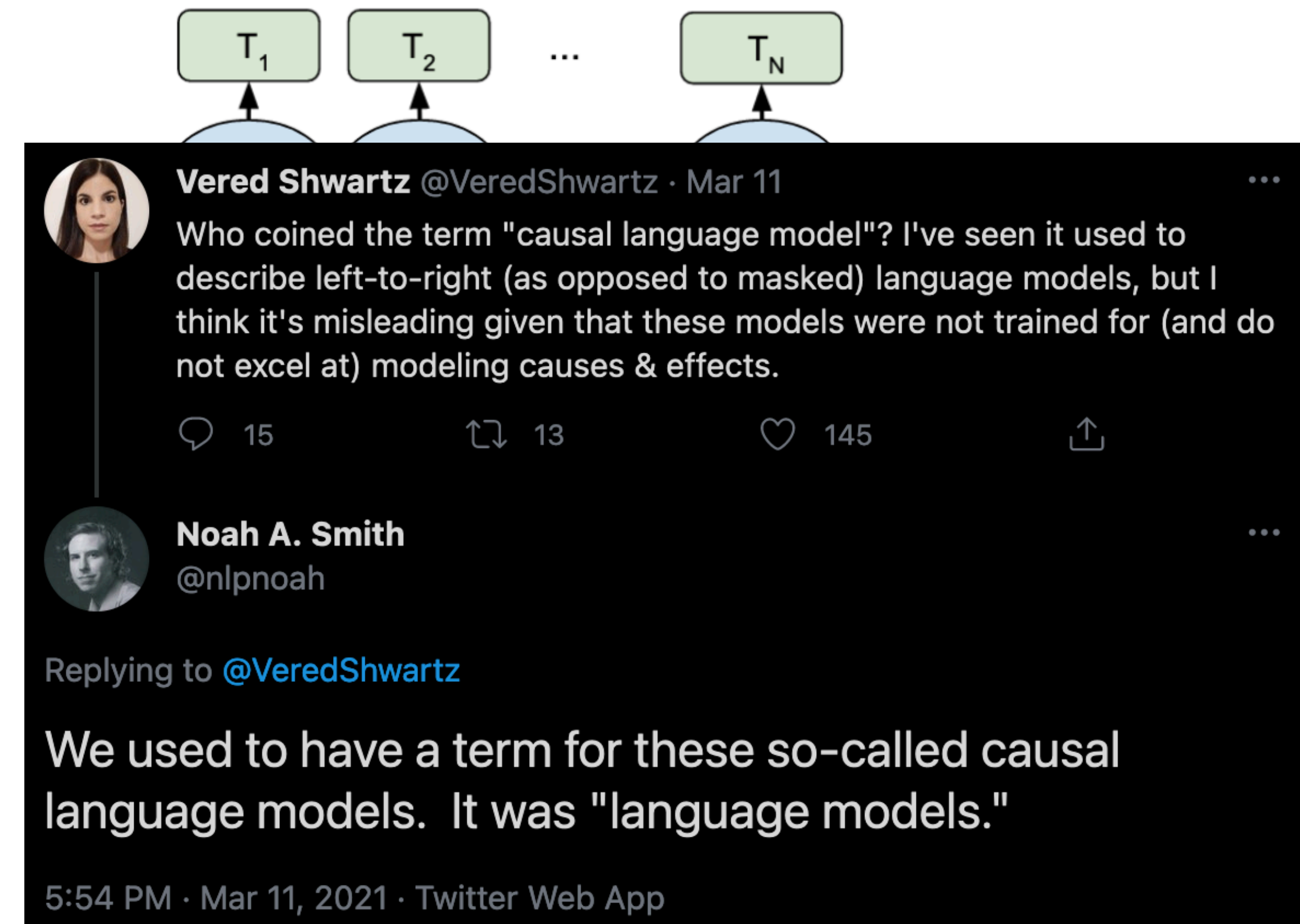
- Can be used any place you would use a decoder
- Masked attention prevents “peeking into the future”
- In seq2seq, for conditional language modeling, e.g.
 - Translation
 - Summarization
- On its own, as a “pure” language model
 - (people now call this “causal language modeling” sometimes)



[source](#)

Transformer Decoders

- Can be used any place you would use a decoder
- Masked attention prevents “peeking into the future”
- In seq2seq, for conditional language modeling, e.g.
 - Translation
 - Summarization
- On its own, as a “pure” language model
 - (people now call this “causal language modeling” sometimes)



Transformer LM (Decoder-only) Results

- Character-level:

- Used several auxiliary losses

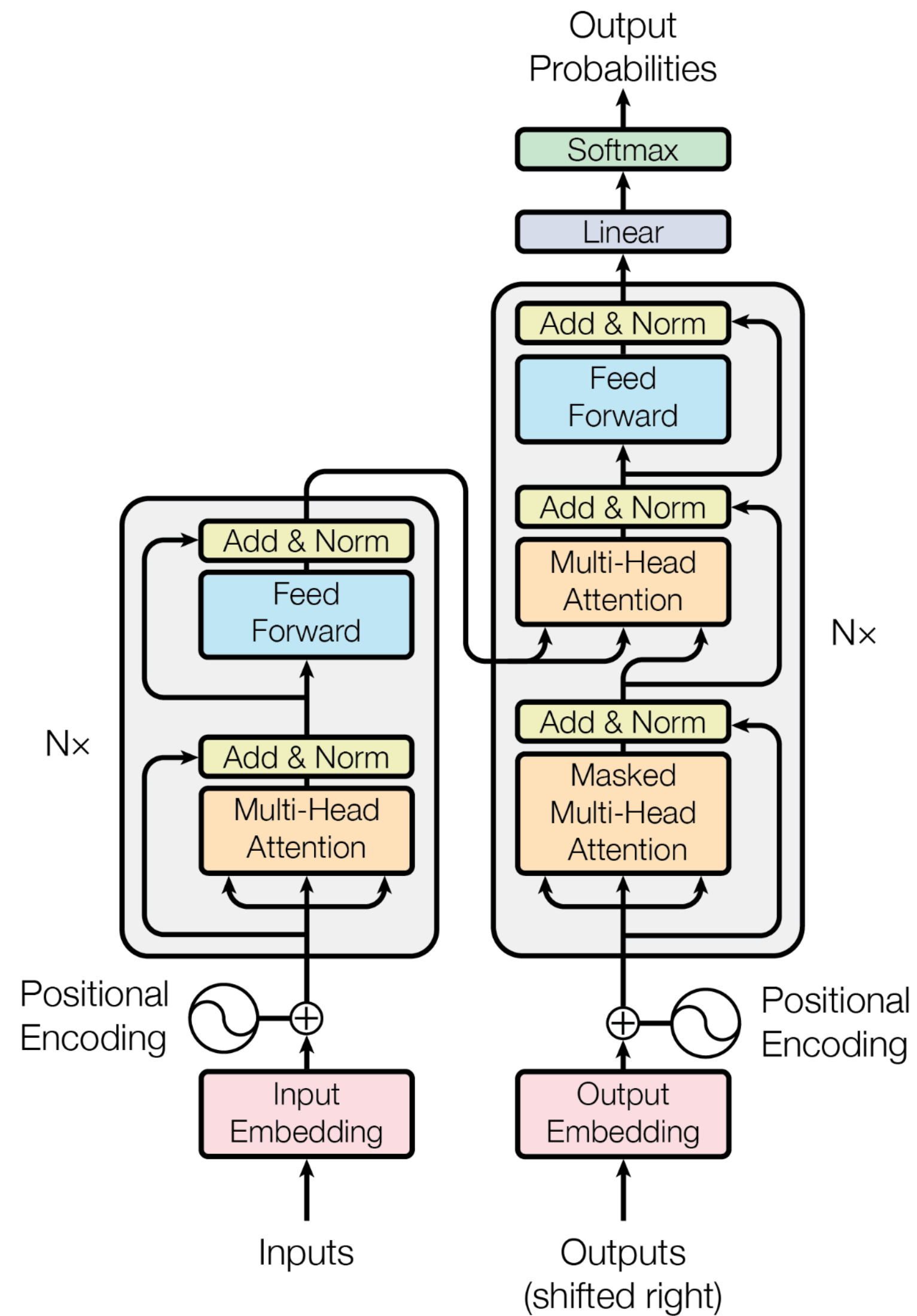
| Model | Parameters ($\times 10^6$) | | bpc |
|---|------------------------------|-----------|-------------|
| | train | inference | |
| LSTM (Cooijmans et al. 2016) | - | - | 1.43 |
| BN-LSTM (Cooijmans et al. 2016) | - | - | 1.36 |
| HM-LSTM (Chung, Ahn, and Bengio 2016) | 35 | 35 | 1.29 |
| Recurrent Highway (Zilly et al. 2016) | 45 | 45 | 1.27 |
| mLSTM (Krause et al. 2016) | 45 | 45 | 1.27 |
| T12 (ours) | 44 | 41 | 1.18 |
| T64 (ours) | 235 | 219 | 1.13 |
| mLSTM + dynamic eval (Krause et al. 2017) | 45 | - | 1.19 |

- GPT2 results (more next time)

- Zero-shot evaluation: trained on very large corpus, evaluated on standard benchmarks

| | WikiText2 (PPL) | PTB (PPL) | enwik8 (BPB) | text8 (BPC) | WikiText103 (PPL) | 1BW (PPL) |
|-------|--------------------|--------------|-----------------|----------------|----------------------|--------------|
| SOTA | 39.14 | 46.54 | 0.99 | 1.08 | 18.3 | 21.8 |
| 117M | 29.41 | 65.85 | 1.16 | 1.17 | 37.50 | 75.20 |
| 345M | 22.76 | 47.33 | 1.01 | 1.06 | 26.37 | 55.72 |
| 762M | 19.93 | 40.31 | 0.97 | 1.02 | 22.05 | 44.575 |
| 1542M | 18.34 | 35.76 | 0.93 | 0.98 | 17.48 | 42.16 |

Full Transformer Encoder-Decoder



Transformer Architecture Summary

- Main building block: *attention*
 - Encoder: self-attention
 - Decoder: *masked* self-attention
 - Decoder-encoder: cross-attention
- Position encodings/embeddings to inject information about sequence order
- Position-wise feed-forward networks for element-wise nonlinearities
- Residual connections + LayerNorm around every component

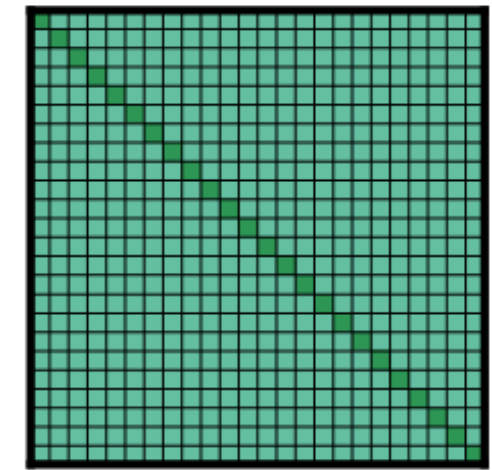
Transformers: Limitations

Quadratic Attention

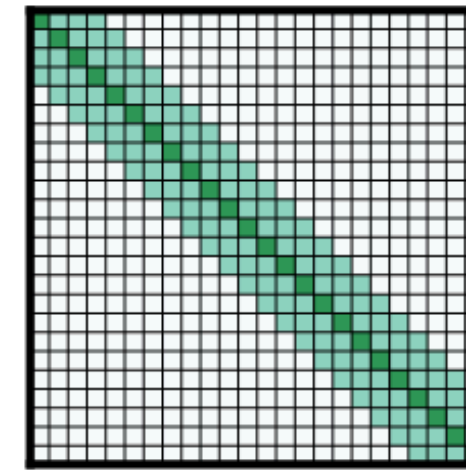
- Attention computes similarity scores between all pairs of tokens
 - QK^T : [seq_len, seq_len] shape
 - In other words, size of attention is $O(n^2)$
- Prevents scaling to *long sequences*
 - Document-level:
 - Summarization
 - QA
 - ...
- Big area of current research: linear(-ish) attention mechanisms.

Some Examples

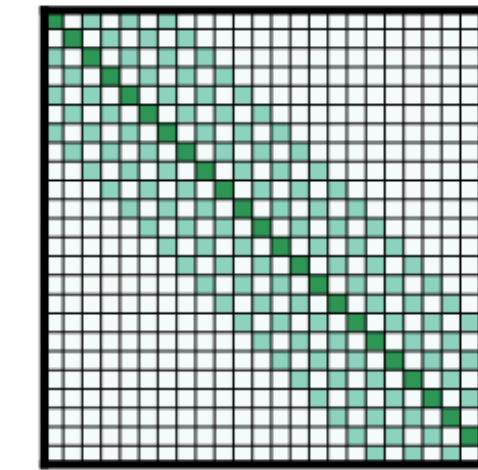
- Longformer:
- Carefully control positions attended to



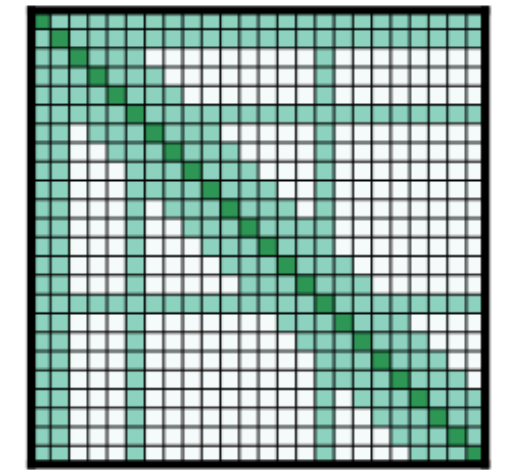
(a) Full n^2 attention



(b) Sliding window attention

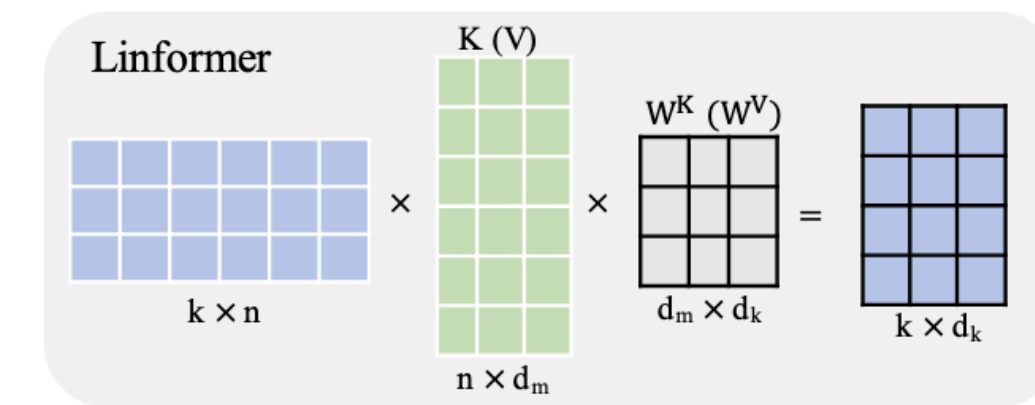
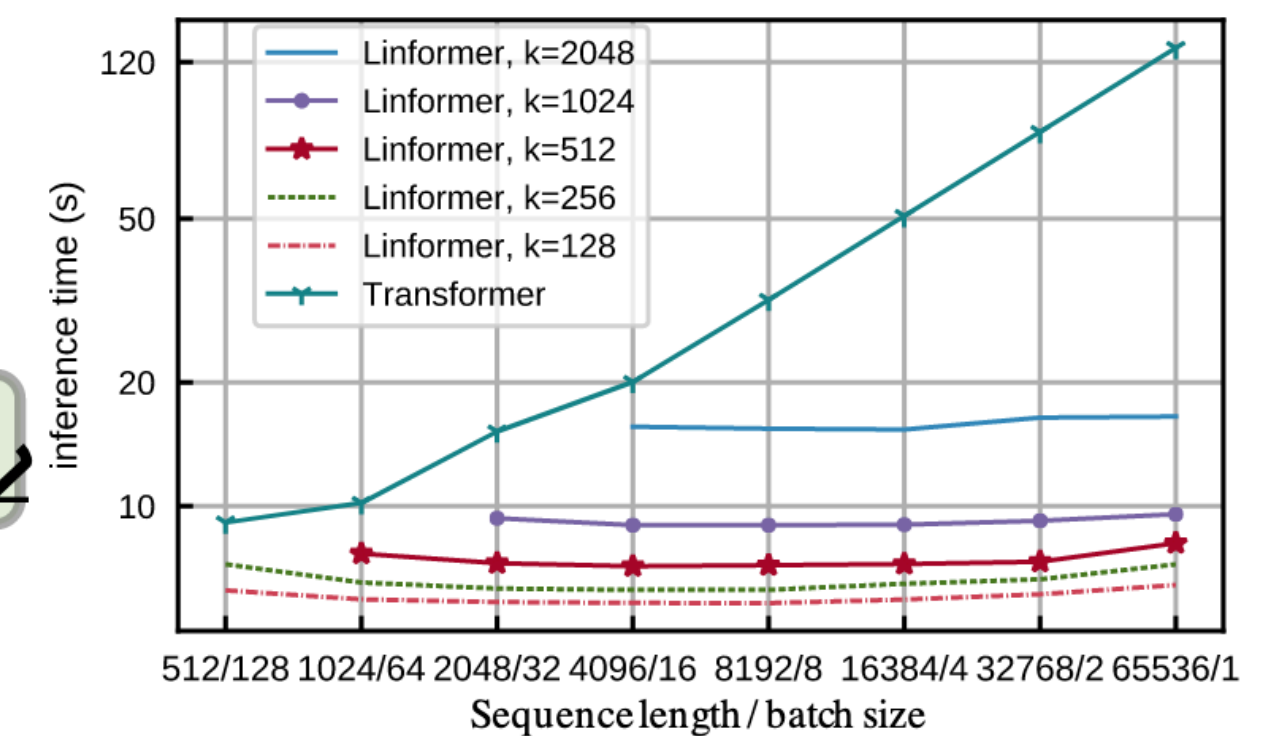
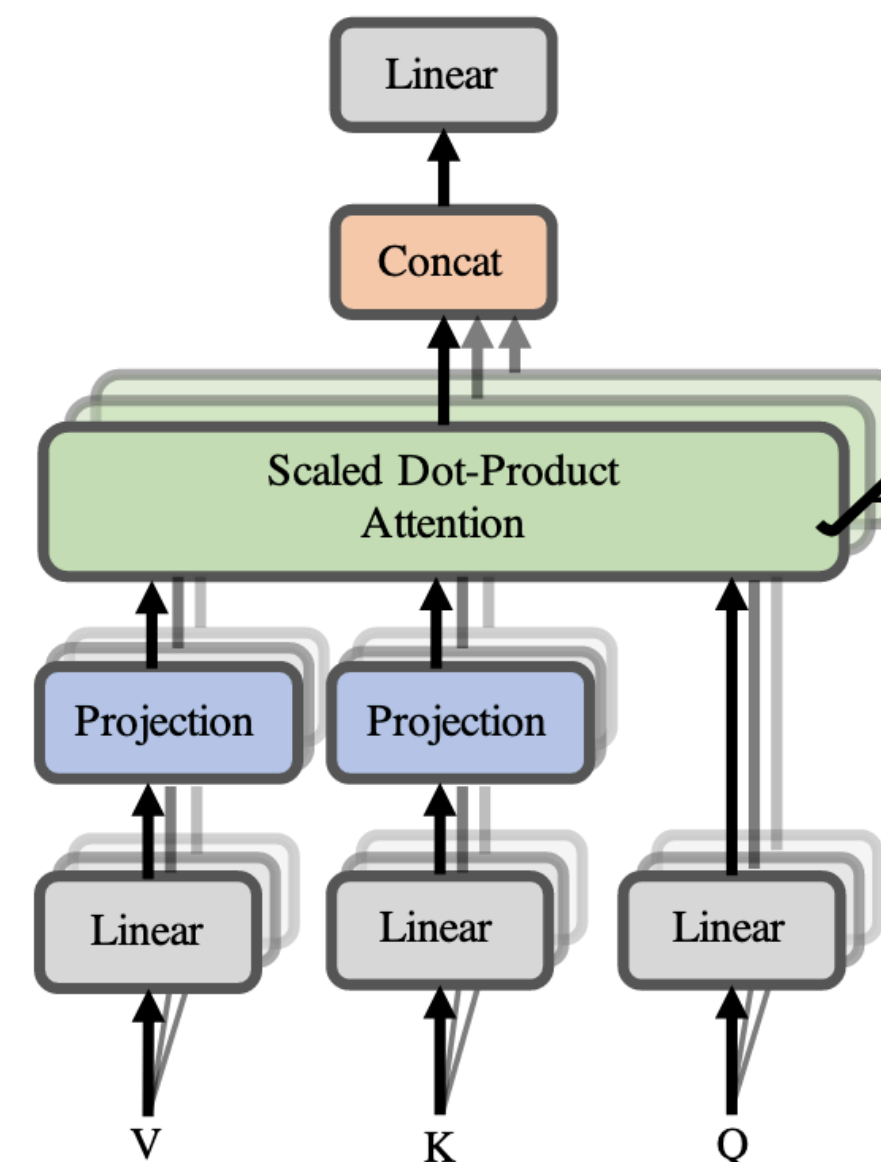


(c) Dilated sliding window



(d) Global+sliding window

- Linformer:
- Additional projection of Keys/ Values to smaller space
- $O(nk)$, with k a hyper-parameter
- Survey paper



Inference speed does not scale with seq length

Recurrence in Generation

- Recall the basic method for generating from a decoder:
 - Feed initial token (e.g. BOS, or just a word/character)
 - Generate probability over next tokens
 - Sample next token from this distribution
 - Repeat until [EOS | max length | other criterion]
- This loop is unavoidable during generation
 - Transformer's gains on paralellizability: work for training, vanish for generation
 - In fact, RNN decoders tend to be much faster at inference time

Mixed/Hybrid Architectures

- Encoder-decoder: a general architecture
 - In principle, any model of the right type can be encoder and/or decoder

- “The Best of Both Worlds” for NMT

- Transformer encoder + RNN decoder

- Google Translate’s newest version

| Encoder | Decoder | En→Fr Test BLEU |
|------------|------------|---------------------|
| Trans. Big | Trans. Big | 40.73 ± 0.19 |
| RNMT+ | RNMT+ | 41.00 ± 0.05 |
| Trans. Big | RNMT+ | 41.12 ± 0.16 |
| RNMT+ | Trans. Big | 39.92 ± 0.21 |

- “Transformer models have been demonstrated to be generally more effective at machine translation than RNN models, but our work suggested that most of these quality gains were from the transformer *encoder*, and that the transformer *decoder* was not significantly better than the RNN decoder. Since the RNN decoder is much faster at inference time, we applied a variety of optimizations before coupling it with the transformer encoder. The resulting hybrid models are higher-quality, more stable in training, and exhibit lower latency.”

Subword Tokenization

OOV and Vocab Size

- Word-level models:
 - Tokenize training data
 - Build vocabulary
 - Learn representations
- Two problems:
 - Cannot generalize at test time to OOV (out of vocab) words
 - (various subtleties, tricks, etc, but generally true)
 - Larger training data → larger vocabulary
 - Its own problems, e.g. very expensive softmax over vocab in decoders
 - (Or put a cap on vocab size, but then miss lower-frequency words entirely)

Finer Representation Levels

- One solution: *character-level* models
 - Pros:
 - Small vocabulary size
 - No (or very little) OOV
 - Cons:
 - Much harder learning problems; need to learn everything about words, on top of phrases, sentences, etc.
- In-between solution: *sub-word* tokenization
 - Split words into pieces, but don't go all the way down to character level
 - Many methods: WordPiece, BytePair Encoding (BPE), ...

WordPiece Embeddings

- Another solution to OOV problem, from NMT context (see [Wu et al 2016](#))
- Main idea:
 - Fix vocabulary size V in advance (e.g. 30k for BERT)
 - Choose V wordpieces (subwords) such that total number of wordpieces in the corpus is minimized
- Frequent words aren't split, but rarer ones are, e.g.:
- “Backpropagation was confusing at first, but now we grok it.”
 - [“Back”, “##prop”, “##ag”, “##ation”, “was”, “confusing”, “at”, “first”, “,”, “but”, “now”, “we”, “gro”, “##k”, “it”, “.”]

Next Time

- This wraps up our general overview of the Transformer architecture
- Next time: why they have become so dominant in NLP in the last several years
 - *Pre-training* and fine-tuning paradigm
 - General idea
 - Several examples
- Then: how to interpret/analyze NLP models, followed by a series of special guest lectures